ACTA MATHEMATICA VIETNAMICA
Volume 22, Number 1, 1997, pp. 123–145

LP RELAXATIONS BETTER THAN CONVEXIFICATION FOR MULTICOMMODITY NETWORK OPTIMIZATION PROBLEMS WITH STEP INCREASING COST FUNCTIONS

V. GABREL AND M. MINOUX

Dedicated to Hoang Tuy on the occasion of his seventieth birthday

Abstract. We address here a class of particularly hard-to-solve combinatorial optimization problems namely multicommodity network optimization when the link cost functions are discontinuous step increasing. The main focus is on the development of relaxations for such problems in order to derive lower bounds. A straightforward way of getting lower bounds is to solve the "convexification" of the problem, i.e. a minimum cost multicommodity flow problem in which each link cost function is replaced by its best lower convex approximation. We propose here an alternative relaxation of the problem in terms of a large scale LP model, which can be solved by a generalized linear programming approach. Computational results on a series of test problems, typical of telecommunication networks, are presented showing that this relaxation leads to lower bounds strictly improving upon convexification in all the examples treated (the values of the improvement typically ranging from 10 to 20%).

As far as we know, this is the first time a systematic way of generating bounds better than convexification to multicommodity network optimization problems with discontinuous step increasing cost functions is proposed.

1. Introduction

Minimum cost multicommodity network optimization 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

Received January 30, 1997

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 e.g. 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. As far as we know, no exact solution method for such combinatorial optimization problems (for practical size instances) has ever been proposed, as they appear to be out of reach of state of the art techniques in the field. Therefore our aim in the present paper will be to bring a first contribution to this difficult subject by addressing the problem of building relaxations and generating lower bounds.

A first natural way for getting lower bounds could be to use "convexification": replacing each link cost function by its best lower convex approximation, a piecewise linear convex cost multicommodity flow problem is obtained, which can be formulated and solved as a standard linear programming problem (cf e.g. Gondran & Minoux 1979). In order to get better lower bounds, we propose here a relaxation of the problem in terms of a large scale LP model, the exact optimal continuous solutions of which may be obtained by combining both column generation and constraint generation. Of course the resulting optimal solutions provide lower bounds to the (unknown) exact solutions for the minimum cost multicommodity flow problem to be solved. Computational results are presented on a series of test problems with structures similar to those encountered in telecommunication networks. They show that this novel relaxation yields lower bounds strictly improving upon convexification in all the examples treated, the resulting improvement typically ranging from 10 to 20%.

As far as we know, this is the first time a systematic way of generating bounds better than convexification to multicommodity network optimization problems with discontinuous step increasing cost functions is proposed.

2. Problem statement and formulation

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

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 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:

$$\forall u \in \mathcal{U} : 0 \leq x_u \leq \beta_u,$$

- 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:

$$z = \sum_{u \in \mathcal{U}} \Phi_u(x_u).$$

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 nontrivial lower bounds for such problems.

For each edge $u \in \mathcal{U}$ in the network, we assume that we are given a cost function $\Phi_u(x_u)$ which is 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 functions and denote

$$\begin{aligned}
\gamma_u^0 &= \Phi_u(v_u^0),\\
\gamma_u^1 &= \Phi_u(v_u^1),\\
\gamma_u^2 &= \Phi_u(v_u^2),\\
\cdots &\quad \cdots\\
\gamma_u^{q(u)} &= \Phi_u(v_u^{q(u)}),
\end{aligned}$$

with $0 = v_u^0 \leq v_u^1 \leq v_u^2 \leq \cdots \leq v_u^{q(u)}$ and $0 = \gamma_u^0 < \gamma_u^1 < \gamma_u^2 < \cdots < \gamma_u^{q(u)}$.

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: $0 \leq x_u \leq \beta_u$ either explicity or implicity.

