A Primal Partitioning Approach for Single and Non-Simultaneous Multicommodity Flow Problems*

Malika Hadjiat, Jean-François Maurras, Yann Vaxes †

Laboratoire d'Informatique de Marseille , ESA CNRS 6077 ‡

October 13, 1997, revised March 22, 1998

To appear in European Journal of Operational Research

Abstract

In the first part of this paper, we study the minimum linear cost multicommodity flow problem where the given traffic demand is satisfied through routes having less than a given maximum number of edges. We propose a simplex-based solution method which uses a refined primal partitioning of the basis matrix. Our program has been tested on some real world instances given by the national French telecommunication operator (France Telecom) and also on randomly generated multicommodity flow problems.
The second part of our paper generalizes the techniques described previously to handle additional survivability constraints. Major telecommunication operators want to design multicommodity survivable networks, i.e. which are able to route all traffic demands even if any node or edge is damaged. We present a simple model of this problem based on non-simultaneous multicommodity flows.

Keywords: telecommunication networks, multicommodity flows, non-simultaneous multicommodity flows, simplex method, primal basis partitioning.

1 Introduction

Typically, a telecommunication network is defined by a set of nodes and a set of capacited edges which represent direct physical links. In the first part of this paper, we study the problem of routing at minimum cost given traffic demands between all node pairs, through paths having less than a fixed maximum length. A path length is the number of edges it contains. This length constraint comes from delay considerations which limit the number of nodes through out a demand is routed. This first problem is to compute a minimum linear cost multicommodity flow, which we solve by a simplex method with delayed column generation.
In order to speed up the linear program solving, we refined a primal partitioning of the basis matrix, initially developed by Farvolden, Powell and Lustig [5], to make all the generic simplex

*This work was achieved thanks to a collaboration with the ORE Department directed by Jérôme Chifflet, at the National Center of Telecommunication Studies (C.N.E.T), Sophia-Antipolis, France.

†e-mail: hadjiat, maurras, vaxes@lim.univ-mrs.fr

‡mailing address: LIM, Faculté des Sciences de Luminy, 163 avenue de Luminy, 13288 Marseille cedex 09, France.

calculations done on a small matrix. The size of this squared matrix is at most equal to the number of edges in the network , which is almost half the size of the working matrix proposed by Farvolden et al. [5]. Our program has been tested on real word instances given by France Telecom, and on randomly generated networks. We are able to solve problems having up to 150 nodes, 1500 edges and 11000 demands.

The second part of our paper concerns network design. Clearly, a telecommunication network must be built in such a way that it satisfies traffic demands. Adding to this, it is of major concern to telecommunication operators to design networks at minimum cost. As the cost of every edge depends on the value of its capacity, the design problem reduces to allocate the most suitable capacities among a finite set of possible values, which is a combinatorial optimization problem.

Among all the constraints that must be satisfied when we design telecommunication networks (routing all demands with respect to links' capacities, delay limit ... etc), insuring a suitable degree of survivability towards components failure and cable cuts seems to be the most difficult to implement. Available works done on this topic can be split up into two subsets:

- Some approaches deal with network topology only. The problem solved by these methods is to find a minimum weighted 2-connected graph. Connectivity is considered with respect to edges or nodes depending on whether only link failures are handled or if damages on nodes are also taken into account. Monma and Shallcross [16] proposed heuristics which give approximate solutions for networks with real-world sizes (200 nodes). In 1990, Grotschel, Monma and Stoer [7] described classes of facet defining inequalities for the polytope associated with the 2-connected graphs, and proposed a branch and cut algorithm using these inequalities. This approach allowed them to solve exactly problems with about a hundred nodes. This also gave, by this way, a validation of Monma and Shallcross' heuristic by proving the good quality of their solutions, which are typically less than a few percent from the optimum. More recently, Fortz et al. [6] suggested a similar method which take in consideration additional constraints, so-called *Mesh constraints*. These constraints require that all edges in the network belong to cycles with a bounded length. They are slightly different from our jump constraints [14] since in Mesh constraints, the length of routing paths is bounded, while in our constraints, it is the number of edges used in these paths which is bounded. However, both constraints are derived essentially from same practical considerations set by telecommunication operators.
- All works in the first subset that we just described concentrate on *uncapacited networks*, i.e., where each link can support all the traffic at once. However, for many important and present telecommunication networks, capacities play a fundamental role. In the *capacited network design problem*, one have to find suitable capacities to install on links in order to obtain a failure-resistant network. In this case, a natural model of such survivability problem is the non-simultaneous multicommodity flow one, which is treated for instance in Minoux [15] by generalizing the algorithm of Gomory and Hu to non-simultaneous multicommodity flows. We can also distinguish basis capacities from reserve ones which are used only in a failure situation. It is the case of Lisser, Sarkissian and Vial [11] which use an interior-point method to treat networks of some fifty nodes. Also, we have to refer to the work done by Stoer and Dahl [17], which extends the results of [7] by introducing a discrete set of capacity alternatives among which one have to choose links' capacities.

