RAIRO Operations Research
RAIRO Oper. Res. **37** (2003) 67-83
DOI: 10.1051/ro:2003014

COLUMN-GENERATION IN INTEGER LINEAR PROGRAMMING

Nelson Maculan[1], Marcos de Mendonça Passini[1], José André de Moura Brito[1] and Irene Loiseau[2]

Communicated by Michel Minoux

Abstract. We present an exact method for integer linear programming problems that combines branch and bound with column generation at each node of the search tree. For the case of models involving binary column vectors only, we propose the use of so-called geometrical cuts to be added to the subproblem in order to eliminate previously generated columns. This scheme could be applied to general integer problems without specific structure. We report computational results on a successful application of this approach to a telecommunications network planning problem.

Keywords. Column-generation, integer programming, branch-and-price.

1. Introduction

The techniques of column-generation were first presented in the early sixties as part of methods to solve linear programming (LP) problems with a huge number of variables. When the size of the problem does not allow one to store the entire

Received January, 2002.

[1] Programa de Engenharia de Sistemas e Computação – COPPE – Universidade Federal do Rio de Janeiro, Brasil; e-mail: {maculan,passini,britom}@cos.ufrj.br. Partially supported by CNPq, CAPES, FUJB, FAPERJ, PRONEX.

[2] Departamento de Computación – Facultad de Ciencias Exactas y Naturales – Universidad de Buenos Aires, Argentina; e-mail: irene@dc.uba.ar. Partially supported by grants UBACYT EX036, CONICET 644/98.

matrix of a LP problem, columns to enter the basis in the Simplex Method are generated by means of an auxiliary problem. Of course this is only possible for some problems with a special structure. In [15] Dantzig and Wolfe introduced those techniques in the context of their very well known decomposition algorithm, developed originally to deal with the limited storage capacity computers had at that time.

In their seminal works [27,28], Gilmore and Gomory used this approach to solve the cutting stock problem. They generate columns by solving in each step a knapsack subproblem. The original problem is an Integer Programming (IP) problem; column generation was used to solve the linear programming relaxation. This relaxation has shown to provide very good lower bounds in practice. Two decades later, Marcotte [37] showed that the optimal value is very often the rounded optimal (LP) relaxation.

Since then several other applications of column generation to obtain tighter relaxations to (IP) problems have been developed, which can be found in the literature.

Column generation methods to solve integer programming problems exactly appeared more recently. So called Branch-and-price methods combine Branch and Bound with column generation methods to solve the (LP) relaxation at each node. Some ad-hoc branching rules have been designed to keep the original structure of the pricing problem tractable, all along the branch and bound tree. Sometimes exact methods derive into heuristics when only a subset of the feasible columns in the nodes of the branch and bound tree is used or only the ones generated in the root node are kept.

Pioneering work in this sense was carried out in the eighties by Desrosiers *et al.* [24], Desrochers and Soumis [23], and Ribeiro *et al.* [48]. In these initial works they mentioned the difficulties of combining branching rules with column generation, and gave two possible ways of getting rid of some of them. One of the first Branch-and-price methods to appear in the literature was the one presented in [24] for the vehicle routing problem with time windows. Desrosiers *et al.* modeled the problem as a set partitioning problem where columns were generated using a modified shortest path algorithm which takes into account time windows. Branching rules that preserve this shortest path structure of the subproblem were used. When it was not possible to enumerate the entire branch and bound tree, they were able to obtain good lower bounds. A similar column generation approach to the Crew Scheduling Problem in Urban Transit that uses the set covering and constrained shortest path problems as subproblems was presented in [23].

In [48] an algorithm for a traffic assignment problem arising in a satellite switching system which can be modeled as a large scale set partitioning problem is presented. The algorithm combines column generation at each node with a ranking procedure of the columns that prevents regeneration of an already generated column at the subproblem. Optimality of the final integer solution is thus ensured. This ranking procedure is based on the particular structure of the subproblem, and it is based on an algorithm for obtaining the sequence of best assignments of a linear assignment problem. Since then numerous applications of column generation, to these and other combinatorial problems appeared in the literature. Several other successful applications of column generation techniques to vehicle routing problems are described in [7, 21, 22, 36, 49].

Ribeiro and Soumis [49] also presented a column generation approach for solving the linear programming relaxation of the multiple depot vehicle scheduling problem. They obtained a better bound than those appearing in the literature till then.

Loebel [36] developed a column generation method for the LP relaxation of vehicle scheduling problems in public transit that provides very good solutions for huge real cases.

In their comprehensive survey on constrained scheduling and vehicle routing problems Desrosiers *et al.* [22] stated that optimal algorithms based on Dantzig–Wolfe decomposition and column generation schemes have been shown to be the most powerful solution methodologies for that kind of problems.

Bramel and Simchi–Levi [7] intend to formalize an empirical conclusion that comes out of applications to VRPTW (vehicle routing problem with time windows) problems: when column generation techniques are applied to problems modeled as set covering ones, the method works well when the gap between the (LP) relaxation and the (IP) solution is small. They show that with some assumptions about the distribution of the customers, in the case of VRPTW this gap decreases to zero as the number of customers increases. In [57] Vance *et al.* present a decomposition algorithm for airline crew scheduling problem that improved bounds provided in previous works. Gamache *et al.* [26] present also a column generation heuristic method for solving a aircrew rostering problem. This method was able to obtain very good results on real large scale problems.