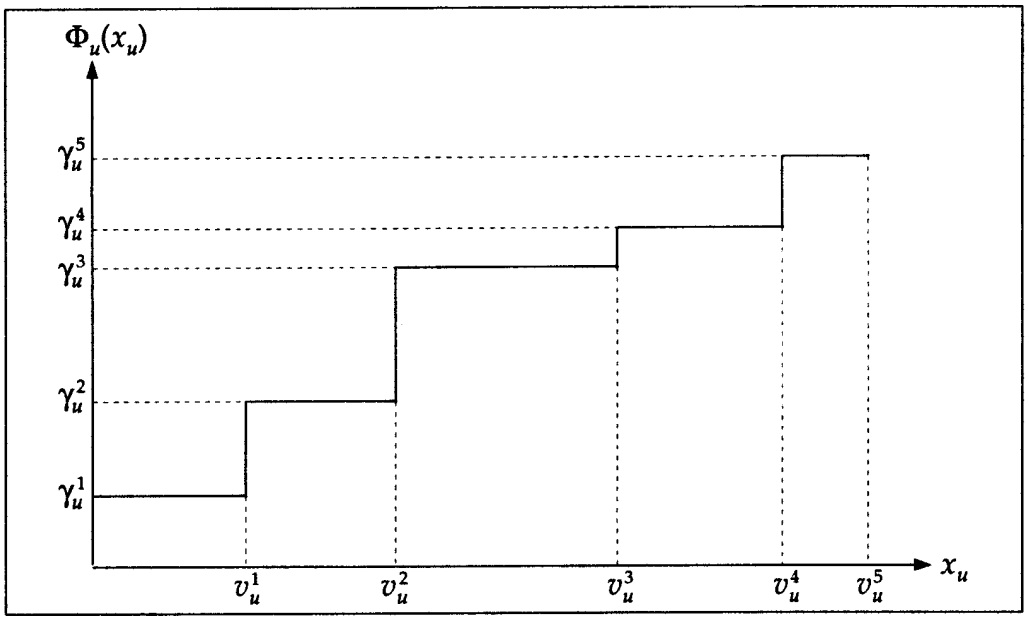

Fig. 1. A typical cost function on one link

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 $X \subset \mathbf{R}^n_+$ the polyhedron representing the set of all feasible multicommodity flows. Thus $x = (x_u)_{u \in \mathcal{U}}$ belongs to X if and only if a feasible multicommodity flow exists when, on each edge $u \in \mathcal{U}$, the total capacity installed is x_u (of course $x_u \geq 0$ 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 node-arc formulation and arc-chain formulation (for an overview, see e.g. Kennington 1978, Minoux 1989).

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

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

where $\ell^*_k(\lambda)$ is the length of the shortest chain joining $s(k)$ and $t(k)$ in G, when each edge $u \in \mathcal{U}$ is given length $\lambda_u \geq 0$ (note that $\theta(\lambda)$ may be interpreted as the value of 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 $x = (x_u)_{u \in \mathcal{U}}$ belongs to X if and only if, for all $\lambda \in \mathbf{R}^n_+$, we have

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

(see e.g. Gondran & Minoux 1979, Chapter 6).

Constraint (1) apparently gives rise to infinitely many inequalities. However it is well known that it is enough to restrict to finitely many λ vectors, namely those corresponding to the extreme points of the polytope dual to the feasible multicommodity flow problem. Moreover, testing whether a given $\overline{x} \in \mathbf{R}^n$ belongs to X can be done in polynomial time since this amounts to solving a linear program.

Now, using the feasible multicommodity flow polyhedron X, a possible formulation of the minimum cost multicommodity flow problem we want to investigate is

$$(P_0) \begin{cases} \text{Minimize} & \displaystyle\sum_{u \in \mathcal{U}} \Phi_u(x_u), \\ \text{s.c.} & x \in X, \\ & 0 \leq x_u \leq \beta_u \quad \forall u \in \mathcal{U}. \end{cases}$$

We first show:

Proposition 1. *Problem* (P_0) *is equivalent to:*

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

Proof. Since, $\forall u \in \mathcal{U} : V_u \subseteq [0, \beta_u]$ (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 $\left(x_u^0\right)_{u\in\mathcal{U}}$ to (P_0) may be converted into a solution to (P_1) having the same objective function value.

For any $u \in \mathcal{U}$ there exists $i \in [1, q(u)]$ such that $v_u^{i-1} < x_u^0 \leq v_u^i$. 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 is a feasible solution to (P_1) such that $\Phi(x^1) = \Phi(x^0)$. □

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 amount of flow requirements 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 $K_i = \left\{k/s(k) = i \text{ or } t(k) = i\right\}$.

Obviously, a necessary condition for $x = (x_u)_{u\in\mathcal{U}}$ 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.

Also, 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, $\forall u \in \mathcal{U}$, V_u is a finite discrete set, then the system

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

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

We denote by 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, denoted by $\overline{y}^i$, such that

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

and

$$e^T \overline{y}^i = 1 \tag{4}$$

(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:

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

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

Therefore equation (3) may be rewritten as

$$A^i \overline{y}^{\,i} - B^i \overline{x} = 0. \tag{5}$$

Now, all the construction above may be carried out for every node $i = 1, 2, \ldots, m$ of the network in turn. From this it is seen that, if $\overline{x}$ is any solution to (P_1), then there exist 0-1 vectors $\overline{y}^{\,i}$ $(i = 1, \ldots, m)$ satisfying the following set of constraints

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

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

Consider then the following mixed integer linear programming problem

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

Proposition 2. *(P_1) and (P_2) are equivalent.*

Proof. Let $\overline{x}$ be any solution to (P_1) and let $\overline{y}^i$ $(i = 1, \ldots, m)$ be the associated 0-1 vectors satisfying (II). We note that the cost of $\overline{x}$: $\sum_{u \in \mathcal{U}} \Phi_u(\overline{x}_u)$ may be rewrriten as

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

From this we conclude that for any solution to (P_1) we can find a solution to (P_2) which has the same cost function value.

Conversely, for any solution to (P_2), equations (6) (7) and (8) ensure that the x vector belongs to $V_1 \times V_2 \times \cdots \times V_m$ and, from the definition of the A^i matrices, no solution in $X \cap (V_1 \times V_2 \times \cdots \times V_m)$ 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). □

The number of y variables in (P_2) depends on the number of nodes ($m \geq 50$ is typical in problems of pratical size), the node degrees (typically an average of 5) and the number of discontinuity points on each edge cost function (typically an average of 5). Therefore, an estimate of the typical number of columns in (P_2) is $m \times 5^5$. 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 sufficient number of linear constraints on the x variables will have to be explicitly brought into the model. Therefore, for examples of practical size ($m \geq 50$ nodes, $n \geq 80$ edges, say), we can expect (P_2) to be a 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 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

$$Fx \geq f, \tag{9'}$$

where F is a known $r \times n$ matrix and f a known right handside vector of dimension r.

