Large scale LP relaxations for minimum cost multicommodity flow problems with step increasing cost functions and computational results

V. Gabrel M. Minoux
Laboratoire MASI, Université Paris 6
4 place Jussieu, 75252 Paris Cedex 05, France
e_mails: {gabrel,minoux}@masi.ibp.fr

Abstract

We address here a class of particularly hard-to-solve minimum cost multicommodity network flow problems when the link cost functions are discontinuous step increasing. Several LP relaxations for such problems are proposed, giving rise to (very) large scale linear programs. It is shown that the continuous optimal solutions to these linear programs can nevertheless be obtained via combined column generation and constraint generation procedures. Preliminary computational results are presented showing that our relaxations lead to lower bounds typically within 10% to 20% off the (unknown) optimal values.

As far as we know, this is the first time a systematic way of generating nontrivial lower bounds to minimum cost multicommodity network flow problems with step increasing (discontinuous) cost functions is proposed.

Key words: Telecommunication network optimization, minimum cost multicommodity network flow, large scale optimization

Résumé

On s'intéresse ici à une classe particulièrement difficile de problèmes de multiflots à coût minimum : celle où les fonctions de coût sur les liens du réseau sont discontinues et croissantes en escalier. Plusieurs relaxations sous forme de programmes linéaires sont proposées, qui conduisent à des problèmes de (très) grande taille. On montre que les solutions optimales continues de ces programmes linéaires peuvent néanmoins être obtenues par des techniques de programmation linéaire généralisée, combinant génération de colonnes et génération de contraintes. On présente des résultats de calcul préliminaires qui montrent que les relaxations proposées permettent d'obtenir des bornes inférieures s'écartant typiquement de 10% à 20% des solutions optimales (inconnues).

A notre connaissance, il s'agit là de la première approche systématique pour générer des bornes inférieures non triviales pour des problèmes de multiflots à fonctions de coût croissantes en escalier.

Mots-clés: Optimisation de réseaux de télécommunications, multiflot à cout minimum, optimisation de problèmes de grande taille

June 1996

Large scale LP relaxations for minimum cost multicommodity flow problems with step increasing cost functions and computational results

V. Gabrel* M. Minoux*

Abstract

We address here a class of particularly hard-to-solve minimum cost multicommodity network flow problems when the link cost functions are discontinuous step increasing. Several LP relaxations for such problems are proposed, giving rise to (very) large scale linear programs. It is shown that the continuous optimal solutions to these linear programs can nevertheless be obtained via combined column generation and constraint generation procedures. Preliminary computational results are presented showing that our relaxations lead to lower bounds typically within 10% to 20% off the (unknown) optimal values.

As far as we know, this is the first time a systematic way of generating nontrivial lower bounds to minimum cost multicommodity network flow problems with step increasing (discontinuous) cost functions is proposed.

1. Introduction

Minimum cost multicommodity flow problems arise as basic models in the context of many applications such as: telecommunication networks, transportation networks and traffic analysis, logistics etc.

Depending on the structure and/or mathematical properties of the cost functions on the links, obtaining exact optimal solutions or good approximate solutions (with guaranteed quality) may lead to various degrees of difficulty. One of the easiest known special cases is when all cost functions are linear (with nonnegative cost per unit flow) since this reduces to shortest path computations. The case of separable convex cost functions (whether differentiable or nonsmooth) may also be considered as a well-solved class, by means of linearization and decomposition techniques (see Fratta et al. 1973, Ouorou 1995).

The case of concave (differentiable) cost functions and the linear with fixed costs case have also been studied by various authors (for a survey see Minoux 1989). However no practically efficient exact algorithms are known for such problems, at least for solving practical size instances, only good approximate solution algorithms (without a priori quality guarantee) are available (see Minoux 1989, Ahuja et al. 1993). One of the reasons for this situation is the lack of lower bounds of reasonable quality to direct tree search in branch and bound approaches (even in the linear with fixed cost case, known lower bounds may be as poor as 40% to 60% off the exact optimum value).

We address here an even more complex class of minimum cost multicommodity flow problems: the case when the link cost functions are step increasing **discontinuous**. We propose various large scale LP relaxations for this problem. Exact optimal continuous solutions to the corresponding (large scale) linear programs are obtained via a novel approach combining both column generation **and** constraint generation. The resulting optimal solutions provide lower bounds to the (unknown) exact solutions for the minimum

*MASI, Université Paris 6, 4 place Jussieu, 75252 Paris Cedex 05, France, e_mails: {gabrel,minoux}@masi.ibp.fr

cost multicommodity flow problem to be solved. Preliminary computational results are presented on a series of test problems of limited size, showing that our relaxations can produce rather tight lower bounds, within 10% to 20% off the (unknown) exact optimal solutions.

As far as we know, this is the first time a systematic way of generating non trivial lower bounds to minimum cost multicommodity network flow problems with step increasing (discontinuous) cost functions is proposed.

2. Problem description

The network structure is given as a non-directed graph $G=[X,U]$ where X is the set of nodes and U the set of (non-directed) edges. We denote $|X| = m$ and $|U| = n$.