Vance *et al.* [56] implemented an exact algorithm for the one dimensional binary cutting stock problem. They propose branching rules that keep the structure of the subproblem tractable at each node (those rules had been previously proposed by Ryan and Foster [47]). In [55] Vance compares two algorithms for the same problem based on two different formulations of the master problem, and presents appropriate branching rules for each one.

Carvalho presents in [16] a different approach for the general, not necessary binary, case of the cutting stock problem. An arc-flow formulation with side constraints is proposed and solved by column generation techniques. The author claims that the algorithm works well when the gap between the first (LP) relaxation and the optimal integer value is less than one. He also concludes that it is sensitive to the width of the rolls. In [58] Vandeberck shows results of a column generation algorithm for the bin packing and cutting stock problems.

Mehrotra and Trick [41] developed a method for solving the graph coloring problem using the independent set formulation, which avoids some extended symmetry. They use customized branching rules that are similar to the ones in [56]. This approach leads to a heuristic when columns are generated only in the root node.

In [33] column generation was used in order to produce heuristics based on linear programming relaxations for the graph partitioning problem.

Bourjolly *et al.* [5] obtain lower bounds for the maximum stable set problem by means of column generation in the framework of a branch and bound method that does not use linear programming explicitly.

Savelsbergh [50] presents a branch-and-price algorithm to obtain optimal integer solutions for the generalized assignment problem using again a set partitioning formulation. He also shows that truncating the search tree in his procedure yields very good approximation algorithms.

In [60] Van Den Akker *et al.* describe a column generation method for solving the (LP) relaxation of a parallel machine scheduling problem. They obtain a very good lower bound and are able to solve some problems to optimality. Hansen *et al.* [29] propose primal and dual algorithms for mixed integer programming and their use to solve the probabilistic maximum satisfability problem.

Barnhart *et al.* [3] present an overview of column generation techniques for solving integer problems to optimality and a review of several classes of problems that were successfully solved in this way. They intend to generalize ideas that were successful in solving special problems by column generation. Vanderbeck and Wolsey [54], and [61], present an exact method that combines branch and bound with column generation. They develope an ad-hoc branching scheme and test the algorithm in three types of problems. In [52] the same approach is used to solve a problem arising in the design of telecommunications networks. In [59] Vanderbeck goes further in this direction and proposes a Dantzig–Wolfe decomposition based on the discretization of the integer polyhedron associated to a group of constraints. Furthermore, he proposes appropriate branching schemes. He tests his ideas on the cutting stock and cutting strip problems.

Although the list of articles on column generation mentioned above is by no means exhaustive, it is representative enough to conclude, as several authors already did [3, 42, 61], that most successful applications of column generation techniques happen in (IP) problems which can be modeled as set partitioning (or set covering) ones. In most of the quoted examples columns of the set partitioning problem have a well defined structure and it has been possible to develop pricing algorithms (exact or heuristic) to identify them. This formulation also enables one usually to handle good branching rules compatible with pricing algorithms and keeping the search tree balanced.

Some articles are also found in the literature that report methods that combine column generation with branch and cut. In [44] Nemhauser and Park use a matching formulation of the edge coloring problem and propose an algorithm which combines a simple separation routine for recognizing odd circuit constraints with a pricing algorithm for the weighted matching problem.

In [2] a multicommodity flow problem is solved by combining column and row (cuts) generation. An algorithm is described which combines a pricing algorithm, branching rules and cut generation that are mutually compatible along the branch and bound tree. Column generation and branching rules stem from a generalization of previous works, and lifted cover inequalities are added to the (LP) in each node.

In this work we present a general scheme that proved to be successful in solving two problems appearing in network design, and can be applied to other combinatorial problems for which the subproblem has no particular structure. We propose a special way of getting rid of columns that we do not want to enter the basis when working with bounded variables, and we show how the columns generated at the previous nodes of the branch and bound tree can be reused.

We also propose, for the binary case, a geometric cut to be added to the subproblem to avoid a column to be regenerated in one of the branchs of the tree when the clasical 0-1 branching rule is used. Although this approach could be used to solve any binary problem it looks particularly suitable for a problem for which there are not apparent branching rules compatible with the subproblem structure, or for problems where we cannot generate the k-best solutions of the subproblem easily, as it is done in [48]. This cut has been implemented in branch-and-price methods for a network design problem [40] and for the Steiner tree packing problem [51]. In the next section we describe how to use a column-generation technique in the framework of the simplex algorithm for linear programming problems with bounding variable constraints following the ideas of the COLGEN algorithm presented in [39]. Section 2 presents a linear inequality that can be used, when the matrix of the linear problem is binary, to avoid forbidden columns to be generated. A branch-and-bound scheme using this column-generation algorithm is presented in Section 3. In Section 4 we briefly report computational results of the network design problem we solved with this method. The last section is devoted to some conclusions.