Our approach belongs to the second family of work, since we study how we can extend the capacities in the network in such a way it survives component failures. More precisely, the purpose of the algorithm presented in this paper is to design a network (or extend the capacities of an existing one) such that each time a component fails, all traffic demands can be re-routed through the network. Local re-routing is not tackled here. As a first approximation, we suppose that the extra-cost of installing additional capacities is linear. We propose here a model that may solve this problem, at least if we suppose that only one damage occurs at a time (on a link or on a node).

This model was introduced by Minoux [15] and involves non-simultaneous multicommodity flows. The algorithm we suggest is based on a generalization of the refined primal partitioning described in the first part of this paper.

2 Linear Cost Multicommodity Flow Problem

Let G be an undirected graph representing a telecommunication network, with a set X of nodes, a set E of edges, upper capacity bounds b_e for each edge e, and a traffic demand table T on node pairs. The problem we have to solve is to determine, for each demand, the routing paths that the traffic follows from the origin point to the destination one, in such a way that on each edge, the total amount of traffic flow is less that the edge's capacity. In this problem, flow variables are continuous. Adding to this, the solution must satisfy the following *jump constraint*:

All demands must be routed on paths having less than δ edges, where δ is an integer constant.

This jump constraint comes from delay issues which limit the number of nodes through out a demand is routed.

The formulation we have chosen is the edge-path one. In other terms, we decompose the flow associated with a commodity into several parts, each one entirely routed on an elementary path from its source to its sink. We have to point out that without jump constraints, the edge-path and node-arc formulations are equivalent. However, while it is easy to add these additional constraints in the edge-path model (through column generation), corresponding formulations with arc variables are difficult to give. In fact, adding jump constraints in the node-arc model may require the characterization of the convex hull of all length constrained paths. Up to now, we have just found a partial description of the 3-cycle polytope [9].

The set of commodities K is defined by the set of all node pairs $(i, j) \in X^2$ that require a positive traffic demand from i to j. Let $P = \bigcup_{i,j \in X} \{P_{ij}\}$, where P_{ij} is the set of all paths with less than δ edges between nodes i and j. If $p \in P_{ij}$ then x_p denotes the flow part of the communication between i and j which is routed on path p. Let c_e be a real positive unitary flow cost associated with each edge e, and $c_p = \sum_{e \in p} c_e$ the unitary cost of path p. We can now present the linear program which models our problem:

$$\text{minimize} \sum_{p \in P} c_p x_p$$

subject to:

$$\sum_{p \in P / e \in p} x_p \leq b_e \ , \quad \forall e \in E \qquad (1)$$

$$\sum_{p \in P_{ij}} x_p = T_{ij} \ , \quad \forall (i,j) \in K \qquad (2)$$

$$x_p \in I\!R^+ \ , \quad \forall p \in P \qquad (3)$$

Constraint 1 expresses that the total amount of traffic flow along each edge must be less than or equal to its capacity. To make the inequalities of constraint 1 equalities, we introduce edge slack variables s_e:

$$\sum_{p \in P / e \in p} x_p + s_e = b_e \ , \ s_e \geq 0 \ , \quad \forall e \in E \qquad (4)$$

The second constraint says that every traffic demand specified in the input is supplied.

Since the number of variables x grows as the number of paths in the graph, it is clear that we cannot apply directly a linear programming algorithm. So, we use a very known approach, which is the column generation simplex based technique.
We initialize our linear program with a small subset of paths, and each time we have solved it to optimality (or near optimality) by the simplex algorithm, we call the column generation routine to add in the linear program more paths with negative reduced costs. This process is repeated until there is no more negative reduced cost paths.

The reduced cost of variables x_p is defined by

$$\bar{c}_p = -w_k + \sum_{e \in p} (c_e - y_e)$$

where k is the commodity routed on path p (i.e., if $p \in P_{ij}$ then $k = (i,j)$) , w_k the dual variable corresponding to commodity k, and y_e the dual variable on edge e. Consequently, the problem of choosing the entering column consists in a cardinality-constrained shortest path problem where edge lengths are equal to the difference between edge costs and edge dual variables. Note that these lengths are fortunately non negative since each edge has a non negative cost and $-y_e$ is exactly the reduced cost of slack variable s_e. We implemented an $O(n^3 \log \delta)$ matrix based algorithm [14] to compute globally shortest paths between all node-pairs, because usual traffic tables are dense. In other words, non-zero demands exist between almost all node pairs. In addition, this algorithm can easily be adapted to handle jump constraints. Of course, dynamic programming may also be used here.

In order to speed up the linear program solving, we also exploited the particular structure of the basis matrix. In fact, the generic simplex calculations can be run much more quickly thanks to a judicious partitioning of the basis matrix.

2.1 Primal Basis Partitioning

Generally, we apply partitioning techniques on linear programs whose constraint matrix presents a particular structure. They consist in partitioning rows and columns sets of the basis matrix, in such a way to highlight particular sub-matrices (usually, identity or bloc-diagonal matrices). The primal and dual linear systems solved at each iteration of the simplex algorithm are then simplified so that essentially only a certain working matrix needs to be inverted or factorized. In the context of multicommodity flow problems, first partitioning studies on the edge-path formulation are due to [12, 5, 1]. Basis partitioning (for edge-path models) consists essentially in using Generalized Upper Bound (GUB) constraints, as introduced initially by Dantzig and Van Slyke [4] and clearly described in [10] or [3].