The problem to be considered is to decide the amount of capacity $x_u \geq 0$ to install on each edge u of the network in order to

- satisfy a given set of multicommodity flow requirements: there are $K \in \mathbb{N}$ source-sink pairs, and for each $k \in [1,K]$ a given requested flow value d_k has to be routed between the source node $s(k)$ and the sink node $t(k)$,
- satisfy upper bound constraints:
  $\forall u \in U: 0 \leq x_u \leq \beta_u$
- minimize the total cost of the network which, in terms of given individual link cost functions $\Phi_u(x_u)\,(u = 1,\ldots,n)$ may be written as:
  $z = \sum\limits_{u \in U} \Phi_u(x_u)$

Minimum cost multicommodity flow problems have been extensively studied in the special cases where the cost functions $\Phi_u(x_u)$ are linear (see Kennington 1978), linear with fixed cost or nonlinear concave but continuous or differentiable (see e.g. Minoux 1989).

We address here the minimum cost multicommodity flow problem in the case of discontinuous step-increasing cost functions. To the best of our knowledge, no systematic study has been carried out before to build relaxations and derive good lower bounds for such problems.

For each edge $u \in U$ in the network, the cost function $\Phi_u(x_u)$ may be defined as follows.

Let $V_u = \left\{v_u^0, v_u^1, \ldots, v_u^{q(u)}\right\}$ be a finite set of values representing the discontinuity points of the Φ_u function and denote

$$\gamma_u^0 = \Phi_u(v_u^0)$$
$$\gamma_u^1 = \Phi_u(v_u^1)$$
$$\gamma_u^2 = \Phi_u(v_u^2)$$
$$\ldots$$
$$\gamma_u^{q(u)} = \Phi_u(v_u^{q(u)})$$
$$0 = v_u^0 \leq v_u^1 \leq v_u^2 \leq \ldots \leq v_u^{q(u)}$$
$$0 = \gamma_u^0 < \gamma_u^1 < \gamma_u^2 < \ldots < \gamma_u^{q(u)}$$

With this notation we have:

$\Phi_u(x_u) = 0$ if $x_u = 0$ and, $\forall i = 1,\ldots,q(u)$: $\Phi_u(x_u) = \gamma_u^i$ for all $x_u \in \left]v_u^{i-1}, v_u^i\right]$.

Figure 1 below shows a typical cost function of this type. Note here that the cost function $\Phi_u(x_u)$ is not defined for values of x_u greater than $\beta_u = v_u^{q(u)}$, therefore our model will include bound constraints of the form: $0 \le x_u \le \beta_u$ either explicitly or implicitly.

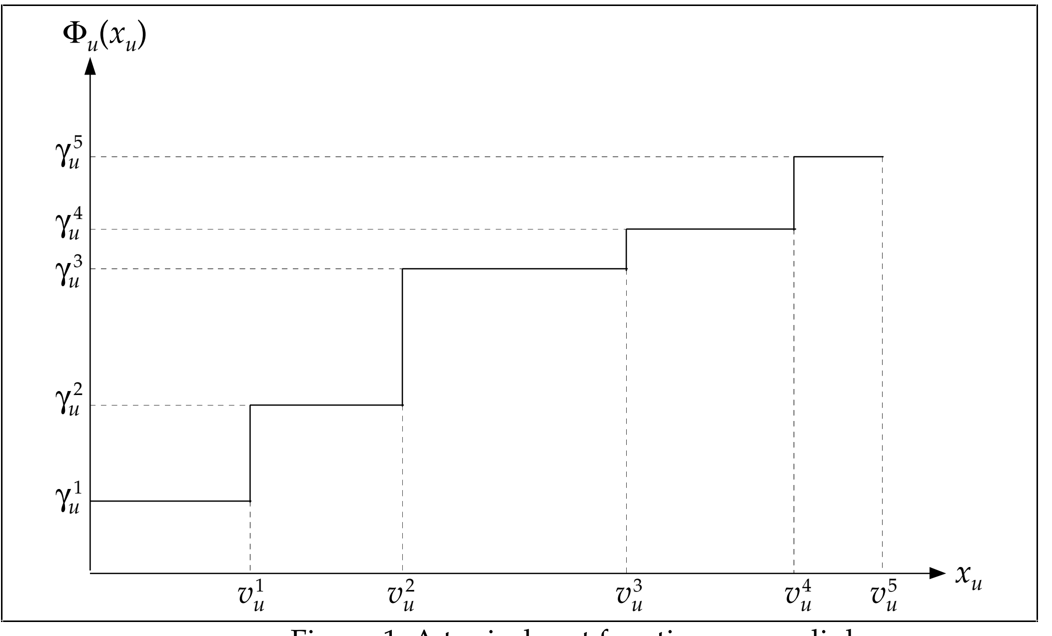

Figure 1: A typical cost function on one link

For a given set of multicommodity flow requirements defined by a list of source-sink pairs $s(k)$, $t(k)$ ($k = 1,\ldots,k$), and a list of requirements d_k (amount of the k^{th} flow to be routed between $s(k)$ and $t(k)$), we denote by $X \subset \mathbb{R}_+^n$ the polyhedron representing the set of all feasible multicommodity flows. Thus $x = (x_u)_{u\in U}$ belongs to X if and only if a feasible multicommodity flow exists when on each edge $u \in U$, the total capacity installed is x_u (of course $x_u \ge 0$ on each edge u).

Several linear representations of X (as a system of linear equality and inequality constraints involving the x variables and possibly other variables) are known, including the so-called arc-node formulation and arc-path formulation (for an overview, see e.g. Minoux 1989).