2. A column-generation method for linear programming with bounded variables

In order to solve IP problems by a branch and bound method, we need to solve a LP problem at each node of the branch and bound tree. We assume that this LP problem is suitable to be solved using column generation. We consider the following linear programming problem:

$$(LP): \text{ minimize } z = \sum_{j=1}^{p} c_j x_j \tag{1}$$

subject to:

$$\sum_{j=1}^{p} a_j x_j = b, \tag{2}$$

$$0 \leq x_j \leq d_j, \; j = 1, 2, ..., p \tag{3}$$

where $a_j \in \mathcal{R}^m,\ b \in \mathcal{R}^m,\ b \geq 0,\ d = (d_1\ ...\ d_p)^T \in \mathcal{R}^p,\ c_j \in \mathcal{R}$. When all $c_j,\ a_j$ and d_j are known *a priori*, this problem can be solved using the very well known upper bounding method proposed by Dantzig [10, 13, 14, 35]. We consider that a_j belongs to a finite set $\mathcal{K}$, and $|\mathcal{K}| = p$. To write explicitly all this (LP) problem can be prohibitive for problems with a huge integer number p of columns. As an exemple we can consider $\mathcal{K}$ the set of 0-1 vectors associated with all spanning trees of a complete graph with n vertices, in this case $|\mathcal{K}| = n^{n-2}$. In this case we would like to solve (LP) using column-generation techniques in the context of the simplex method. For that we have to suppose $c_j = f(a_j)$ and $d_j = g(a_j)$, where $f : \mathcal{K} \to \mathcal{R}$ and $g : \mathcal{K} \to \mathcal{Z}_+$. Following the usual notation, see [10], let $B = (a_{B(1)}\ a_{B(2)}\ ...\ a_{B(m)})$ be an m by m matrix with $\det(B) \neq 0$ and N a m by $p - m$ matrix formed with the a_j columns which are not in B. Let $x_B = (x_{B(1)}\ x_{B(2)}\ ...\ x_{B(m)})^T$ and x_N be the vector associated with the columns of N. Thus (2) can be written as follows: $Bx_B + Nx_N = b$, then we have

$$x_B = B^{-1}b - B^{-1}Nx_N. \tag{4}$$

Now let N_2 be the matrix whose a_j columns of N are associated with $x_j = d_j$, and N_1 the matrix whose columns of N are associated with $x_j = 0$. We can present (4) as follows:

$$x_B = B^{-1}b - B^{-1}N_1x_{N_1} - B^{-1}N_2x_{N_2}. \tag{5}$$

Let $\bar{x}_B = B^{-1}b$, and $\hat{x}_B = \bar{x}_B - B^{-1}N_2\bar{x}_{N_2}$, where $\bar{x}_{N_2} = d^T_{N_2}$. If $0 \leq \hat{x}_{B(i)} \leq d_{B(i)},\ i = 1, 2, ..., m$, then B is a feasible basic solution to (LP). Let $u = c_B^T B^{-1}$, where $c_B = (c_{B(1)}\ c_{B(2)}\ ...\ c_{B(m)})^T \in \mathcal{R}^m$, $z_j = ua_j$, and $\bar{z}_j = z_j - c_j$. We will assume that $\bar{z}_j \geq 0$, for such j associated with $x_j = d_j$. It is always possible to have this situation if (LP) has a not empty solution set. If we have $\bar{z}_j < 0$ for $x_j = d_j$, then solving (LP) without the columns in N_1, we obtain a new N_2 for which $\bar{z}_j \geq 0$ for all $x_j = d_j$.

We know that at each iteration of the simplex method, we need to solve the following pricing subproblem (oracle or pricing algorithm) in order to recognize an optimal basic solution of (LP), or to determine a column to enter the basis:

$$(SP): \text{ maximize } \ ua - f(a)$$

subject to:

$$a \in \mathcal{K}.$$

Using this scheme we may need to find not only the optimal solution to this subproblem but the second one, third one, and so on, because we do not want columns in N_2 to be chosen. Let a_k be an optimal solution for (SP), that is $\text{val}(SP) = ua_k - f(a_k)$, where val(.) denotes the optimum value of the objetive function in $(\cdot)$. We will need to be able to compute the $s-$best solution of (SP) for $s = 1, 2, ...$ such that $\text{val}(s-\text{best}) \geq \text{val}([s+1]-\text{best})$. Thus $\text{val}(1-\text{best}) = \text{val}(SP)$.

Let $I = \{1,\ 2,\ \dots\ ,p\}, I_{N_1} = \{j \mid a_j \in N_1\},\ I_{N_2} = \{j \mid a_j \in N_2\}$, and $I_B = \{j \mid a_j \in B\}$. We will solve our LP problem by means of solving iteratively:

$$(LP, B, N_2):\ \text{minimize } ub - \sum_{j \in N_2} (z_j - c_j)x_j$$

subject to:

$$x_B = \bar{x}_B - \sum_{j \in N_2} y_j x_j,$$

$$0 \le x_{B(i)} \le d_j,\ i = 1, 2, ..., m$$

$$0 \le x_j \le d_j,\ j \in N_2$$

where $u = c_B^T B^{-1},\ z_j = ua_j,\ c_j = f(a_j),\ d_j = g(a_j)$ and $y_j = B^{-1}a_j$.