In [14], we implemented first a primal partitioning of the basis matrix developed in 1992 by Farvolden, Powell and Lustig [5]. We will refer to it as the *initial* primal partitioning. In the present paper, we refine this partitioning technique and we call the new variant, the *refined* primal partitioning.

With respect to the current basic solution, the commodity set K can be partitioned into two subsets of commodities:

- K_m: those split on multiple paths when they are routed in the network,
- K_u: the unsplit ones which are routed on only one path.

In the refined partitioning, the paths associated with basic variables x are divided into three subsets:

- P_u, the set of paths on which unsplit commodities are routed,
- P_r, the set obtained by choosing among the set of paths associated with split commodities, one reference path for each commodity.
- The remaining paths form the subset P_a of additional paths.

In the initial partitioning developed by Farvolden et al. [5], the basic paths are divided in two subsets only: P_u, the set of paths associated with unsplit commodities , and P_m, those routing split commodities.

Now, if we rearrange rows and columns of the basis matrix such that we put split commodities together, unsplit commodities together, and we do the same thing with paths, we then get equivalent matrices B_{ini} and B_{ref}, associated respectively with the initial and refined partitioning. We also isolated the edges having their slack variables in the basis. The other edges (i.e., whose slack variables are out of base) form a subset $\mathcal{E}$ and are called *saturated* edges. Let $\bar{\mathcal{E}} = E \setminus \mathcal{E}$ and $\mathcal{V}$ the set of basic slack variables. Note that this simple partitioning allowed us to highlight a very nice structure in the basis matrix. As shown bellow, B_{ini} is near block-triangular.

$$B_{ini} = \begin{array}{cc} \left[\begin{array}{ccc} I & S^m_{\bar{\mathcal{E}}} & S^u_{\bar{\mathcal{E}}} \\ 0 & S^m_{\mathcal{E}} & S^u_{\mathcal{E}} \\ 0 & \Pi & 0 \\ 0 & 0 & I \end{array}\right] & \begin{array}{c} \bar{\mathcal{E}} \\ \mathcal{E} \\ K_m \\ K_u \end{array} \\ \begin{array}{ccc} \mathcal{V} & P_m & P_u \end{array} & \end{array} \qquad B_{ref} = \begin{array}{cc} \left[\begin{array}{cccc} I & S^a_{\bar{\mathcal{E}}} & S^r_{\bar{\mathcal{E}}} & S^u_{\bar{\mathcal{E}}} \\ 0 & S^a_{\mathcal{E}} & S^r_{\mathcal{E}} & S^u_{\mathcal{E}} \\ 0 & \Pi & I & 0 \\ 0 & 0 & 0 & I \end{array}\right] & \begin{array}{c} \bar{\mathcal{E}} \\ \mathcal{E} \\ K_m \\ K_u \end{array} \\ \begin{array}{cccc} \mathcal{V} & P_a & P_r & P_u \end{array} & \end{array}$$

Notations:

- I: identity matrix (of suitable order);
- $S_F^{\mathcal{P}}$ denotes the edge-path incidence matrix corresponding to some subsets F of edges and $\mathcal{P}$ of paths; (For convenience sake, the P letter is omitted: e.g., S^{P_m} is simply noted S^m)
- Π: matrix of 1s in stairs, i.e. $\Pi = \begin{bmatrix} 1\cdots1 & & & \\ & 1\cdots1 & & \\ & & \ddots & \\ & & & 1\cdots1 \end{bmatrix}$

Let us consider the matrix $W = S^a_{\mathcal{E}} - S^r_{\mathcal{E}}.\Pi$, and referred to as the *working matrix*. The rows of this matrix are the saturated edges, and its columns are associated with the additional paths. Each column of W represents the difference between the characteristic vectors of an additional path and that of the corresponding reference path (i.e., the one associated with the same commodity).
Clearly, since the basis matrix is squared and since the first and last diagonal sub-matrices are identity ones, W is also a nonsingular square matrix. Its size is the same as that of $S^a_{\mathcal{E}}$, which is equal to the number of saturated edges.

Simplex generic calculations can be interpreted as solving right side or left side linear systems on the basis matrix. More precisely, computing the dual variables and the representative column of the chosen entering variable correspond respectively to solving the systems $yB = c$ and $Bv = A^j$, where c is the cost vector, A^j the column in the initial constraint matrix associated with the entering variable x_j, and B the basis matrix. When this later is partitioned, systems solving are simplified since the hole non-unimodular part of the basis matrix is restricted to a very small sub-matrix (which is W). So, the unique basis sub-matrix that must be inverted is W. In fact, vector multiplication by all other unimodular sub-matrices can be realized by simple sums:

$$
yB_{ref} = c \iff \left\{ \begin{array}{llllcl}
y_1 & & & & = & cost(\mathcal{V}) \\
y_1 S^a_{\bar{\mathcal{E}}} + & y_2 S^a_{\mathcal{E}} + & y_3.\Pi & & = & cost(P_a) \\
y_1 S^r_{\bar{\mathcal{E}}} + & y_2 S^r_{\mathcal{E}} + & y_3 & & = & cost(P_r) \\
y_1 S^u_{\bar{\mathcal{E}}} + & y_2 S^u_{\mathcal{E}} + & & y_4 & = & cost(P_u)
\end{array} \right.
$$

$$
\iff \left\{ \begin{array}{lcl}
y_1 & = & cost(\mathcal{V}) \\
y_2 & = & (cost(P_a) - cost(P_r).\Pi - y_1.(S^a_{\bar{\mathcal{E}}} - S^r_{\bar{\mathcal{E}}}.\Pi)).W^{-1} \\
y_3 & = & cost(P_r) - y_1 S^r_{\bar{\mathcal{E}}} - y_2 S^r_{\mathcal{E}} \\
y_4 & = & cost(P_u) - y_1 S^u_{\bar{\mathcal{E}}} - y_2 S^u_{\mathcal{E}}
\end{array} \right.
$$