Later in the paper we will use the following representation involving the x variables only. For any $\lambda = (\lambda_1,\ldots,\lambda_n) \in \mathbb{R}_+^n$, let $\theta(\lambda)$ denote the quantity:

$$\theta(\lambda) = \sum_{k=1}^{K} d_k \times l_k^*(\lambda)$$

where $l_k^*(\lambda)$ is the length of the shortest chain joining $s(k)$ and $t(k)$ in G when each edge $u \in U$ is given length $\lambda_u \ge 0$ (note that $\theta(\lambda)$ may be interpreted as the minimum cost multicommodity flow solution when, on each edge u, the cost function $\Phi_u(x_u)$ is linear of the form $\lambda_u x_u$). Then $x = (x_u)_{u\in U}$ belongs to X if and only if for all $\lambda \in \mathbb{R}_+^n$, we have

$$\sum_{u\in U} \lambda_u x_u \ge \theta(\lambda) \tag{1}$$

(see e.g. Gondran & Minoux 1995, chapter 6).

One might object that this gives rise to infinitely many inequalities of the form (1). However it is well known that it is enought to restrict to finitely many λ vectors, namely those corresponding to the extreme points of the potype dual to the feasible multicommodity flow problem. Moreover, testing whether a given $\overline{x} \in \mathbb{R}^n$ belongs to X can be done in polynomial time since it reduces to finding the minimum solution value to the convex (nondifferentiable) problem:

$$\begin{cases} \text{Minimize}\, \theta(\lambda) - \sum\limits_{u \in U} \lambda_u \overline{x}_u \\ \text{s.c.} \quad \sum\limits_{u \in U} \lambda_u = 1 \\ \qquad \lambda_u \geq 0 \quad \forall u \in U \end{cases}$$

In view of the above, a possible formulation of the minimum cost multicommodity flow problem we want to investigate is:

$$(P_0)\begin{cases} \text{Minimize} \sum\limits_{u \in U} \Phi_u(x_u) \\ \text{s.c.} \quad x \in X \\ \qquad 0 \leq x_u \leq \beta_u \quad \forall u \in U \end{cases}$$

We first show that this problem is equivalent to:

$$(P_1)\begin{cases} \text{Minimize} \sum\limits_{u \in U} \Phi_u(x_u) \\ \text{s.c.} \quad x \in X \\ \qquad x_u \in V_u \quad (\forall u \in U) \end{cases}$$

Since, $\forall u \in U: V_u \subset [0, \beta_u]$ (remember that $\beta_u = v_u^{q(u)}$), (P_0) is a relaxation of (P_1). So it will be enough to prove that any solution $(x_u^0)_{u \in U}$ to (P_0) may be converted into a solution to (P_1) having the same solution value.

For any $u \in U$ there exists $i \in [1, q(u)]$ such that $v_u^{i-1} \leq x_u^0 \leq v_u^i$. Also $\Phi_u(v_u^i) = \Phi_u(x_u^0)$. By choosing $x_u^1 = v_u^i$ we can see that $x^1 \geq x^0$ hence $x^1 \in X$ and $\Phi(x^1) = \Phi(x^0)$.

In the rest of the paper we will consider problem (P_1).

3. A large scale mixed integer linear programming reformulation

Consider any node $i \in X$ and r_i the total amount of traffic to be routed between node i and all other nodes in the network. So $r_i = \sum\limits_{k \in K_i} d_k$ where $K_i = \{k \,/\, s(k) = i \text{ or } t(k) = i\}$.

Obviously, a necessary condition for $x = (x_u)_{u \in U}$ to be a feasible solution to (P_1) is that

$$\sum_{u \in \omega(i)} x_u \geq r_i \tag{2}$$

where $\omega(i)$ denotes the subset of edges having i as an endpoint.

For convenience denote t the index number of edges in $\omega(i)$ for t running from 1 to $\delta(i) = |\omega(i)|$. We agree that $u = \alpha_i(t)$ is the index number of the t^{th} edge in $\omega(i)$.

Since, $\forall u \in U$, V_u is a finite discrete set, then the system

$$(I)\begin{cases} \sum_{u\in\omega(i)} x_u \geq r_i \\ x_u \in V_u \quad \forall u \in \omega(i) \end{cases}$$

has only finitely many distinct solutions. Each solution may be described as a vector with $\delta(i)$ components.

We denote A^i the matrix, the columns of which are the various vectors solving (*I*). It has $\delta(i)$ rows indexed by $t = 1,\ldots,\delta(i)$, and $P(i)$ columns, indexed $p = 1,\ldots,P(i)$. $A^i_{t,p}$ denotes the entry of A^i in the t^{th} row and the p^{th} column. From the definition $A^i_{t,p} \in V_u$ where $u = \alpha_i(t)$.

Also with each column p of the matrix A^i ($p \in [1, P(i)]$) we associate a cost:

$$\gamma^i_p = \sum_{t=1\ldots\delta(i)} \Phi_{\alpha_i(t)}(A^i_{t,p})$$

(this is the part of the objective function value corresponding to the edges in $\omega(i)$ only, when the capacities installed on those edges are the ones appearing as the components of the p^{th} column vector of A^i).

Now, associated with each column p of the matrix A^i we consider a 0-1 decision variable y^i_p expressing the fact that we select ($y^i_p = 1$) or not ($y^i_p = 0$) the p^{th} column of A^i to be part of the solution. The corresponding cost of selecting one and only one of the column vectors of A^i to be part of the solution is

$$\sum_{p=1}^{P(i)} \gamma^i_p \times y^i_p$$

the binary y^i_p variables being constrained to satisfy:

$$\sum_{p=1}^{P(i)} y^i_p = 1$$

Let $\overline{x} = (\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n)$ be any solution of (P_1). Then clearly the subvector of $\overline{x}$ formed by the components $\overline{x}_u$ with $u \in \omega(i)$ satisfies (*I*). So there exists a 0-1 y^i vector, denote it $\overline{y}^i$, such that :

$$A^i \overline{y}^i = \begin{pmatrix} \overline{x}_{\alpha_i(1)} \\ \overline{x}_{\alpha_i(2)} \\ \cdots \\ \overline{x}_{\alpha_i(\delta(i))} \end{pmatrix} \tag{3}$$

and

$$e^T \overline{y}^i = 1$$

(e denotes the vector of all ones of appropriate dimension, here $P(i)$).