(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 $[0, 1]$).

This leads to a first LP relaxation (R_1) of the following general form

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

In Subsection 4.2, we show how to further improve relaxation (R_1) *by adding valid inequalities on the y variables.*

Remark. We observe here that for some instances of the problem, $(P_2)'$ may have a huge number of y variables. For instance, for a node i having degree 5, if the cost function of each adjacent link of node i has 10 discontinuity points, the corresponding A^i matrix may have 5^{10} columns. In that case the use of a *column generation technique* will be necessary. We show in Appendix 1 how to solve the resulting column generation subproblem with a dynamic programming-based approach.

4.1. Building relaxation (R_1) via constraint generation

For building our first relaxation (R_1) we have chosen a constraint generation approach in which the constraints (9') consist of inequalities of the form (1)

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

There are generated iteratively according to a cutting plane technique. The current step of the procedure is as follows:

1) Solve the LP problem $(P_2)'$ including all constraints (9') generated so far. Let $\overline{x} = (\overline{x}_u)_{u \in \mathcal{U}}$ be the resulting optimal solution to this LP, and $\overline{z}$ the optimal solution value.

2) In order to check whether $\overline{x} \in X$ or to generate a new constraint (9'), we solve the following optimization subproblem:

$$\text{(10)} \qquad \begin{cases} \max \theta(\lambda), \\ \sum\limits_{u \in \mathcal{U}} \lambda_u \overline{x}_u = 1, \\ \forall u \in \mathcal{U}, \quad \lambda_u \geq 0. \end{cases}$$

Let λ^* be an optimal solution to (10). If $\theta(\lambda^*) > 1$, then the constraint $\sum\limits_{u \in \mathcal{U}} \lambda^*_u x_u - \theta(\lambda^*) \geq 0$ should be added to $(P_2)'$. Otherwise $\overline{x} = (\,\overline{x}_u\,)_{u \in \mathcal{U}}$ is a feasible multicommodity flow and $\overline{z}$ is the lower bound obtained by solving (R_1).

In practice, since we are building relaxations of (P_2), it will be enough to solve (10) *approximately*. Indeed if we use $\overline{\lambda}$, an approximate optimal solution to (10), instead of λ^*, it remains true that, when $\theta(\overline{\lambda}) > 1$ the constraint $\sum\limits_{u \in \mathcal{U}} \overline{\lambda}_u x_u - \theta(\overline{\lambda}) \geq 0$ is a necessary condition for feasibility. Therefore it may be added to $(P_2)'$ to build an improved relaxation.

In the computational experiments of Section 5, we use a subgradient algorithm to determine a near-optimal solution to (10).

We have $\theta(\lambda) = \sum\limits_{u \in \mathcal{U}} \lambda_u \Psi_u(\lambda)$ where, $\forall u \in \mathcal{U}$, $\Psi_u(\lambda)$ is the total flow through edge u when, for each flow k, $k = 1, \ldots, K$, d_k is routed on the shortest chain joining $s(k)$ and $t(k)$ in G, each edge $u \in \mathcal{U}$ being given the length $\lambda_u \geq 0$. We observe that $\Psi = \big(\Psi_u(\lambda)\big)_{u \in \mathcal{U}}$ is a *subgradient* of function $\theta(\lambda)$. We state below the subgradient algorithm used in our experiments, where T is the maximum number of iterations and η the reduction factor of the step size at each iteration (in our experiments we have taken $T = 300$ and $\eta = 0.98$).

(a) $\forall u \in \mathcal{U}$, set $\lambda_u^{(0)} \leftarrow \dfrac{1}{\sum\limits_{u \in \mathcal{U}} \overline{x}_u}$, $t \leftarrow 0$ (iteration counter), $\xi^{(0)} \leftarrow \dfrac{1}{\sqrt{n}}$ (initial step size).

(b) At iteration t, let $\lambda^{(t)}$ be the current solution. Compute $\Psi(\lambda^{(t)}) = \big(\Psi_u(\lambda^{(t)})\big)_{u \in \mathcal{U}}$ (this is done via shortest path computations).

$$\forall u \in \mathcal{U}, \text{ set } w_u^{(t+1)} \leftarrow \lambda_u^{(t)} + \xi^{(t)} \frac{\Psi_u(\lambda^{(t)})}{\|\Psi(\lambda^{(t)})\|} \,,$$

$\lambda^{(t+1)}$ is the projection of $\omega^{(t+1)}$ onto $C = \Big\{ \sum\limits_{u \in \mathcal{U}} \lambda_u \overline{x}_u = 1, \ \lambda \geq 0 \Big\}$,

$\xi^{(t+1)} \leftarrow \eta\xi^{(t)}$ and go to (c).

(c) If $t < T$, set $t \leftarrow t+1$ and go to (b).

Otherwise stop: $\overline{\lambda}$ is taken as the best solution obtained during the iterations.