So given a feasible basic solution B for (LP), the COLGEN algorithm for this problem can stated as:

```
BEGIN
        solve $(LP, B, I_{N_2})$;
        $s := 0$;
1       $s := s + 1$;
        $t :=$ val($s -$ best) $:= u\bar{a} - f(\bar{a})$; (oracle)
        if $\bar{a} \in N_2$, goto 1;
        if $t \le 0$, an optimal solution found STOP;
        $\bar{a} \in I_{N_1}$ will enter $B$ or $N_2$; (simplex pivot rule)
        solve $(LP, B, I_{N_2})$;
        $s := 0$;
        goto 1;
END.
```

3. An inequality to avoid regenerating forbidden columns when matrix A is binary

When the set of all a in $\mathcal{K}$ can be described by:

$$Ka \le h, \tag{6}$$

$$a \in \{0, 1\}^m, \tag{7}$$

where $K \in \mathcal{R}^{q \times m}$ is a known matrix and $h \in \mathcal{R}^q$ is a given vector, we propose a special way to avoid to generate a forbidden column to enter the basis.

We will use a linear inequality that eliminates a given point of $\{0, 1\}^m$. Let $\bar{a} \in \{0, 1\}^m$, we wish to construct an inequality to eliminate just $\bar{a}$ in (6) and (7). That is given $\mathcal{B} = \{0, 1\}^m - \{\bar{a}\}$ how to find a facet of the convex hull of $\mathcal{B}$, which keeps out $\bar{a}$?. All points of a sphere centered in $\bar{a}$ with radius equal to 1 satisfy the following equation: $\sum_{j=1}^m (a_j - \bar{a}_j)^2 = 1$. All the neighbors of $\bar{a}$ in the cube whose vertices are all points of $\{0,1\}^m$ belong to the surface of this sphere. If we want to consider all points of $\mathcal{B}$ we will write:

$$\sum_{j=1}^{m}(a_j - \bar{a}_j)^2 \geq 1. \tag{8}$$

As we have $a_j^2 = a_j$,

$$\sum_{j=1}^{m}(a_j - \bar{a}_j)^2 \geq 1$$

implies

$$\sum_{j=1}^{m}(a_j^2 - 2\bar{a}_j a_j + \bar{a}_j^2) \geq 1$$

then

$$\sum_{j=1}^{m}(1 - 2\bar{a}_j)a_j \geq 1 - \sum_{j=1}^{m}\bar{a}_j. \tag{9}$$

It is easy to see that (9) is a facet of the convex hull of $\mathcal{B}$, which keeps out $\bar{a}$.

We include this inequality in the COLGEN algorithm which in this case can be presented as follows:

```
BEGIN
1     solve($LP, B, I_{N_2}$);
      solve($SP$), including $\sum_{j=1}^{m}(1-2\bar{a}_j)a_j \geq 1 - \sum_{j=1}^{m}\bar{a}_j, \forall \bar{a} \in N_2$;
      val($SP$) := $u\hat{a} - f(\hat{a})$; (oracle)
      if val($SP$) $\leq 0$, an optimal solution is found STOP;
      $\hat{a}$ will enter B or N$_2$; simplex pivot rule
      goto1;
END.
```

4. Solving integer programming with COLGEN

We define an integer program as follows:

$$(IP): \text{ minimize } \sum_{j=1}^{p} c_j x_j$$

subject to:

$$\sum_{j=1}^{p} a_j x_j = b,$$
$$0 \leq x_j \leq d_j,\ j = 1, 2, ..., p,$$
$$x_j \in \mathcal{Z},\ j = 1, 2, ..., p,$$

where $c_j,\ a_j,\ b$ and d_j are defined as in (1) to (3). We solve (IP) using branch-and-bound techniques, see [12] and [45] and at each node of the enumerative scheme the column-generation algorithm will be applied.

We consider (LP) the linear relaxation of (IP), that is, we do not consider in (LP) constraints $x_j \in \mathcal{Z},\ j = 1, 2, ..., p$.

At the initial node we solve (LP) if its optimal solution is integer then we have solved (IP), otherwise we have a k such that $\hat{x}_{B(k)}$ is not integer. Thus we will branch this node solving two linear programming problems:

- $(LP) \cap \{x_{B(k)} \leq \lfloor \hat{x}_{B(k)} \rfloor\}$, and
- $(LP) \cap \{x_{B(k)} \geq \lfloor \hat{x}_{B(k)} \rfloor + 1\}$,

where $\lfloor r \rfloor$ is the largest integer $\leq r$.

We suppose to be at node i of the branch-and-bound tree and its associated linear programming problem is under the following form

$$(LP_i): \text{ minimize } \sum_{j=1}^{p} c_j x_j$$

subject to:

$$\sum_{j=1}^{p} a_j x_j = b,$$
$$\alpha_j \leq x_j \leq \beta_j,\ j \in S_i,$$
$$0 \leq x_j \leq d_j,\ j \in I - S_i,$$

where S_i is the set of the indices associated with all generated columns along the current branch, including the columns in B at node i. We can have $\alpha_j = 0$ or $\beta_j = d_j$ for some $j \in S_i$. An initial feasible basic solution for (LP_i) is a matrix B for which $\alpha_{B(i)} \leq \bar{x}_{B(i)} \leq \beta_{B(i)},\ i = 1, 2, ..., m,\ \bar{x}_j = \alpha_j$ or $\bar{x}_j = \beta_j,\ j \in S_i - I_B$, and $\bar{x}_j = 0,\ j \in I - S_i$.