We actually specialized the computation of the representative column, for each type of entering variable. Three cases may occur: the entering variable may be either a slack variable, a path associated with a split commodity, or a path for an unsplit commodity. The general formulas, which are given below, can be drastically simplified according to each one of these cases:

$$
B_{ref} v = A^j \iff \left\{ \begin{array}{lllllcl}
v_1 + & S^a_{\bar{\mathcal{E}}} v_2 + & S^r_{\bar{\mathcal{E}}} v_3 + & S^u_{\bar{\mathcal{E}}} v_4 & = & A^j_{\bar{\mathcal{E}}} \\
& S^a_{\mathcal{E}} v_2 + & S^r_{\mathcal{E}} v_3 + & S^u_{\mathcal{E}} v_4 & = & A^j_{\mathcal{E}} \\
& \Pi.v_2 + & v_3 + & & = & A^j_{K_m} \\
& & & v_4 & = & A^j_{K_u}
\end{array} \right.
$$

$$
\iff \left\{ \begin{array}{lcl}
v_1 & = & A^j_{\bar{\mathcal{E}}} - (S^a_{\bar{\mathcal{E}}} - S^r_{\bar{\mathcal{E}}}.\Pi).v_2 - S^r_{\bar{\mathcal{E}}} A^j_{K_m} - S^u_{\bar{\mathcal{E}}} A^j_{K_u} \\
v_2 & = & W^{-1}.(A^j_{\mathcal{E}} - S^r_{\mathcal{E}} A^j_{K_m} - S^u_{\mathcal{E}} A^j_{K_u}). \\
v_3 & = & A^j_{K_m} - \Pi.v_2 \\
v_4 & = & A^j_{K_u}
\end{array} \right.
$$

Clearly, our solving method is efficient if sub-matrix W is not large, which is true since the maximum size of W is the number of edges in the network. An important remark is that even if we have in the problem input a lot of commodities, the size of W will never grow more than the total number of edges, which is usually small compared to the number of commodities in telecommunication problems. For instance, in a network with 100 nodes there is about 700 edges, and about 4000 commodities.

The first simplex version we implemented is the one based on keeping in memory explicitly the inverse basis matrix. Actually, we keep in memory only the inverse of W and the structure of various basic paths. In order to speed up our program, we are currently implementing a sparse LU decomposition of the working matrix based on the works done by [13, 2].

When we apply the initial primal partitioning of Farvolden et al., the working sub-matrix is

$$
W_{ini} = \left[\begin{array}{c} S^m_{\mathcal{E}} \\ \Pi \end{array} \right]
$$

and its size of is equal to the number of paths corresponding to split commodities. In order to assess theoretically the efficiency of the refined partitioning relatively to the initial one, let us

compare the overall complexity of a simplex iteration in both cases. A simplex iteration consists in computing the dual variables, then the representative column of the chosen entering variable, and finally updating the inverse of the working matrix. This is done in $O(\delta \times |K| + \delta \times |P_m| + |P_m|^2)$ if we use a primal partitioning of the basis matrix, and it is performed in $O(\delta \times |K| + \delta \times |\mathcal{E}| + |\mathcal{E}|^2)$ with the refined version. But, we can easily prove that on one hand, $|P_m| = |\mathcal{E}| + |K_m|$ and on another hand, $|P_m| \leq 2 \times |\mathcal{E}|$. Hence, even if there is no change in the worst case complexity bound between both partitioning versions, the gain is significant because of the following remark:
From our experience we noted that split commodities are generally routed on 2 paths and rarely on 3 or more paths. We can then say that, when we use the refined partitioning instead of the initial version, we divide by almost two the size of the working matrix. Consequently, in practice, the time for computing the representative column of the chosen entering variable is divided by almost 2, and the time for updating the working inverse matrix is divided by 4.

3 Linear Cost Non-simultaneous Multicommodity Flow Problem

A survivable network is such that all demands can be routed even if a damage occurs on an arc or on a node of the network. When we design such a network, the problem we have to solve consists in choosing the most suitable capacities to allocate to the links, as cheaply as possible. We will suppose here that a capacited network is given and the problem concerns extending capacities. The linear relaxation of this problem, were the added capacities are real values, is treated in this paper. In other words, we are interested in a network design problem with continuous design variables. However, as stated in the conclusions, the solution method we propose here for this relaxation is of interest in a branch and bound algorithm for the corresponding problem with integral design variables.

We must point out that we are only interested in global re-routing , i.e. for example if a component fails we do not look for a possibility to change locally (around the broken component) the routing to make it feasible. Consequently, the network designed by the algorithm we developed insure that if any component (but only one at a time) fails, the network remain feasible, and that this network property has been achieved at minimum cost.