In the computational results shown in Section 5 we indicate, for each instance solved, the number of constants (9') generated to obtain the bound corresponding to relaxation (R_1).

4.2. An improved relaxation (R_2)

We show in this subsection that we can strengthen relaxation (R_1) by including additional constraints which are valid inequalities on the y_p^i variables, thus leading to an improved relaxation (R_2).

Let u be an edge linking the nodes i and j. For any $x_u \in V_{u'}$ we denote $C^i(x_u)$ the set of all column indices p of A^i in which the capacity installed on edge u is equal to x_u. Consider then the following constraints.

$$\sum_{p \in C^i(x_u)} y_p^i - \sum_{p' \in C^j(x_u)} y_{p'}^j = 0, \quad \forall x_u \in V_u, \ \forall u = (i,j) \in \mathcal{U}. \tag{11}$$

Proposition 3. *Constraints* (11) *are valid inequalities for* (P_2).

Proof. Let u be an edge linking the nodes i and j. 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. Consequently, any solution to (P_2) should meet the constraints (11). □

Therefore, to set up (R_2), we add all possible constraints (11) to (R_1); contraints (9') are also added and generated according to the same procedure as described in Section 4.1. Note that the number of constraints (11) is $\sum_{u=1}^{n} q(u)$ (we recall that for each u in $\mathcal{U}$, $q(u)$ denotes the number of discontinuity points of Φ_u - cf. Section 2).

The computational results presented in Section 5 below include, for each instance tested, the bound obtained with relaxation (R_2), the number of constraints (11), and the number of constraints (9') generated for solving the instance.

5. Preliminary computational experiments and conclusions

We describe in this section computational experiments aimed at comparing the results of relaxations (R_1) and (R_2) with those derived from the solution of the convexified problems.

In order to carry out our experiments, we have designed a random generator of instances of the min cost multicommodity flow problem with step increasing cost function resembling those encountered in real telecommunication network design applications. In this generator we have tried to reproduce some of the basic characteristics of real problems in particular:

- the node degrees are usually small, typically ranging from 2 to 6;
- the graph corresponding to the physical links is usually planar or almost planar;
- the link lengths are closely approximated by an Euclidian distance in 2-dimensions ($\mathbf{R}^2$).

A detailed description of the procedure used to generate our test problems (and the associated graphs $G = [S, \mathcal{U}]$) is given in Appendix 2.

For solving LP relaxations (R_1) and (R_2) we have used CPLEX library subroutines. We present 4 series of results for $|S| = 20$ (Table 1), $|S| = 30$ (Table 2), $|S| = 40$ (Table 3) and $|S| = 50$ (Table 4). Each table displays results corresponding to 30 distinct generated instances of Problem (P_2). For each example solved, the tables show:

NV: number of x variables (number of links),

ND: average number of discontinuity points of function $\Phi_u(x_u)\ \forall u \in \mathcal{U}$,

LP: size of the starting linear program $(P_2)'$ (without any constraint (9')),

C_1: number of generated constraints (9') in (R_1),

LB_1: first lower bound obtained by solving (R_1),

CY_2: number of constraints (11) in (R_2),

C_2: number of generated constraints (9') in (R_2),

LB_2: second lower bound obtained by solving (R_2),

$s = \dfrac{LB_2 - LB_1}{LB_2}(\%)$ (improvement of LB_2 upon LB_1),

LB_c: optimal solution value to the convexified problem,

$r = \dfrac{LB_2 - LB_c}{LB_2}(\%)$ (improvement of LB_2 upon convexification).

Table 1. Results for $|S| = 20$ nodes

NV	ND	LP	C_1	LB_1	CY_2	C_2	LB_2	s	LB_c	r
37	4.73	6932×95	33	263.42	138	45	269.54	2.27%	200.26	25.70%
36	4.64	6016×93	56	302.80	131	38	312.98	3.25%	253.66	18.95%
35	4.66	5179×91	27	402.54	128	29	426.11	5.53%	282.97	33.59%
35	4.60	4645×91	27	241.76	126	25	249.83	3.23%	215.05	13.92%
36	4.81	6677×93	60	484.07	137	62	508.13	4.73%	438.92	13.62%
35	4.74	5333×91	35	404.87	131	45	414.00	2.20%	324.13	21.71%
35	4.63	5048×91	61	298.20	127	48	320.49	6.96%	248.71	22.40%
38	4.79	8140×97	56	283.75	144	34	287.29	1.23%	268.14	6.67%
36	4.58	5299×93	70	338.63	129	60	362.21	6.51%	271.49	25.05%
35	4.57	4865×91	29	361.51	125	24	371.61	2.72%	254.76	31.44%
35	4.71	5616×91	68	323.48	130	41	339.63	4.76%	279.52	17.70%
36	4.67	5754×93	27	313.40	132	37	323.42	3.10%	259.86	19.65%
36	4.72	6355×93	33	424.66	134	69	447.72	5.15%	343.41	23.30%
36	4.64	5719×93	67	462.65	131	68	469.10	1.38%	419.37	10.66%
37	4.70	7208×95	35	261.73	137	51	270.01	3.07%	236.25	12.50%
35	4.74	5786×91	51	341.42	131	66	360.30	5.24%	299.89	16.77%
36	4.58	5734×93	41	171.92	129	35	184.50	6.82%	151.73	17.76%
35	4.60	5069×91	42	406.70	126	32	424.78	4.25%	334.66	21.21%
35	4.69	5435×91	59	412.40	129	54	425.93	3.18%	354.44	16.78%
37	4.59	6120×95	46	373.19	133	26	389.30	4.14%	312.76	19.66%
36	4.75	6348×93	28	307.65	135	32	322.17	4.51%	265.43	17.61%
36	4.67	6350×93	21	386.47	132	28	407.90	5.25%	303.01	25.72%
34	4.62	4459×89	25	284.60	123	22	300.07	5.16%	242.38	19.22%
35	4.69	4840×91	36	304.47	129	37	312.43	2.55%	264.09	15.47%
35	4.63	5064×91	31	375.87	127	36	385.42	2.48%	357.76	7.18%
37	4.65	6816×95	60	438.94	135	56	460.74	4.73%	400.35	13.11%
36	4.72	6292×93	48	333.22	134	65	355.57	6.29%	293.13	17.56%
36	4.58	5673×93	30	339.39	129	25	353.39	3.96%	273.19	22.69%
35	4.66	5210×91	48	248.68	128	45	272.69	8.81%	216.96	20.44%
36	4.69	5985×93	27	258.24	133	21	274.24	5.83%	213.19	22.26%