We consider now a feasible basic solution of (LP_i) associated with an optimal basic solution of

$$(LXP_i): \text{ minimize } \sum_{j \in S_i} c_j x_j$$

subject to:

$$\sum_{j \in S_i} a_j x_j = b,$$
$$\alpha_j \leq x_j \leq \beta_j,\ j \in S_i,$$

where $\bar{z}_j = 0,\ j \in I_B;\ \bar{z}_j \leq 0$ for $\bar{x}_j = \alpha_j$ and $j \in S_i - I_B;\ \bar{z}_j \geq 0$ for $\bar{x}_j = \beta_j$ and $j \in S_i - I_B$.

If (LP_i) is a leaf of the branch-and-bound tree then we resort to backtracking, otherwise we choose a $k \in I_B$ such that $\hat{x}_{B(k)}$ is not integer, and we branch as follows:

- $(LP_{i+1}) := (LP_i) \cap \{x_{B(k)} \leq \lfloor \hat{x}_{B(k)} \rfloor\}$, and
- $(LP_{i+2}) := (LP_i) \cap \{x_{B(k)} \geq \lfloor \hat{x}_{B(k)} \rfloor + 1\}$.

We can take linear programming problems considering just the columns generated up to reach node i.

- $(LXP_{i+1}) := (LXP_i) \cap \{x_{B(k)} \leq \lfloor \hat{x}_{B(k)} \rfloor\}$, and
- $(LXP_{i+2}) := (LXP_i) \cap \{x_{B(k)} \geq \lfloor \hat{x}_{B(k)} \rfloor + 1\}$.

It is important to note that an optimal basic solution of (LXP_i) is still a dual feasible basic solution for (LXP_{i+1}) and (LXP_{i+2}).

Starting the (dual) simplex algorithm using an optimal basic solution of (LXP_i) we can solve easily (LXP_{i+1}) and (LXP_{i+2}), see [10, 17] or [38]. Unfortunately (LXP_{i+1}) can be empty and (LP_{i+1}) is not. The same can be assumed for (LXP_{i+2}) with respect to (LP_{i+2}).

When (LXP_{i+1}) is not empty we have an optimal basic solution for it, and this optimal solution will be the initial feasible basic solution for (LP_{i+1}). Then we use COLGEN algorithm to solve (LP_{i+1}). The same can be done for (LP_{i+2}), from an optimal basic solution for (LXP_{i+2}).

If (LXP_{i+1}) has no feasible solution we have to solve (LP_{i+1}) starting the simplex algorithm using an artificial solution. We introduce an artificial variable r in the k-row as follows:

$$x_{B(k)} + r = \bar{x}_{B(k)} - \sum_{j \in I_{N_1}} y_{kj} x_j - \sum_{j \in I_{N_2}} y_{kj} x_j.$$

We put $x_{B(k)} = \lfloor \hat{x}_{B(k)} \rfloor$, then

$$\bar{r} = \bar{x}_{B(k)} - \sum_{j \in I_{N_1}} y_{kj} \alpha_j - \sum_{j \in I_{N_2}} y_{kj} \beta_j - \lfloor \hat{x}_{B(k)} \rfloor > 0.$$

As $a_{B(k)}$ is in the basis B we have $y_{B(k)} = B^{-1} a_{B(k)} = e_k$. If we consider $I_{N_2} := I_{N_2} \cup \{B(k)\}$, and $\beta_{B(k)} = \lfloor \hat{x}_{B(k)} \rfloor$, then we can solve the following linear programming problem using COLGEN algorithm:

$$(AP_1): \text{ minimize } r$$

subject to:

$$\sum_{j=1}^{p} a_j x_j + a_{B(k)} r = b,$$

$$\alpha_j \leq x_j \leq \beta_j, \; j \in S_i,$$

$$0 \leq x_j \leq d_j, \; j \in I - S_i, \text{ and } r \geq 0.$$

Let B a primal feasible basic solution for (AP_1). If $\text{val}(AP_1) = 0$ we have an initial feasible basic solution for (LP_{i+1}). Otherwise (LP_{i+1}) is empty and the $(i+1)$-node is closed, and backtracking takes place.

For solving (LP_{i+2}) when (LXP_{i+2}) is empty we proceed the same way.

5. Computational results for a network design problem

The above-described techniques were successfully applied to a problem that arises in telecommunications network planning which was presented in detail in [40, 46] and [6].

In recent years, studies about network dimensioning and survivability focused mainly on ring based and mesh based architectures. Most works treated both models separately. But several companies deploy networks based on rings, and at the same time they have many meshed networks in operation and even some curently being built. In large metropolitan centers it is easy to identify clusters of nodes generating big data traffic among themselves. Meanwhile peripheral areas of the same network may present reduced traffic. If the area under study is already served by a meshed network it is possible to deal with the growth of traffic demands by superimposing self-healing rings on the existing mesh. This is called the two-level architecture network design solution.

The *mesh* part of the problem can be modeled quite well as a capacitated multicommodity flow problem with expandable arc capacities. This will be the initial master problem. The other way the flow can circulate on the net is on an additional net of *rings.* Only flow between nodes on each ring is allowed on it.

The complete formulation of the problem is a large mixed integer linear programming one and includes a column associated with each possible ring (see [40] for details). As it is not possible to have all the columns available since the beginning, a column generation procedure (SP) was chosen to solve the problem. The problem of generating each column can be modeled as an integer programming problem.