Hence, the linear optimization problem treated in this section is obtained by juxtaposing several multicommodity flow problems each one is defined on a different sub-network derived from the initial one by deleting the damaged component (edge or node). This lead us to define a set of network states L which include every state associated with one damage. The total number of states is then equal to $|E| + |X|$. This model, called *non-simultaneous multicommodity flow problem*, was introduced by Minoux [15]. We will suppose that a feasible capacited network is given initially, i.e. for each demand there exists a routing. We also assume that these routings are already computed and given as input. For each network state, we restrict the commodities to those having at least one route which passes through the damaged component (arc or node), i.e. the commodities that must be re-routed (denoted by K^l). We also subtract from each edge capacity the total amount of flow routed on it and which is associated with satisfied commodities (i.e., those that are not routed through the damage component). We obtain consequently residual capacities, denoted by β_e.

The linear program which corresponds to our non-simultaneous multicommodity flow problem is defined as follows. The only variables that link the multicommodity flow problems associated with each network state (independent otherwise), are investment variables z whose total cost $c'z$ have to be minimized. New costs c' are then associated with arcs, since routing costs and investment costs are different. We use again the same notations by indexing them on each network state $l \in L$, except the investment variables z which are common. Let $P^l = \bigcup_{i,j \in X} \{P^l_{ij}\}$ denote the set of all paths on which the demands can be routed through the network in state l.

$$\text{minimize} \sum_{l \in L} \sum_{p \in P^l} c_p x^l_p + \sum_{e \in E} c'_e z_e$$

subject to:

$$\sum_{p \in P^l / e \in p} x^l_p + s^l_e - z_e = \beta^l_e \ , \quad \forall e \in E \ , \quad \forall l \in L \quad (5)$$

$$\sum_{p \in P^l_{ij}} x^l_p = T_{ij} \ , \quad \forall (i,j) \in K^l \ , \quad \forall l \in L \qquad (6)$$

$$z_e \geq 0, x^l_p \geq 0 \ , \quad \forall e \in E, \quad \forall p \in P^l \ , \quad \forall l \in L \quad (7)$$

As previously, we will solve this problem using an adapted simplex algorithm where we exploit the particular structure of the basis matrix. For this, we generalized the refined primal partitioning technique to non-simultaneous multicommodity flows.

Let us perform the same partitioning as the one described in previous section for each network state. So, we divide, for each state $l \in L$, the set of commodities K^l into two subsets: a set of split commodities (K^l_m) and a set of unsplit ones K^l_u. In the same way, we have three categories of basic paths: P^l_u (paths of unsplit commodities), P^l_r (reference paths of split commodities) and P^l_a (additional paths of split commodities).

To sum up, we have $K^l = K^l_m \cup K^l_u$ and $P^l_B = P^l_u \cup P^l_r \cup P^l_a$, where P^l_B denote the set of basic paths associated with state l. It is clear that the partitioning of our global problem can be obtained by simply joining the partitioning of all network states. (It must be pointed out that same commodities associated with different states are considered to be different, likewise for paths and edges). In other words, we have $K = K_m \cup K_u$ and $P_B = P_u \cup P_r \cup P_a$, where:

$$K_m = \bigcup_{l \in L} K^l_m \ , \ K_u = \bigcup_{l \in L} K^l_u \ , \ P_u = \bigcup_{l \in L} P^l_u \ , \ P_r = \bigcup_{l \in L} P^l_r \ , \ P_a = \bigcup_{l \in L} P^l_a$$

We distinguish again, for each state l, among constraints of type (5) those corresponding to saturated edges (denoted by $\mathcal{E}^l$), and let $\mathcal{E} = \bigcup_{l \in L} \mathcal{E}^l$. Recall that, for each state l, a saturated edge e is such that the slack variable s^l_e is out-of-base. Note that the investment variable z_e may be basic or not. At last, let $\mathcal{V}_s$ denote the set of basic slack variables and $\mathcal{V}_z$ the set of basic investment variables. Consequently, the basis matrix B looks as follows ($\tilde{I}$ denotes a $(0,1)$-matrix):

$$
\begin{array}{cc}
B = \left[\begin{array}{ccccc}
I & -\tilde{I}_{\bar{\mathcal{E}}} & S^a_{\bar{\mathcal{E}}} & S^r_{\bar{\mathcal{E}}} & S^u_{\bar{\mathcal{E}}} \\
0 & -\tilde{I}_{\mathcal{E}} & S^a_{\mathcal{E}} & S^r_{\mathcal{E}} & S^u_{\mathcal{E}} \\
0 & 0 & \Pi & I & 0 \\
0 & 0 & 0 & 0 & I
\end{array}\right] & \begin{array}{c} \bar{\mathcal{E}} \\ \mathcal{E} \\ K_m \\ K_u \end{array} \\
\begin{array}{ccccc} \mathcal{V}_s & \mathcal{V}_z & P_a & P_r & P_u \end{array} &
\end{array}
$$

The working matrix is defined by $W = \left[\begin{array}{c|c} -\tilde{I}_{\mathcal{E}} & S^a_{\mathcal{E}} - S^r_{\mathcal{E}}.\Pi \end{array}\right]$. Clearly, matrix W is squared and regular. Adding to this, we can remark that it inherits from the global basis matrix its block structure. That is, the rows relative to different network states are independent (i.e., there are no common variables).