Table 2. Results for $|S| = 30$ nodes

NV	ND	LP	C_1	LB_1	CY_2	C_2	LB_2	s	LB_c	r
53	4.75	8967×137	71	617.15	199	65	656.41	5.98%	498.03	24.13%
54	4.69	9180×139	156	574.54	199	102	585.29	1.84%	534.32	8.71%
54	4.76	9901×139	131	466.45	203	155	482.33	3.29%	415.91	13.77%
52	4.73	8202×135	108	510.74	194	76	538.85	5.22%	447.84	16.89%
55	4.73	10267×141	150	661.08	205	170	669.21	1.21%	635.09	5.10%
55	4.71	9832×141	65	551.48	204	89	575.51	4.17%	469.33	18.45%
54	4.63	8692×139	182	621.06	196	125	624.77	0.59%	580.45	7.09%
54	4.70	9385×139	83	501.09	200	70	521.93	3.99%	453.02	13.20%
54	4.72	9708×139	139	666.06	201	108	693.95	4.02%	567.70	18.19%
51	4.63	6872×133	84	701.67	185	89	728.14	3.64%	614.10	15.66%
55	4.62	9495×141	120	752.48	199	114	789.08	4.64%	652.06	17.36%
52	4.81	8499×135	41	362.03	198	59	373.93	3.18%	343.23	8.21%
54	4.83	10339×139	147	550.30	207	105	568.11	3.13%	494.89	12.89%
53	4.53	7334×137	104	483.64	187	121	492.93	1.88%	440.26	10.69%
56	4.66	10459×143	119	585.74	205	152	600.23	2.41%	520.49	13.28%
54	4.61	8377×139	122	386.69	195	185	395.35	2.19%	362.76	8.24%
54	4.61	8430×139	130	564.20	195	160	586.59	3.82%	490.53	16.38%
52	4.71	7639×135	94	601.48	193	101	622.92	3.44%	548.50	11.95%
52	4.65	7781×135	122	730.86	190	181	748.63	2.37%	690.08	7.82%
54	4.63	8788×139	123	549.76	196	125	561.70	2.13%	505.84	9.94%
52	4.77	8593×135	53	499.65	196	80	540.15	7.50%	417.11	22.78%
53	4.75	9236×137	120	594.47	199	101	609.37	2.45%	546.52	10.31%
55	4.80	11340×141	206	639.19	209	220	682.14	6.30	586.58	14.01%
55	4.80	10556×141	88	697.76	209	113	714.69	2.37%	636.08	11.00%
55	4.67	10051×141	107	658.24	202	83	689.52	4.54%	601.87	12.71%
54	4.69	9337×139	105	436.55	199	89	460.05	5.11%	366.10	20.42%
54	4.74	9866×139	177	667.81	202	107	689.80	3.19%	626.04	9.24%
56	4.63	9660×143	88	389.49	203	60	403.96	3.58%	336.88	16.61%
54	4.72	9962×139	104	626.79	201	102	659.54	4.97%	512.52	22.29%
53	4.77	8884×137	111	749.57	200	132	759.81	1.35%	715.61	5.82%