Initially in [40], four instances of the problem were solved this way, two real problems corresponding to 7-nodes and two corresponding to 10-node networks. The program was written in C, and the XPRESS-MP library was used. All experiments have been carried on a Pentium III 450 MHz computer. These results are summarized in Tables 1 and 2; problems with 20, 30, 40 and 50 nodes were also artificially generated. The number of traffic requirements and the values of the requirements were randomly generated. The maximum rings capacities were taken as 16. Detailed results for these new problems and one of the previous 10 node problems (with capacities of rings equal to 16) are presented in Table 3. In all cases the exact optimum was found.

The characteristics of each problem, the cost of the solution of the problem that only consider the mesh architecture and the number of generated support cycles are first shown in the three tables. The next lines correspond to the Branch and Bound solution of the problem where only a subset of the rings is considered. Then the total number of generated columns, active columns (columns whose associated variables have positive values at the optimal solution), open nodes used in the complete branch-and-price procedure are then shown as well as the number of the geometric cuts proposed in Section 3 which were necessary to add in each problem. We also show the gap between the optimum and the linear and Branch and Bound solutions. Looking at Table 2, we can see that using column generation for the

TABLE 1. Results on the 7 node network.

Ring capacity	**12**	**16**
Edges	11	11
Demands	10	10
Cost of the mesh network	50	50
Support cycles generated	15	15
Linear solution with Colgen	10.067	8.740
CPU time (sec.)	9	8
B&B Integer solution	12	10
Generated columns	25	24
Active columns	2	2
ring cost	10	10
mesh cost	2	0
CPU time (sec.)	1	1
Optimum (branch-and-price)	11	10
Generated columns	43	50
Active columns	3	2
Open nodes	3	5
CPU time (sec.)	10	7
Max depth in Branch Bound tree	2	1
Geometric cuts	2	1
GAP(%): Optimum and linear solution	9.27	14.42
GAP(%): BB Integer solution and optimum	9.09	0.00
Master problem		
Number of constraints	65	65
Number of columns	231	231
Slave problem		
Number of constraints	65	65
Number of columns	44	44

linear relaxation of the problem with ring capacity 12, we generated 88 columns, and we obtain a solution cost equal to 24.403. Using only these columns we started a branch-and-bound procedure and we obtain a solution with a value of 29 that provides an upper bound for the solution of the original problem. Among these 88 generated columns, 7 were related to nonzero variables, which means that only 7 rings with capacity 12 were used in the final solution of this branch and bound procedure. Then starting from the linear relaxed solution mentioned above, a branch-and-price method as the one described in Section 4 was used.

Table 2. Results on the 10 node network.

Rings capacities	**12**	**16**
Edges	22	22
Demands	36	36
Cost of the mesh network	152	152
Support cycles generated	169	169
Linear solution with Colgen	24.403	21.958
CPU time (sec.)	48	57
B&B Integer solution	29	27
Generated columns	88	78
Active columns	7	6
CPU time (sec.)	6	3
Optimum (branch-and-price)	27	24
Generated columns	4450	11104
Active column	6	6
Open nodes	540	685
CPU time (hours)	2	2.5
Max depth in Branch Bound tree	39	48
Geometric cuts	154	466
GAP(%): Optimum and linear solution	10.64	9.30
GAP(%): BB Integer solution and optimum	7.41	12.50
Master problem		
Number of constraints	291	291
Number of columns	1606	1606
Slave problem		
Number of constraints	201	201
Number of columns	128	128

6. Conclusions

The solution of combinatorial optimization problems for which the columns of the constraint matrix belong to a huge finite set can be found using the techniques we described in this paper. Solving (SP) problem may be difficult. The use of (LXP_i) instead of (LP_i) may accelerate the branch-and-bound implementation as was the case in the application mentioned in Section 5. We observe that, when the matrix A of the problem is binary, the inequality presented in Section 3 may be used in the context of a branch-and-price method to avoid a column to reenter the successors of the node where it has been deleted.

TABLE 3. Results on the 10, 15, 20, 30, 40 and 50 node networks.

Nodes	**10**	**15**	**20**	**30**	**40**	**50**
Edges	22	25	40	58	60	80
Demands	36	20	25	50	45	70
Rings capacities	16	16	16	16	16	16
Cost of the mesh network	152	108	142	252	230	380
Support cycles generated	169	1408	13694	12083	78543	145117
Linear solution with Colgen	21.96	18.33	25.74	46.71	46.64	72.27
CPU (time secs)	67	46	61	507	401	1099
B&B Integer solution	27	20	27	52	50	75
Generated columns	78	73	84	224	225	494
Active columns	6	4	5	8	8	13
CPU (time secs)	3	1	1	8	6	457
Optimum (branch-and-price)	24	19	26	48	48	74
Generated columns	11104	105	124	1098	2471	7021
Active columns	6	4	5	8	8	13
Open nodes	685	6	4	55	140	327
CPU (time secs)	9000	22	32	1704	5128	36345
Max depth in Branch Bound tree	48	3	2	4	10	18
Geometric cuts	466	3	2	28	75	170
GAP(%)						
Optimum and linear solution	9.30	3.64	1.03	2.75	2.91	2.39
B&B Int. solution and optimum	12.50	5.26	3.85	8.33	4.17	1.35
Master problem						
Number of constraints	291	280	481	1414	1762	3417
Number of columns	1606	1353	2550	5858	8281	17202
Slave problem						
Number of constraints	201	132	166	311	306	451
Number of columns	128	90	115	210	215	310

REFERENCES

[1] R. Anbil, J. Forrest and W. Pulleyblank, Column generation and the airline crew pairing problem. *DOC. Math. J. DMV* (1998) 677-686.