For convenience we introduce the $\delta(i) \times n$ 0-1 matrix B^i such that:

$$\begin{pmatrix} \overline{x}_{\alpha_i(1)} \\ \overline{x}_{\alpha_i(2)} \\ \cdots \\ \overline{x}_{\alpha_i(\delta(i))} \end{pmatrix} = B^i \times \begin{pmatrix} \overline{x}_1 \\ \overline{x}_2 \\ \cdots \\ \overline{x}_n \end{pmatrix}$$

Note that B^i is contructed as follows: for $t \in [1, \delta(i)]$ its only nonzero coefficent in row t has value 1 and belongs to column $\alpha_i(t)$.

Therefore equation (3) may be rewritten as:

$$A^i \overline{y}^i - B^i \overline{x} = 0 \tag{4}$$

Now, all the construction above may be carried out for every node $i = 1,2,...,m$ of the network in turn.

From this it is shown that, if $\overline{x}$ is any solution to (P_1) then there exist 0-1 vectors $\overline{y}^i$ ($i = 1,...,m$) satisfying the following set of constraints:

$$\begin{cases} \overline{x} \in X \text{ and:} \\ \forall i = 1,\dots,m: \\ \qquad A^i \overline{y}^i - B^i \overline{x} = 0 \\ \qquad e^T \overline{y}^i = 1 \\ \qquad \overline{y}^i \in \{0,1\}^{P(i)} \end{cases}$$

Note that, in the above, the requirement $\overline{x} \in V_1 \times V_2 \times ... \times V_m$ would be redundant.

Moreover, the cost of $\overline{x}$: $\sum_{u \in U} \Phi_u(\overline{x}_u)$ may be rewritten as:

$$\sum_{u \in U} \Phi_u(\overline{x}_u) = \frac{1}{2} \sum_{i=1}^{m} \sum_{u \in \omega(i)} \Phi_u(\overline{x}_u) = \frac{1}{2} \sum_{i=1}^{m} \sum_{p=1}^{P(i)} \gamma_p^i \overline{y}_p^i \tag{5}$$

Consider then the following mixed integer linear programming problem:

$$(P_2) \begin{cases} \text{minimize } z = \frac{1}{2} \sum_{i=1}^{m} \sum_{p=1}^{P(i)} \gamma_p^i y_p^i \\ \text{subject to:} \\ \forall i = 1,\dots,m: A^i y^i - B^i x = 0 & (6) \\ \qquad\qquad e^T y^i = 1 & (7) \\ \qquad\qquad y^i \in \{0,1\}^{P(i)} & (8) \\ x \in X & (9) \end{cases}$$

Property 1:

(P_1) and (P_2) are equivalent.

Proof:

From the above we know that for any solution to (P_1) we can find a solution to (P_2) which, due to (5), has the same cost function value. Conversely, for any solution to (P_2) equation (6) (7) and (8) ensure that the x vector belongs to $V_1 \times V_2 \times ... \times V_m$ and, from the definition of the A^i matrices, no solution in $X \cap (V_1 \times V_2 \times ... \times V_m)$ is excluded. Therefore the x variables form a solution to (P_1) and again the objective function values for (P_1) and (P_2) are the same. This proves equivalence between (P_1) and (P_2) ■

We note here that for problems of practical size (m = 100 nodes, n = 200 to 500 edges, say), (P_2) will have a huge number of y variables. In practical networks, the node degrees typically range from 3 to 6-7. Assuming that each edge cost function features an average of 10 discontinuity points, an estimate of the typical number of columns in each A^i matrix is between 10^3 and 10^7. On the other hand, observe that the total number of constraints (6)-(7) is limited to $m + 2n$.

Another aspect of the problem is that, if we want an accurate linear description of the feasible multicommodity flow polyhedron X, a large number of linear constraints will have to be explicitly brought into the model.
Therefore, for examples of practical size, we can expect (P_2) to be a (very) large scale mixed integer linear program for which there is no hope of getting guaranteed exact optimal solutions with the best currently available techniques.

4. Large scale LP relaxations

In order to work out practically solvable LP relaxations of (P_2) we will combine two ideas:

(*i*) Generate and use only a part of the linear inequalities (1) which define the feasible multicommodity flow polyhedron. For convenience, the inequalities actually appearing in the relaxed model will be written in matrix form

$$Fx \geq f \qquad (9)'$$

where F is a known $r \times n$ matrix and f a known right handside vector of dimension r.
In section 4.2 we will discuss the possibility of iteratively improving the relaxation by generating additional valid inequalities.

(*ii*) Replace the integrality requirements (8) by simply $y^i \geq 0$ (observe that $y^i \geq 0$ together with constraint (7) guarantees that each y^i_p variable will be in the range [0,1]).