Table 3. Results for $|S| = 40$ nodes

NV	ND	LP	C_1	LB_1	CY_2	C_2	LB_2	s	LB_c	r
71	4.82	13187×183	123	789.97	138	327	823.37	4.06%	719.73	12.59%
74	4.76	14495×189	149	815.39	131	297	836.82	2.56%	744.51	11.03%
72	4.72	12995×185	251	724.03	128	343	753.90	3.96%	600.96	20.29%
71	4.66	11965×183	157	640.54	126	272	668.52	4.18%	569.87	14.76%
71	4.72	12024×183	171	927.73	137	344	940.24	1.33%	905.66	3.68%
72	4.71	13188×185	238	640.23	131	342	648.49	1.27%	621.26	4.20%
70	4.54	9369×181	150	723.92	144	217	758.85	4.60%	596.23	21.43%
72	4.67	12764×185	205	738.69	129	403	757.31	2.46%	687.84	9.17%
72	4.75	14874×185	335	1120.24	125	478	1146,78	2,33%	1013.85	10.04%
73	4.74	13910×187	194	635.01	130	366	648,42	2,07%	584.09	9.92%
71	4.72	11609×183	188	686.29	132	317	710.09	3.46%	606.22	14.73%
73	4.79	14046×187	260	1002.16	134	409	1038.00	3.45%	853.43	17.78%
71	4.62	11286×183	205	953.81	131	294	976.32	2.31%	907.29	7.07%
74	4.69	13253×189	213	867.31	137	511	986.18	1.91%	848.54	13.96%
73	4.67	12853×187	239	837.78	131	394	863.88	3.02%	777.79	9.97%
72	4.63	12377×185	122	817.58	129	261	855.29	4.41%	734.96	14.07%
70	4.73	11301×181	263	859.04	126	381	888.53	3.32%	808.34	9.03%
70	4.69	10925×181	112	594.88	129	238	608.89	2.32%	551.91	9.37%
74	4.66	12880×189	274	925.32	133	326	968.12	4.42%	831.04	14.16%
72	4.76	13415×185	103	417.55	135	225	438.88	4.86%	364.63	16.92%
72	4.75	13593×185	228	641.72	132	358	664.84	3.48%	580.03	12.76%
72	4.75	13177×185	280	766.66	123	351	777.82	1.43%	716.60	7.87%
71	4.69	11825×183	89	689.59	129	239	728.70	5.37%	571.29	21.60%
71	4.65	11340×183	211	763.79	127	353	775.97	1.57%	709.39	8.58%
71	4.68	11670×183	256	671.51	135	376	673.89	0.35%	664.25	1,43%
72	4.75	12646×185	222	938.52	134	359	952.18	1.44%	880.50	7.53%
72	4.63	11407×185	241	798.91	129	361	817.79	2.31%	767.25	6.18%
74	4.72	13899×189	176	565.95	128	323	578.90	2.24%	501.14	13.43%
72	4.79	13854×185	353	694.55	133	336	710.08	2.19%	661.74	6.81%

Table 4. Results for $|S| = 50$ nodes

NV	ND	LP	C_1	LB_1	CY_2	C_2	LB_2	s	LB_c	r
91	4.81	18662×233	313	863.94	347	239	895.69	3.54%	805.5899	10.06%
93	4.74	19030×237	299	1098.09	348	311	1116.49	1.65%	1045.529	6.36%
93	4.73	17991×237	362	838.90	347	452	856.81	2.09%	804.2123	6.14%
90	4.72	15974×231	500	1166.91	335	500	1226.50	4.86%	1067.224	12.99%
94	4.74	19154×239	342	1383.86	352	370	1429.16	3.17%	1292.874	9.54%
91	4.68	16096×233	293	988.16	335	250	1012.74	2.43%	907.3354	10.41%
94	4.72	19028×239	500	1218.22	350	392	1229.36	0.91%	1174.971	4.42%
89	4.72	16786×229	195	1009.29	331	288	1112.80	9.30%	796.5293	28.42%
93	4.74	18824×237	452	1326.19	348	500	1354.22	2.07%	1268.811	6.31%
91	4.69	16698×233	299	1223.65	336	301	1259.19	2.82%	1136.339	9.76%
90	4.77	16485×231	500	1137.16	339	462	1178.08	3.47%	1061.931	9.86%
93	4.70	17677×237	500	1114.51	344	500	1119.08	0.41%	1109.636	0.84%
92	4.73	17416×235	457	847.59	343	391	874.62	3.09%	795.2771	9.07%
89	4.61	13996×229	316	1029.88	321	264	1057.44	2.61%	946.4389	10.50%
89	4.71	15045×229	334	875.36	330	312	889.87	1.63%	834.1946	6.26%
89	4.72	15572×229	418	1166.70	331	350	1176.89	0.87%	1087.305	7.61%
91	4.58	15421×233	362	1260.22	326	345	1271.77	0.91%	1217.655	4.26%
90	4.76	17083×231	429	1208.29	338	337	1228.44	1.64%	1141.489	7.08%
89	4.76	15869×229	297	882.59	335	207	897.11	1.62%	828.9133	7.60%
90	4.78	17272×231	273	983.71	340	377	1008.66	2.47%	940.5136	6.76%
94	4.66	18263×239	478	1020.79	344	418	1042.22	2.06%	948.0095	9.04%
92	4.73	17211×235	332	1073.07	343	430	1108.37	3.19%	987.9522	10.86%
93	4.67	17187×237	226	820.62	341	289	850.43	3.51%	738.4644	13.17%
92	4.64	16531×235	197	1173.22	335	218	1198.11	2.08%	1087.289	9.25%
92	4.66	16291×235	327	912.51	337	265	950.87	4.03%	855.6042	10.02%
92	4.77	17673×235	465	1398.40	347	496	1432.21	2.36%	1316.483	8.08%

The computational results in Tables 1-4 show that the difference between the lower bounds obtained by solving (R_2) and those obtained by solving (R_1) lie in the range $[0, 35\%, 9.30\%]$ with an average of 3.3%. In all the examples treated, the number of constraints (9') generated for building relaxation (R_2) is rather limited, ranging from an average of 50 for 20 node networks to an average of 350 for 40 node networks. (Note that this is significantly less than the total number of constraints necessary in a node-arc formulation of the feasible multicommodity flow problem).