In what follows, for any set of columns or rows Y we will denote by $Y^l \subset Y$ the subset containing the elements of Y associated with network state l.

When we put together paths and edges associated with a same state, we then get an equivalent matrix of W:

$$
W = \left[\begin{array}{c|cccc}
-\tilde{I}_{\mathcal{E}^1} & S^{a1}_{\mathcal{E}^1} - S^{r1}_{\mathcal{E}^1}.\Pi^1 & & & \\
-\tilde{I}_{\mathcal{E}^2} & & S^{a2}_{\mathcal{E}^2} - S^{r2}_{\mathcal{E}^2}.\Pi^2 & & \\
\vdots & & & \ddots & \\
-\tilde{I}_{\mathcal{E}^{|L|}} & & & & S^{a\,|L|}_{\mathcal{E}^{|L|}} - S^{r\,|L|}_{\mathcal{E}^{|L|}}.\Pi^{|L|}
\end{array}\right]
$$

Since matrix W is nonsingular, it is clear that the columns of each submatrix $(S^{al}_{\mathcal{E}^l} - \Pi^l.S^{rl}_{\mathcal{E}^l})$ are linearly independent. Hence, we can extract from each block, a subset of rows $\mathcal{E}^l_1$ from $\mathcal{E}^l$ such that the corresponding blocks $S^{al}_{\mathcal{E}^l_1} - S^{rl}_{\mathcal{E}^l_1}.\Pi^l$ are squared and regular. The set of remaining rows is denoted by $\mathcal{E}^l_2 = \mathcal{E}^l \setminus \mathcal{E}^l_1$. As before, we set $\mathcal{E}_1 = \bigcup_{l\in L} \mathcal{E}^l_1$ and $\mathcal{E}_2 = \bigcup_{l\in L} \mathcal{E}^l_2$. For convenience sake, we use the notation Δ^l to refer to $S^{al} - S^{rl}.\Pi^l$.

We rewrite the matrix W with these new notations :

$$
W = \left[\begin{array}{c|cccc}
-\tilde{I}_{\mathcal{E}_2} & \Delta^1_{\mathcal{E}_2} & \Delta^2_{\mathcal{E}_2} & \dots & \Delta^{|L|}_{\mathcal{E}_2} \\
-\tilde{I}_{\mathcal{E}^1_1} & \Delta^1_{\mathcal{E}^1_1} & & & \\
-\tilde{I}_{\mathcal{E}^2_1} & & \Delta^2_{\mathcal{E}^2_1} & & \\
\vdots & & & \ddots & \\
-\tilde{I}_{\mathcal{E}^{|L|}_1} & & & & \Delta^{|L|}_{\mathcal{E}^{|L|}_1}
\end{array}\right]
$$

When we apply again block inversion rules, we deduce that matrix $W' = -\tilde{I_{\mathcal{E}_2}} - \Delta_{\mathcal{E}_2}.(\Delta_{\mathcal{E}_1})^{-1}.(-\tilde{I_{\mathcal{E}_1}})$ is regular. Consequently, we will take this new matrix W' (which is smaller than W) as our working matrix. So, while running the simplex algorithm, we just have to keep in memory and to update after each iteration the inverse matrices of some $\Delta^l_{\mathcal{E}^l_1}$ blocks, and to maintain subsets $\mathcal{E}^l_1$ and $\mathcal{E}^l_2$.

4 Experimental Results

In this section, we present computational results of both our single and non-simultaneous multicommodity flow algorithms. Tests have been performed on some real-word instances supplied by the national French telecommunication operator (France Telecom) and also on randomly generated multicommodity flow problems. Table 1 gives a description of these instances. All runs were done on a SUN SPARC 5 with 64 MB RAM.

Concerning the single multicommodity flow problem, table 2 compares on random instances the running times (in seconds) between both levels of primal partitioning techniques. $Time_{ini}$ refers to the initial primal partitioning and $Time_{ref}$ to the refined version. We also report in percent the saving of running time between them. One may remark that for some instances we reduced the running time by almost half time, which is a good performance. The average saving is more than 25 percent.

On tables 3 and 4, we report for each problem instance some characteristics of the optimal solution found using the refined primal partitioning approach, i.e.:

- the number of split commodities ($|K_m|$),
- the number of saturated edges ($|\mathcal{E}|$),
- the ratio of the number of pivots performed on the working matrix over the total number of pivots,
- and finally the running time expressed in seconds.

Adding to this, we give explicitly for random instances (table 4) the times for simplex calculations ($Tsimplex$), column generation ($Tgencol$), and the total running time ($Total$).