[2] C. Barnhart, C. Hanne and P.H. Vance, Using branch-and-price and cut to solve origin destination integer multicommodity flow problems. *Oper. Res.* **48** (2000) 318-326.

[3] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh and P.H. Vance, Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.* **46** (1998) 316-329.

[4] R. Borndorfer, M. Grotschel and A. Lobel, *Scheduling duties by adptive column generation.* ZIB-Report 01-02 (2001).

[5] J. Bourjolly, G. Laporte and H. Mercure, A combinatorial column generation algorithm for the maximun stable set problem. *Oper. Res. Lett.* (1997).

[6] J.A.M. Brito, *Um modelo de otimização para dimensionamento de uma rede de telecomunicações*, Tese de mestrado, COPPE. Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil (1999).

[7] J. Bramel and D. Simchi–Levi, On the effectiveness of set covering formulations for the vehicle routing problem with time windows. *Oper. Res.* **45** (1997) 295-301.

[8] S. Butt and D.M. Ryan, An optimal solution procedure for the multiple tour maximum collec tion problem using column generation. *Comp. Oper. Res.* **26** (1999) 427-441.

[9] Z. Chen and W. Powell, A column generation based descomposition algorithm for a parallel machine scheduling problem. *EJOR* **116** (1999) 220-232.

[10] V. Chvátal, *Linear programming.* W.H. Freeman and Company, New York, San Francisco (1983).

[11] Y. Crama and J. Van de Klundert, Approximation algorithms for integer covering problems *via* greedy column generation. *RAIRO: Oper. Res.* **28** (1994) 283-302.

[12] R. Dakin, A tree search algorithm for mixed integer programming problems. *Comput. J.* **8** (1965) 250-255.

[13] G.B. Dantzig, Upper bounds, secondary constraints and block triangulary in linear programming. *Econometrica* **23** (1995) 174-183.

[14] G.B. Dantzig, *Linear programming and extensions.* Princeton University Press, New Jersey, USA (1963).

[15] G.B. Dantzig and P. Wolfe, Decomposition principle for linear programming. *Oper. Res.* **8** (1960) 101-111.

[16] J.M.V. de Carvalho, Exact solution of cutting stock problems using column generation and branch-and-bound. *Int. Trans. Oper. Res.* **5** (1998) 35-43.

[17] J. Delorme, *Contributions à la résolution du problème de recouvrement : méthode de troncature*, Docteur-ingenieur dissertation. Université Paris VI, Paris, France (1974).

[18] G. Desaulniers, J. Desrosiers and M. Solomon, *Accelerating strategies in Column Generation methods for vehicle routing and crew scheduling.* Cahiers du Gerad G-99-36 (1999).

[19] G. Desaulniers, J. Desrosiers, Y. Dumas and M. Solomon, *Daily Aircraft routing and Scheduling.* Cahiers du Gerad G-94-21 (1994).

[20] G. Desaulniers, J. Desrosiers and M. Solomon, *Accelerating strategies in Column Generation Methods for Vehicle Routing and crew scheduling problems.* Cahiers du Gerad G-99-36 (1994).

[21] M. Desrochers, J. Desrosiers and M. Solomon, A new optimization algorithm for the vehicle routing problem with time windows. *Oper. Res.* **40** (1992) 342-353.

[22] J. Desrosiers, Y. Dumas, M.N. Solomon and F. Soumis, *Time constrained routing and scheduling*, edited by M.O. Ball, T.L. Magnanti, C.L. Monma and G.L. Nemhauser, *Network Routing.* INFORMS - North Holland, *Handbooks Oper. Res. Management Sci.* **8** (1995) 35-139.

[23] J. Desrochers and F. Soumis, A column-generation approach to the urban transit crew scheduling problem. *Transportation Sci.* **23** (1989) 1-13.

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

[25] M. EbenChaime, C. Tovey and J.C. Ammons, Circuit partitioning *via* set partitioning and column generation. *Oper. Res.* **44** (1996) 65-76.

[26] M. Gamache, F. Soumis, G. Marquis and J. Desrosiers, A column generation approach for large-scale aircrew rostering problems. *Oper. Res.* **48** (1992) 247-263.