Concerning the difference between the best relaxation (R_2) and the relaxation using convexification, we observe that (R_2) is always better: in 90% of the test examples treated the improvement over the optimal solution to the convexified problem lies in the range 5-25% (the average improvement is equal to 13.3%).

These results confirm the relevance of the LP model studied here to generate relaxations significantly better than convexification.

Acknowledgements

We thank Professor E. Boros (RUTCOR/Rutgers University) for fruitful discussions concerning relaxation (R_2).

References

1. R. K. Ahuja, T. Magnanti and J. Orlin, Network Flows: Theory, Algorithms and Application, Prentice Hall 1993.
2. A. A. Assad, *Multicommodity network flows - A survey*, Networks **8** (1978), 37-91.
3. L. Fratta, M. Gerla and L. Kleinrock, *The flow-deviation method: an approach to store-and-forward communication network design*, Network **3** (1973), 97-133.
4. M. Gondran and M. Minoux, *Graphes et Algorithmes*, Eyrolles, Paris 1979 (3ème édition) 1995.
5. J. L. Kennington, *A survey of linear cost multicommodity network flows*, Operations Research **26** (1978), 209-236.
6. M. Minoux (1983), Programmation Mathématique: Théorie et Algorithmes, Dunod, Paris, English translation, John Wiley & Sons, New York, 1986.
7. M. Minoux, *Network synthesis and optimum network design problems: Models, solution methods and applications*, Networks **19** (1989), 313-360.
8. A. Ouorou (1985), *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
Solving $(P_2)'$ by a column generation procedure

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 to solving the following subproblem

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

where $\pi = \big(\pi_u\big)_{u\in\omega(i)}$ 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.

For solving subproblem (Q), let us first consider the following set of problems

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

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 $w^* = \min\limits_{E\geq r_i} \big\{w^*_{\delta(i)}(E)\big\}$, 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

$$(Q_j(E)) : w^*_j(E) = \min_{x_j\in V_j} \Big\{(\Phi_j(x_j) - \pi_j x_j) + w^*_{j-1}(E - x_j)\Big\},$$

where it should be understood that $w^*_j(E) = +\infty$, 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 values (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. 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 to (Q). The procedure called **solution** simply outputs the best solution to (Q) by using the $\sigma_j(E)$ pointers.