About the jump constraints, we have chosen to define the parameter δ by the minimum number of edges used to route a commodity, to which we add a difference λ. A different bound δ_k is then fixed for each commodity k, i.e.,: (Let i and j be respectively the source and the sink of commodity k)

$$\delta_k = \lambda + \min_{p \in P_{ij}} \{|p|\} \text{ , for each } k \in K$$

If we analyse an optimal solution found without the additional jump constraints, we usually see that most routing paths are short (no more that 7 edges). So, jump constraints have an impact on the solution quality if $0 \leq \lambda \leq 5$. In all experiments reported in this paper, we set λ to a constant equal to 4. We used the same values of δ_k and λ in solving the non-simultaneous multicommodity flow problems. Nevertheless, it may be more reasonable to increase λ in the sub-problems associated with an arc or vertex damage.
Jump constraints have small influence on pricing (column generation) time, and consequently on the total running time. This is due to the fact that cardinality constrained shortest path algorithms are very efficient.

We also performed some comparative tests with CPLEX 4.0 as a standard commercial simplex code. CPLEX is used inside the same column generation scheme, and it is configured in such a way to induce an equivalent number of iterations (e.g., we put off the steepest edge criteria). The gain is approximately 50 % on random instances. On France Telecom instances, which are easier, column generation time increases relatively to the total time. So, it is difficult to decide between both solvers.

One may note that the number of split commodities is always close to the number of saturated edges. Since it can be easily proved that the number of saturated edges is equal to the number

France Telecom instances				Random instances			
Problem	Nodes	Edges	$\|K\|$	Problem	Nodes	Edges	$\|K\|$
T.01	53	310	1377	R.10	12	25	66
T.02	63	328	1950	R.15	16	47	120
T.03	63	324	1950	R.25	25	139	300
T.04	63	340	1950	R.30	32	132	496
T.05	74	369	2698	R.45	45	295	990
T.06	74	369	2698	R.50	49	339	1176
T.07	74	365	2696	R.60	59	454	1711
T.08	79	381	3075	R.70	68	586	2278
T.09	79	387	3074	R.80	80	788	3160
T.10	84	393	3480	R.90	84	500	3486
T.11	84	407	3481	R.100	99	952	4851
T.12	84	400	3480				
T.13	90	421	3997				
T.14	90	420	3998				
T.15	106	452	5556				
T.16	106	752	3728				

Table 1: Description of France Telecom and random instances

Problem	Time_{ini}	Time_{ref}	Gain	Time_{CPLEX}
R.25	1.18	0.78	52 %	1.69
R.30	0.84	0.74	14 %	2.31
R.45	6.40	4.78	34 %	9.07
R.50	9.95	7.01	42 %	13.61
R.60	18.85	13.28	42 %	30.03
R.70	78.92	52.97	49 %	68.31
R.80	24.16	19.97	21 %	44.25
R.90	19.51	16.40	19 %	47.40
R.100	88.57	72.60	22 %	149.67

Table 2: Time comparison between initial/refined partitioning and CPLEX (random instances)

of additional routing paths, this means that split commodities are rarely routed on more than two paths. Also, we remark that random instances are more difficult to solve than real-word telecommunication ones. This can be explained by the fact that our random graphs are more dense (generally, $|E| = 8 \times |X|$ instead of less than $4 \times |X|$). Consequently, there are more alternative paths to route traffic demands.

Concerning non-simultaneous multicommodity flows, we are able at this time to solve problems having up to 20 nodes and 50 edges, where states are associated with some or all edge failures. Such graphs imply linear programs having about 3000 constraints. Keeping this in mind, the times shown on tables 5 and 6 are reasonable.

5 Conclusions

In this paper, we presented an algorithm to solve a linear cost multicommodity flow problem. This is a simplex-based algorithm which we make faster using a refined primal partitioning of

Problem	$\|K_m\|$	$\|\mathcal{E}\|$	nb. pivots	Tsimplex	Tgencol	Total
R.25	71	78	1694/1853	0.62	0.16	0.78
R.30	43	47	1068/1261	0.48	0.26	0.74
R.50	104	108	5736/6263	6.00	1.01	7.01
R.60	140	146	7285/7959	11.43	1.85	13.28
R.70	169	177	15386/16305	50.02	2.95	52.97
R.80	144	151	7253/8504	16.49	3.48	19.97
R.90	97	98	5265/6500	11.81	4.59	16.40
R.100	170	175	16107/17941	64.27	8.33	72.60

Table 3: Experiments with refined partitioning on random instances

Problem	$\|K_m\|$	$\|\mathcal{E}\|$	nb. pivots	Time_{ref}	Time_{CPLEX}
T.01	23	28	426/1302	2.83	5.55
T.02	15	16	398/1172	4.64	5.11
T.03	10	10	101/670	2.76	3.75
T.04	30	38	554/1689	4.91	5.34
T.05	33	36	876/2334	7.01	12.34
T.06	32	35	470/2072	5.48	12.30
T.07	14	14	135/924	4.20	6.87
T.08	19	20	256/1391	6.56	8.36
T.09	37	37	590/1476	7.97	8.31
T.10	60	63	1428/3267	13.15	25.48
T.11	20	26	296/2295	8.99	11.17
T.12	25	27	678/2135	12.55	17.73
T.13	15	16	370/1641	12.08	13.91
T.14	44	44	584/2160	10.35	17.17
T.15	5	5	60/704	11.65	17.27
T.16	38	39	135/1223	17.77	19.64