We will therefore study LP relaxations of (P_2) of the following general form:

$$(P_2)' \begin{cases} \text{minimize } z = \dfrac{1}{2}\sum_{i=1}^{m}\sum_{p=1}^{P(i)} \gamma^i_p y^i_p \\ \text{subject to:} \\ \forall i = 1,\ldots,m\colon A^i y^i - B^i x = 0 & (6) \\ \qquad\qquad\quad e^T y^i = 1 & (7) \\ \qquad\qquad\quad y^i \geq 0 & (8) \\ Fx \geq f & (9)' \end{cases}$$

4.1 Solving $(P_2)'$ to optimality through generalized linear programming techniques

The idea here is to explicitly handle only a few columns out of each A^i matrix, i.e. to solve only a restricted LP, and iteratively generate negative reduced cost columns when needed. This column generation procedure is completed when a state has been reached where no new negative reduced cost column can be generated. To guarantee that the procedure yields an optimal continuous solution to the whole problem $(P_2)'$ we only have to check that the *column generation subproblem* can be solved exactly at each step (see e.g Minoux 1983, chapter 8). We show below how to solve this subproblem.

The columns of matrix A^i correspond to the various vectors solving (*I*). Then generating negative reduced cost columns of this matrix, or proving that no negative reduced cost column exists, leads solving the following subproblem:

$$
(Q)\begin{cases}\text{minimize } w=\sum\limits_{u\in\omega(i)}\Phi_u(x_u)-\sum\limits_{u\in\omega(i)}\pi_u x_u\\ \sum\limits_{u\in\omega(i)}x_u\geq r_i\\ x_u\in V_u\quad \forall u\in\omega(i)\end{cases}
$$

where $\pi=(\pi_u)_{u\in\omega(i)}$ is the vector of optimal simplex multipliers for the current restricted LP.

This column generation subproblem can be solved exactly with a dynamic programming algorithm as shown in appendix 1.

4.2 The relaxations: (R_1) and (R_2)

There are many possibilities for choosing constraints expressing necessary conditions for x to be a feasible multicommodity flow. Among all those possibilities we have chosen the following ones for our computational experiments.

Relaxation (R_1)

For building relaxation (R_1) we set up the constraints (9)′ by including inequalities of the form (1):

$$
\sum_{u\in U}\lambda_u x_u\geq\theta(\lambda)
$$

These are generated iteratively according to a cutting plane technique. The constraint generation procedure used in the computational experiments of section 5 is based on the (approximate) solution of the dual feasible multicommodity flow subproblem by means of a subgradient algorithm (for details see Gondran & Minoux 1995, chapter 6). Of course the constraint generation subproblem could also be solved exactly as a linear program using CPLEX.

Relaxation (R_2)

To set up (R_2) we define additional constraints on the y_p^i variables. Let u be an edge linking the nodes i and j. We denote $C^i(x_u)$ the set of all column numbers p of A^i in which the capacities installed on edge u is equal to x_u ($x_u \in V_u$). It is obvious that any solution to (P_2) verifies the following constraints:

$$
\sum_{p\in C^i(x_u)}y_p^i-\sum_{p'\in C^j(x_u)}y_{p'}^j=0\quad \forall x_u\in V_u,\forall u\in U \tag{10}
$$

Indeed, if the p^{th} column of A^i, for which the capacity installed on edge u is x_u, is part of the solution, then the column of A^j chosen to be part of the solution has necessarily the same capacity x_u for u.

Therefore, to set up (R_2), we add all possible constraints (10) to (R_1). Note that the number of constraints (10) is $\sum_{u=1}^{n}q(u)$.

In (R_2) the constraints (9)′ are generated in the same way as for (R_1).

Remark: The relaxations (R_1) and (R_2) above readily extend to the case where X is the polyhedron of feasible solutions associated with a set of nonsimultaneous multicommodity flow requirements. In that case, the constraints (9)′ in (P_2)′ should include all the sets of constraints (1) corresponding to the various individual multicommodity flow requirements ‖

5. Computational experiments

5.1 Data

The LP relaxations described above have been implemented and tested on a series of test problems derived from a basic network structure (see figure 2).

The basic network considered has 16 nodes and 25 edges. The distances (in kilometers) between these nodes are presented in matrix 1 of appendix 2. The multicommodity flow requirements between each node pair are presented in matrix 2 of appendix 2.