[27] P.C. Gilmore and R.E. Gomory, A linear programming approach to the cutting stock problem. *Oper. Res.* **9** (1961) 849-859.
[28] P.C. Gilmore and R.E. Gomory, A linear programming approach to the cutting stock problem – Part II. *Oper. Res.* **11** (1963) 863-888.
[29] P. Hansen, B. Jaumard and M.V. Poggi de Aragão, Un algorithme primal de programmation linéaire généralisée pour les programmes mixtes. *C. R. Acad. Sci. Paris Sér. I Math.* **313** (1991) 557-560.
[30] B. Jaumard, P. Labit and C. Ribeiro, *A Column Generation Aproach to Cell Formation Problems in Cellular Manufacturing.* Cahiers du Gerad G-99-20 (1999).
[31] B. Jaumard, C. Meyer and T. Vovor, *How to combine a column and row generation method with a column or row elimination procedures – Application to a channel Assiggnment problem.* Cahiers du Gerad G-99-18 (1999).
[32] B. Jaumard, C. Meyer and T. Vovor, *Column/Row Generation and Elimination Methods.* Cahiers du Gerad G-99-34 (1999).
[33] E. Johnson, A. Mehrotal and G.L. Nemhauser, Min-cut clustering. *Math. Programming* **62** (1993) 133-152.
[34] L. Kroon and M. Fischetti, *Crew Scheduling for Netherlands Railways "Destination Customer"* (2001).
[35] L.S. Lasdon, *Optimization Theory for Large Systems.* Macmillan, New York, USA (1970).
[36] A. Lobel, Vehicle scheduling in public transit and Lagrangian pricing. *Management Sci.* **44** (1998) 1637-1649.
[37] O. Marcotte, The cutting stock problem and integer rounding. *Math. Programming* **13** (1985) 82-92.
[38] N. Maculan, M. Fampa and P. Michelon, *Programação linear e inteira.* Notes – COPPE/Universidade Federal do Rio de Janeiro (1999).
[39] N. Maculan, P. Michelon and G. Plateau, Column generation in linear programming with bounding variable constraints and its applications in integer programming. *Pesquisa Operacional* **12** (1992) 45-57.
[40] N. Maculan, M.M. Passini, J.A.M. Brito and A. Lisser, Column generation method for network design, *Transportation and Network Analysis Current Trends*, edited by M. Gendreau and P. Marcotte. Kluwer Academic Publishers (2002) 165-179.
[41] A. Mehrotra and M. Trick, A column generation approach for graph coloring. *INFORMS J. Comput.* **8** (1996) 344-353.
[42] M. Minoux, Optimal traffic assignment in a SS/TDMA frame: A new approach by set covering and column generation. *RAIRO: Oper. Res.* **20** (1986) 273-286.
[43] M. Minoux, A class of combinatorial problems with polynomially solvable large scale set covering/partioning relaxations. *RAIRO: Oper. Res.* **21** (1987) 105-136.
[44] G.L. Nemhauser and S. Park, A polyhedral approach to edge coloring. *Oper. Res. Lett.* **10** (1991) 315-322.
[45] G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization.* John Wiley & Sons Inc. (1988).
[46] M.M. Passini, *Um modelo de otimização combinatória para o dimensionamento de uma rede urbana de telecomunicações*, M.Sc. Thesis. COPPE, Universidade Federal do Rio de Janeiro (1996).
[47] D.M. Ryan and B.A. Foster, An integer programming approach to scheduling, in *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling.* North Holland (1981).
[48] C. Ribeiro, M. Minoux and M.C. Penna, An optimal column-generation-with-ranking algorithm for very large scale partitioning problems in traffic assignment. *Eur. J. Oper. Res.* **41** (1989) 232-239.
[49] C. Ribeiro and F. Soumis, A column generation approach to the multiple-depot vehicle scheduling problem. *Oper. Res.* **42** (1994) 41-52.
[50] M. Savelsbergh, A branch-and-price algorithm for the generalized assignment problem. *Oper. Res.* **46** (1997) 831-841.

[51] L.G. Simonetti, *Geração de colunas para o problema de empacotamento de árvores de Steiner*, M.Sc. dissertation. Universidade Federal do Rio de Janeiro (2003).

[52] A. Sutter, F. Vanderbeck and L. Wolsey, Optimal placemente of add/drop multiplexers: Heuristic and exact algorithms. *Oper. Res.* **46** (1998) 719-728.

[53] E.D. Taillard, A heuristic generation method for the heterogeneous fleet VRP. *RAIRO: Oper. Res.* **33** (1999) 1-14.

[54] F. Vanderbeck, *Decomposition and Column Generation for Integer Programming*, Thèse de doctorat en Sciences Appliquées. Université Catholique de Louvain, Louvain, Belgique (1994).

[55] P.H. Vance, Branch-and-price algorithms for the one-dimensional cutting stock problem. *Comput. Optim. Appl.* **9** (1998) 211-228.

[56] P.H. Vance, C. Barnhart, E.L. Johnson and G.L. Nemhauser, Solving binary cutting stock problems by column generation and branch-and-bound. *Comput. Optim. Appl.* **3** (1994) 111-130.

[57] P.H. Vance, C. Barnhart, E.L. Johnson and G.L. Nemhauser, Airline crew scheduling: A new formulation and decomposition algorithm. *Oper. Res.* **45** (1997) 188-200.

[58] F. Vanderbeck, Computational study of a column generation algorithm for bin packing and cut stocking problems. *Math. Programming* **86** (1999) 565-594.

[59] F. Vanderbeck, On Dantzig–Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Oper. Res.* **48** (2000) 111-128.

[60] J.M. Van den Akker, J.A. Hoogeveen and S.L. van de Velde, Parallel machine scheduling by column generation. *Oper. Res.* **47** (1999) 862-872.

[61] F. Vanderbeck and L. Wolsey, An exact algorithm for IP column generation. *Oper. Res. Lett.* **19** (1996) 151-159.