Table 4: Experiments with refined partitioning on France Telecom instances

Problem	States	$\|\mathcal{E}\|/(\|E\| \times \|L\|)$	nb. pivots	Time_{ref}
R.10	5	97/125	1050	1.4
	10	140/250	1645	3.6
	15	192/375	3330	10.4
	25	277/625	4950	28.1
R.15	5	224/235	5891	28.2
	10	442/470	18556	161.9
	25	945/1175	44670	1028.1
	47	1648/2209	102170	5224.4

Table 5: Non-simultaneous multicommodity flows on random networks

Problem	Nodes	Edges	$\|K\|$	States	$\|\mathcal{E}\|/(\|E\| \times \|L\|)$	nb. pivots	Time$_{ref}$
T.a	9	23	36	5	82/115	272	0.25
				10	129/230	1343	2.38
				15	163/345	1473	4.41
				23	200/529	2425	9.85
T.b	10	32	45	5	122/160	908	1.69
				10	197/320	3341	14.68
				15	306/480	5567	30.27
				32	543/1024	17880	281.15
T.c	13	40	78	5	163/200	533	2.09
				10	292/400	7503	44.83
				15	441/600	7088	59.52
				40	740/1600	29115	591.54
T.d	15	34	105	5	112/170	600	1.31
				10	170/340	1492	5.13
				15	249/510	7441	46.30
				79	391/1156	6034	85.04
T.e	15	58	105	5	249/290	3682	22.75
				10	411/580	15098	172.67
				15	622/870	27230	445.95
				58	1707/3364	19319	978.52

Table 6: Non-simultaneous multicommodity flows on France Telecom networks

the basis matrix. Next, we generalized this approach to solve non-simultaneous multicommodity flow problems with linear objective cost functions. These algorithms solve problem instances of practical interest in satisfactory times.

Both problems discussed here are linear relaxations of capacited network design ones, respectively with and without survivability constraints. We implemented the linear cost multicommodity flow algorithm inside a branch and bound methodology to design networks by choosing edge-capacities among a finite set of discrete available values [8]. Maybe the second algorithm described in this paper will also allow us (with branch and bound) to design networks that satisfy additional constraints, such as: surviving any edge or vertex failure, or satisfying different traffic tables associated with non simultaneous time periods.

Acknowledgments

The authors wish to thank the referees for their helpful comments.

References

[1] Barnhart, C., Johnson, E.L., Nemhauser, G.L., Sigismondi, G., and Vance, P., "Formulating a mixed integer programming problem to improve solvability", *Operations Research* 41/6 (1993) 1013–1019.

[2] Barr, R.S., Farhangian, K., and Kennington, J.L., "Networks with side constraints: an LU factorization update", *Ann. Soc. Logist. Engin.* 1 (1986) 66-85.

[3] Chvátal, V., *Linear Programming*, W.H. Freeman and Company, New York/San Francisco, 1983.

[4] Dantzig, G.B., and Van Slyke, R.M., "Generalized upper bounding techniques", *Journal of Compter System Science* 1 (1967).

[5] Farvolden, J.M., Powell, W.B., and Lustig, I.J., "A primal partitioning solution for multicommodity network flow problems", *Operations Research* 41/4 (1993) 669–693.

[6] Fortz, B., Labbe, M., and Maffioli, F., "Two-Connected Network with Bounded Meshes", Technical Report IS-MG 96/8, Université Libre de Bruxelles, August 1996.

[7] Grötschel, M., Monma, C., and Stoer, M., "Polyhedral Approaches to Network Survivability", Technical Report 189, Universität Augsburg, Inst. f. Mathematik, August 1990.

[8] Hadjiat, M., Maurras, J.F., and Vaxes, Y., "Solving Telecommunication Network Design by Branch and Bound", *Tenth Conference of the European Chapter on Combinatorial Optimization (ECCO X)*, Tenerife, 1997.

[9] Kovalev, M., Maurras, J.F., and Vaxes, Y., "About the convexe hull of the cycles of length three in complete graphs", *Tenth Conference of the European Chapter on Combinatorial Optimization (ECCO X)*, Tenerife, 1997.

[10] Lasdon, L., *Optimization Theory for Large Systems*, MacMillan Co., New York, 1970.

[11] Lisser, A., Sarkissian, R., and Vial, J.P., "Optimal joint synthesis of base and reserve telecommunication networks", Technical Report, Department of management studies, University of Geneva, Switzerland, 1995.

[12] MacCallum, C.J., "A generalized upper bounding approach to a communication network planning problem", *Networks* 7/1 (1977) 1–23.

[13] Markovitz, H.M., "The elimination form of inverse and its applications to linear programming", *Management Science* 3 (1957) 255–269.

[14] Maurras, J.F., and Vaxes, Y., "Multicommodity Network Flow with Jump Constraints", *Discrete Mathematics* 165/166 (1997) 481–486.

[15] Minoux, M., "Optimum synthesis of a network with non-simultaneous flow requirements", in: P. Hansen (ed.), *Studies on Graphs and Discrete Programming*, North-Holland Publishers, 1981, 313–360.

[16] Monma, C., and Shallcross, D. F., "Methods for Designing Communications Networks with Certain Two-Connected Survivability Constraints", *Operations Research* 37/4 (1989) 531–541.

[17] Stoer, M., and Dahl, G., "A polyhedral approach to multicommodity survivable network design", *Numerische Mathematik* 68 (1994) 149–167.