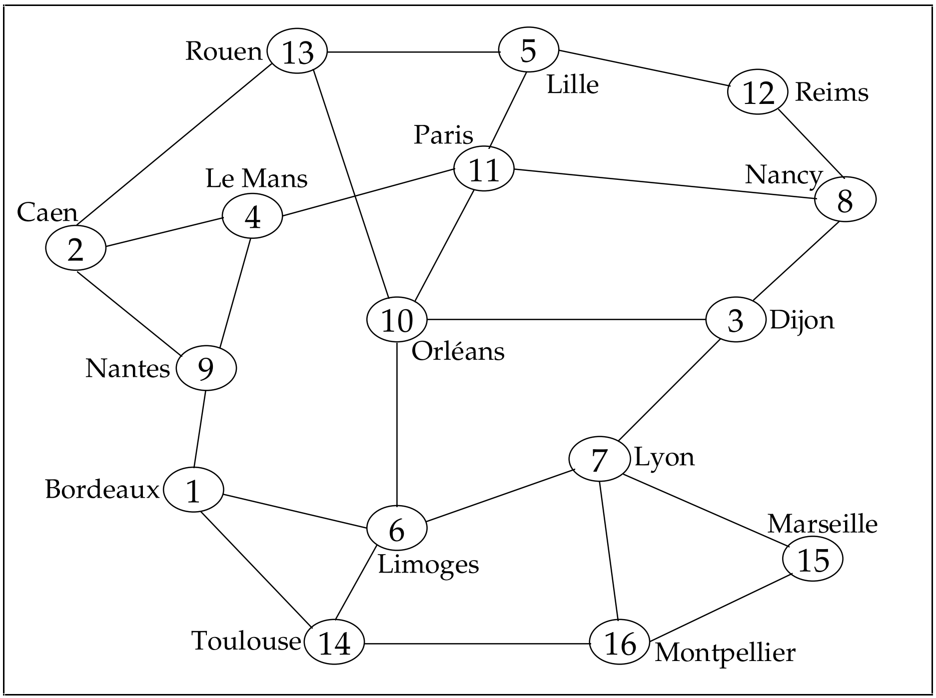

Figure 2: The basic network considered

To compute the upper bound β_u for each variable x_u, we first determine a feasible multicommodity flow in which each individual flow k ($k \in [1, K]$) is routed on the minimum length (in terms of distance) chain between $s(k)$ and $t(k)$. And $\forall u \in U$, β_u is equal to twice the flow carried by edge u in this solution.

We define for each edge $u \in U$ a cost function $\Phi_u(x_u)$ in the following way:

1. choose the number $q(u)$ of discontinuity points (in the computational experiments reported in section 5.3 $q(u)$ has been chosen equal to 14),
2. draw at random the $q(u)$ discontinuity points in the range $[0, \beta_u]$,
3. compute the cost γ_u^i associated with each discontinuity point v_u^i $(i = 0, \ldots, q(u))$ according to the formula:

$$\gamma_u^i = \Phi_u(v_u^i) = a(v_u^i)^\alpha d_u$$

where α is a coefficient chosen in the range [0.7,1], a is a random number chosen in the range [0.9, 1.1] and d_u is the distance (in kilometer) of link u.

In appendix 3, two examples of such function $\Phi_u(x_u)$ are presented.

5.2 Computing upper bounds

In order to assess the quality of our LP relaxations, we have to compare the optimal solution values to (R_1) and (R_2) to a feasible solution value of the original minimum cost multicommodity flow problem.

The approximate algorithm we use is a variant of the one proposed by Minoux (see Minoux 1989). Starting from a feasible multicommodity flow solution, the main idea of this algorithm is to rerout any edge carrying strictly positive flow in this solution on another single chain if and only if the resulting global cost is lower. At the beginning, we determine a feasible multicommodity flow by routing each individual flow k ($k \in [1, K]$) on the shortest chain (in terms of number of edges) between $s(k)$ and $t(k)$. And at each iteration, we compute for each edge u carrying strictly positive flow, the minimum extra cost of globally rerouting all the flow which was running through edge u on a single alternate chain in the network. Then the flow carried by the edge which achieves the minimum of the extra costs previously computed is rerouted on the associated chain. The algorithm stops when all the minimum extra costs are nonnegative.

In order to adapt this algorithm to our particular multicommodity flow problem with step increasing cost functions, we compute at each iteration and for each edge u the minimum extra cost of rerouting not all but a part of the flow which was running through edge u. For each edge u, this flow decomposition is function of V_u.

5.3 Some computational results

For solving LP relaxations (R_1) and (R_2) we use CPLEX library routines. We present 4 series of results for $\alpha = 0.7$ (table 1), $\alpha = 0.8$ (table 2), $\alpha = 0.9$ (table 3) and $\alpha = 1$ (table 4). Each table displays results corresponding to 3 distinct random instances of problem (P_2) generated on the same basic network of figure 2. For each example solved, the tables show:

LP: size of linear program $(P_2)'$ without constraint (9)′

HS: heuristic solution value (upper bound)

LB_1: first lower bound obtained by solving (R_1)

LB_2: second lower bound obtained by solving (R_2)

NC: number of constraints (9)′ in $(P_2)'$

$r = \dfrac{\text{HS} - \text{LB}_2}{\text{HS}} (\%)$

These preliminary computational results were primarily intended to validate the proposed approach in terms of quality of the resulting bounds, on a few typical test problems of reduced size. Therefore we did not need to implement at this stage, the column generation procedure (which would be necessary to deal with problems of larger sizes). The LP sizes in the tables therefore show the total number of columns in $(P_2)'$.

Also the process of generating constraints (9)′ was stopped when « almost feasible » solutions were obtained (more precisely when the last generated constraints were such that the relative difference between the left handside $\sum_{u \in U} \lambda_u \bar{x}_u$ and the right handside $\theta(\lambda)$ was less than 0.001).

Table 1: $\alpha = 0.7$

LP	HS	LB_1	NC for (R_1)	LB_2	NC for (R_2)	r
67×62607	1094	885	300	934	300	14.62
67×76658	1071	850	223	857	232	19.98
67×74788	1092	862	285	873	285	20.05

Table 2: $\alpha = 0.8$

LP	HS	LB_1	NC for (R_1)	LB_2	NC for (R_2)	r
67×72629	1755	1511	282	1545	343	11.96
67×95487	1721	1407	201	1414	203	17.83
67×81083	1797	1444	260	1467	260	18.36

Table 3: $\alpha = 0.9$

LP	HS	LB_1	NC for (R_1)	LB_2	NC for (R_2)	r
67×76286	2779	2499	253	2506	253	9.82
67×104892	2719	2329	229	2336	236	14.08
67×99707	2873	2378	258	2385	258	16.98

Table 4: $\alpha = 1$

LP	HS	LB_1	NC for (R_1)	LB_2	NC for (R_2)	r
67×83316	4479	4002	255	4004	290	10.6
67×109658	4333	3797	185	3803	186	12.23
67×98765	4636	3919	261	3923	261	15.37

The computational results in tables 1-4 show that the differences between the lower bounds (obtained from the LP relaxations) and the upper bounds (obtained from the heuristic primal solutions) consistently lie in the range [10%,20%]. As expected we observe that the lower bounds produced by (R_2) are always better than those given by (R_1), though the differences are usually rather small (between 0.1% and 5%).