$w_0(0) = 0$
$w_0(E) = +\infty \quad \forall E = 1, \ldots, r_i$
$\sigma_j(E) = \emptyset \quad \forall E = 0, 1, \ldots, r_i, \ \forall j = 0, 1, \ldots, \delta(i)$
`for` $j = 1$ `to` $\delta(i)$
    $w_j(E) = +\infty \quad \forall E = 0, 1, \ldots, v_j^{q(j)}$
    `for` $x_j = v_j^0$ `to` $v_j^{q(j)}$
        `forall` E `such that` $w_{j-1}(E) \neq +\infty$
            `if` $(w_j(E + x_j) > \Phi_j(x_j) - \pi_j x_j + w_{j-1}(E))$
                $w_j(E + x_j) \leftarrow \Phi_j(x_j) - \pi_j x_j + w_{j-1}(E))$
                $\sigma_j(E + x_j) \leftarrow \{(u, x_j, \sigma_{j-1}(E))\}$
            `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') ← +∞
            σ_j(E') ← ∅
        end if
    end forall
end forall
```

Figure 4. The procedure efficiency (j)

Appendix 2
Randomly generating realistic test problems

The generation of the test networks used in the experiments of Section 5 has been performed according to the following procedure:

1) Choose the number of nodes m; thus, the node set will be $S = \{1, 2, \ldots, m\}$.

2) Draw at random (according to a uniform distribution on $[0, 1000]$) the coordinates of m points in the euclidian square $[0, 1000] \times [0, 1000]$. The points are generated iteratively as follows. Suppose $k-1$ points have been generated so far. A candidate for being the k^{th} point is obtained by drawing its two coordinates in $[0, 1000]$. This point is accepted if its Euclidian distance to the closest point i ($i \in \{1, 2, \ldots, k-1\}$) is larger than a given threshold value $\theta(m)$ (in our experiments $\theta(m) = \dfrac{1000}{5\sqrt{m}}$). If this condition is not fulfilled, a new candidate is drawn at random. At the end of this step, we have the m nodes $k = 1, \ldots, m$ of the network given by their integer coordinates $\begin{pmatrix} a_k \\ b_k \end{pmatrix}$ in the square $[0, 1000] \times [0, 1000]$.

3) Build the edges of a graph $\overline{\overline{G}}$ on this set of nodes as follows. For each node k, and for any angle $\alpha \in \{\alpha_0, \alpha_1, \ldots, \alpha_7\}$ with $\alpha_p = p \times \frac{\pi}{4}$, denote $S_k(\alpha_p)$ the subset of nodes $j \in \{1, \ldots, m\} \setminus \{k\}$ such that

$$\frac{(a_j - a_k) + i(b_j - b_k)}{\sqrt{(a_j - a_k)^2 + (b_j - b_k)^2}} = e^{i\sigma} \text{ with } \alpha_p \leq \sigma < \alpha_p + \frac{\pi}{4} \cdot$$

For each $p \in [0,7]$ such that $S_k(\alpha_p) \neq \emptyset$, let $j^*(p)$ be the point in $S_k(\alpha_p)$ closest to k (in terms of Euclidian distance), take $(k, j^*(p))$ in the edge set $\overline{\overline{\mathcal{U}}}$ of $\overline{\overline{G}}$. At the end of this step we have a graph $\overline{\overline{G}} = \big[S, \overline{\overline{\mathcal{U}}}\big]$.

4) Define the final graph $G = [S, \mathcal{U}]$ as a partial graph of $\overline{\overline{G}} = \big[S, \overline{\overline{\mathcal{U}}}\big]$ by iteratively removing edges from $\overline{\overline{\mathcal{U}}}$ in order to obtain all node degrees in the range $[d_{\min}, d_{\max}]$ (where $d_{\min}$ and $d_{\max}$ are given parameters, $d_{\min} = 3$ and $d_{\max} = 4$ in our experiments). One iteration of the process is as follows. Let $\overline{G} = \big[S, \overline{\mathcal{U}}\big]$ be the current graph and let $\overline{d}$ be the maximum degree in $\overline{G}$

$$\overline{d} = \max_{i=1,\ldots,m} \Big\{ d_{\overline{G}}(i) \Big\}.$$

If $\overline{d} \leq d_{\max}$ terminate; otherwise let $W \subset \overline{\mathcal{U}} \times \overline{\mathcal{U}}$ the set of pairs of edges (u, v) with $u = (i, j)$ and $v = (i, k)$ such that

$$\big(d_{\overline{G}}(i) = \overline{d}\big) \wedge \big(\big(d_{\overline{G}}(j) > d_{\min}\big) \vee \big(d_{\overline{G}}(k) > d_{\min}\big)\big). \tag{12}$$

Determine in W the pair such that the angle in the euclidian plane between the vectors $\begin{pmatrix} a_j - a_i \\ b_j - b_i \end{pmatrix}$ and $\begin{pmatrix} a_k - a_i \\ b_k - b_i \end{pmatrix}$ is smallest. Delete the longest of the two edges u and v provided that the degree in $\overline{G}$ of both its endpoints exceeds $d_{\min}$. Otherwise, delete the other edge (note that due to (12) at least one of the two edges may be deleted).

In all our experiments, the networks generated according to the above construction were connected.

On the obtained network $G = \big[S, \mathcal{U}\big]$, the multicommodity flow requirements between each node pair are defined as follows:

- draw at random a weight p_i for each node i in the range $[1, 20]$,

- compute the amount of the k^{th} flow to be routed between each node pair $(s(k), t(k)) \in X \times X$ according to the following formula

$$d_k = \left[\frac{\mu \times p_{s(k)} \times p_{t(k)}}{\mathrm{dist}(s(k), t(k))} \right],$$

where $\mathrm{dist}(i, j)$ is the Euclidian distance between nodes i and j, and μ is a scaling factor chosen in order to make the largest flow value equal to 100.

Having obtained a graph instance $G = [S, \mathcal{U}]$, the cost functions Φ_u on the edges have to be generated.

To computer 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 chain between $s(k)$ and $t(k)$ (in terms of euclidian distance). Then $\forall u \in \mathcal{U}$, β_u is taken to be equal to three times the flows carried by edge u in this solution.

We now define for edge $u \in \mathcal{U}$ the associated cost function $\Phi_u(x_u)$ in the following way:

1) choose the number $q(u)$ of discontinuity points (in our computational experiments $q(u)$ has been chosen equal to 5 for all $u \in \mathcal{U}$),

2) draw at random $q(u)-1$ discontinuity points v_u^i ($i = 1, \ldots, q(u)-1$) in the range $[1, \beta_u[$, and impose $v_u^0 = 0$ and $v_u^{q(u)} = \beta_u$,

3) computer the cost γ_u^i associated with each discontinuty point v_u^i ($i = 0, \ldots, q(u)$) according to the formula

$$\gamma_u^i = \Phi_u(v_u^i) = a \times \left(v_u^i\right)^\alpha \times \text{dist}(u),$$

where α is a coefficient equal to 0.6, a is a random number chosen in the range $[0.9, 1.1]$ and dist(u) is the Euclidian distance between the two endpoints of link u. However when we compute γ_u^i ($i = 1, \ldots, q(u) - 1$), if it appears that $\gamma_u^{i-1} \geq \gamma_u^i$, we delete the discontinuity point v_u^i of the function $\Phi_u(x_u)$.

LIPN, Université Paris 13, Institut Galilée,
Avenue J-B. Clément, 93430 Villetaneuse, France,
E-mail address: gabrel@ura1507.univ-paris13.fr

MASI, Université Paris 6,
4 place Jussieu, 75252 Paris Cedex 05, France
E-mail address: minoux@masi.ibp.fr