The gap between lower bounds and upper bounds might still be reduced

- by further investigating the possibility of improving the quality of primal solutions (reducing upper bounds),
- by further investigating the possibility of strengthening the LP relaxation (R_2) by introducing new classes of extra constraints to be satisfied by any integral feasible solution (integrality cuts).

Another area for future research will be to investigate the possibility of deriving good primal solutions from the optimal x_u values in the LP relaxations.

Aknowledgements: We thank Prof. E. Boros for some fruitful discussions concerning relaxation (R_2).

References

Ahuja R.K., Magnanti T., Orlin J. (1993), *Network flows: Theory, Algorithms and Application*, Prentice Hall.

Assad A.A. (1978), Multicommodity Network Flows - A Survey, *Networks* **8**, pp.37-91.

Fratta L., Gerla M., Kleinrock L. (1973), The Flow-Deviation Method: an Approach to Store-and-Forward Communication Network Design, *Networks* **3**, pp.97-133.

Gondran M., Minoux M (1995), *Graphes et Algorithmes*, Eyrolles, Paris (3ème édition).

Kennington J.L. (1978), A survey of linear cost Multicommodity Network Flows, *Operations Research* **26**, 2, pp.209-236.

Minoux M. (1983), *Programmation mathématique: théorie et algorithmes*, Dunod, Paris, English translation, John Wiley & Sons, New York, 1986.

Minoux M. (1989), Network Synthesis and Optimum Network Design Problems: Models, Solution Methods and Applications, *Networks* **19** pp.313-360.

Ouorou A. (1995), Décomposition proximale des problèmes de multiflots à critère convexe - Application aux problèmes de routage dans les réseaux de communication, *Thèse de Doctorat*, Université de Clermont Ferrand, 17 novembre 1995.

Appendix 1

For solving (Q) in subsection 4.1, let us first consider the following set of problems:

$$(Q_j(E))\begin{cases} \text{minimize } w_j(E) = \sum\limits_{u=1}^{j} \Phi_u(x_u) - \sum\limits_{u=1}^{j} \pi_u x_u \\ \sum\limits_{u=1}^{j} x_u \geq E \\ x_u \in V_u \quad \forall u = 1,\ldots,j \end{cases}$$

for j running from 1 to $\delta(i)$. Here we have assumed for simplicity that the edges in $\omega(i)$ are numbered $u = 1,\ldots,\delta(i)$.

The optimal solution value w^* of (Q) can be obtained by solving problems $(Q_{\delta(i)}(E))$ for all possible values of E since:

$$w^* = \min_{E \geq r_i}\left\{w^*_{\delta(i)}(E)\right\}$$

where $w^*_{\delta(i)}(E)$ is the optimal solution value of $(Q_{\delta(i)}(E))$.

There is a classical recurrence relation between the optimal solution values $w^*_j(E)$ of these different problems $(Q_j(E))$:

$$w^*_j(E) = \min_{x_j \in V_j}\left\{(\Phi_j(x_j) - \pi_j x_j) + w^*_{j-1}(E - x_j)\right\}$$

with $w_j^*(E) = +\infty$ if there is no solution to $(Q_j(E))$ (in particular when $E < 0$).

So for determining w^*, we have to solve iteratively all the problems $(Q_j(E))$ for j running from 1 to $\delta(i)$. This algorithm is presented in detail in figure 3. For solving problem $(Q_j(E))$, we retain among all the possible pairs of value (E', x_j) such as $\forall x_j \in V_j$, $\forall E'$: $w_{j-1}(E') \neq +\infty$, $E = E' + x_j$, the one of minimum cost. Moreover, we use two procedures called **efficiency** and **solution** respectively. The procedure called **efficiency** (see figure 4) deletes an optimal solution $w_j(E')$ of $Q_j(E')$ when there exists an optimal solution $w_j(E)$ of $Q_j(E)$ such as: $E > E'$ and $w_j(E) < w_j(E')$. Indeed, if $w_j(E')$ is part of a solution of (Q), it is possible to replace $w_j(E')$ by $w_j(E)$ in order to obtain another feasible solution with lower cost. Thus $w_j(E')$ would never be part of an optimal solution of (Q). The procedure called **solution** is not described here, it simply outputs the best solution of (Q) by using the $S_j(E)$ pointers.

```
w_0(0) = 0
w_0(E) = +∞      ∀E = 1,...,r_i
S_j(E) = ∅       ∀E = 0,1,...,r_i, ∀j = 0,1,...,δ(i)
for j = 1 to δ(i)
        w_j(E) = +∞    ∀E = 0,1,...,v_j^{q(j)}
        for x_j = v_j^0 to v_j^{q(j)}
                forall E such as w_{j-1}(E) ≠ +∞
                        if (w_j(E + x_j) > Φ_j(x_j) − π_j x_j + w_{j-1}(E))
                                w_j(E + x_j) ← Φ_j(x_j) − π_j x_j + w_{j-1}(E))
                                S_j(E + x_j) ← {(u, x_j, S_{j-1}(E))}
                        end if
                end forall
        end for
        efficiency(j)
end for
solution(i)
```

Figure 3: The dynamic programming algorithm

```
forall E such as w_j(E) ≠ +∞
        forall E' such as w_j(E') ≠ +∞ and E' < E
                if (w_j(E') > w_j(E))
                                w_j(E') ← +∞
                                S_j(E') ← ∅
                end if
        end forall
end forall
```

Figure 4: The procedure **efficiency**(j)

Appendix 2

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1						175			275					225		
2				138					238				100			
3							175	175		238						
4		138							150		200					
5											200	163	213			
6	175						288			238				250		
7			175			288									288	250
8			175								275	163				
9	275	238		150												
10			238			238					113		188			
11				200	200			275		113						
12					163			163								
13		100			213					188						
14	225					250										213
15							288									150
16							250							213	150	

Matrix 1 : Distances in kilometers between nodes

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	2														
3	4	2													
4	2	2	2												
5	4	4	9	3											
6	5	1	3	1	3										
7	9	3	27	3	12	7									
8	4	3	18	2	14	3	19								
9	10	5	7	6	9	5	12	7							
10	2	2	4	2	5	2	6	4	5						
11	13	13	30	11	53	9	40	39	30	28					
12	1	1	3	1	4	1	3	4	2	1	13				
13	3	6	5	3	10	2	7	6	7	4	42	2			
14	15	3	8	3	7	7	21	8	12	4	23	2	5		
15	8	2	11	2	8	5	33	11	9	4	25	2	5	23	
16	3	1	3	1	2	2	9	3	3	1	7	1	2	9	15

Matrix 2: Flow requirements in example

Appendix 3

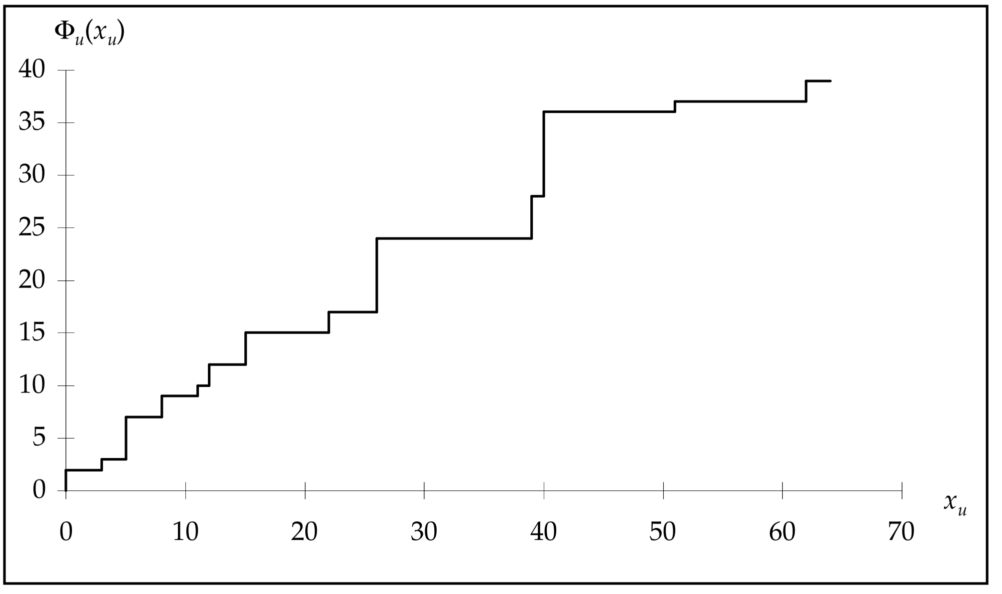

Figure 5: A typical cost function with α = 0.8

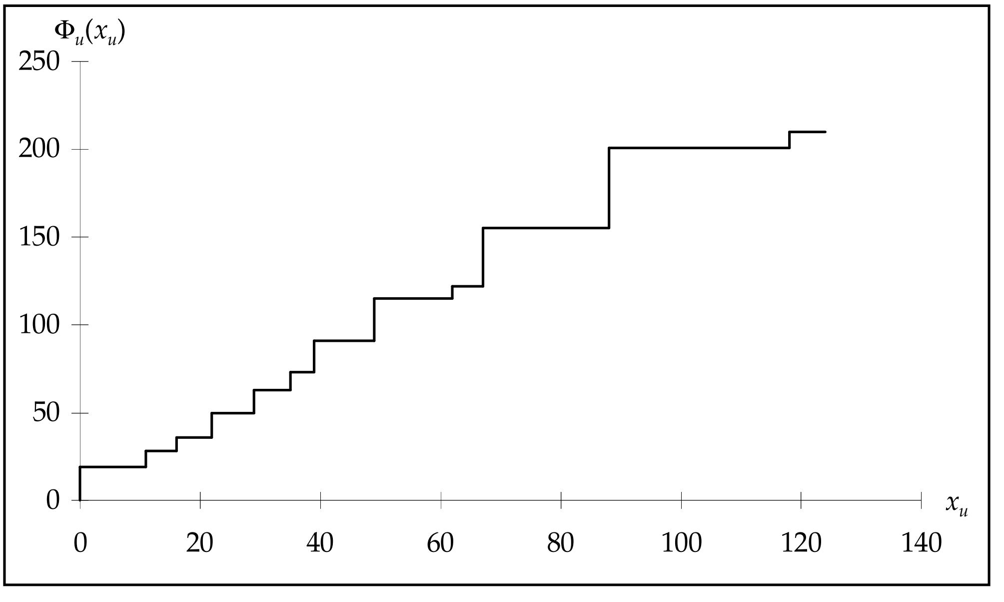

Figure 6: A typical cost function with α = 1