ELSEVIER

Operations Research Letters 25 (1999) 15–23

operations research letters

www.elsevier.com/locate/orms

Exact solution of multicommodity network optimization problems with general step cost functions

V. Gabrel[a], A. Knippel[b], M. Minoux[b,*]

[a] *Université Paris 13 LIPN-Avenue J.-B. Clément, 93430 Villetaneuse, France*
[b] *Laboratoire d' Informatique de Paris 6, Université Paris 6-LIP6-4 Place Jussieu, 75005 Paris, France*

Received 1 August 1998; received in revised form 1 January 1999

Abstract

We describe an exact solution procedure, based on the use of standard LP software, for multicommodity network optimization problems with general discontinuous step-increasing cost functions. This class of problems includes the so-called single-facility and multiple-facility capacitated network loading problems as special cases. The proposed procedure may be viewed as a specialization of the well-known BENDERS partitioning procedure, leading to iteratively solving an integer 0–1 linear programming relaxed subproblem which is progressively augmented through constraint generation. We propose an improved implementation of the constraint generation principle where, at each step, several ($O(N)$) new constraints are included into the current problem, thanks to which the total number of iterations is greatly reduced (never exceeding 15 in all the test problems treated). We report on systematic computational experiments for networks up to 20 nodes, 37 links and cost functions with an average six steps per link.

Keywords: Optimum network design; Multicommodity flows; Benders method; Multiple constraint generation

1. Introduction

The minimum cost multicommodity network flow problem with discontinuous step increasing cost functions is a basic model in Telecommunication network design. Previous work on the subject has been carried out on the following special cases:

(a) The uncapacitated network design problems [15] where, on each link of the network, it is only possible either to open the link (in which case infinite capacity is available at some given fixed cost attached to the link) or not (in which case the cost is zero and no capacity is available).

(b) The so-called single facility capacitated network loading problem, where capacity expansion on any given link u can be done by installing an integer number of units of a given basic facility characterized by its capacity C and its cost γ_u (see [13,3]).

(c) The so-called two-facility capacitated network loading problem which generalizes the previous model in that, on each link u, capacity expansion can be achieved by means of two types of facilities, one with capacity C^1 and cost γ_u^1 and the other with capacity C^2 and cost γ_u^2 (see [14]).

* Corresponding author. Fax: 33-1-44-27-62-86.
E-mail address: michel.minoux@lip6.fr (M. Minoux)

0167-6377/99/$ - see front matter
PII: S0167-6377(99)00020-6

We address here the more general class of problems where, on each link u, an arbitrary discontinuous step-increasing cost function is given.

This general model has received, up to now, only very little attention. Stoer and Dahl [19] is one of the only references we are aware of considering general step cost functions. They propose a cutting plane approach using polyhedral properties (valid inequalities and facet-defining inequalities) but the computational experiments reported there are limited to a single network structure with 27 nodes and 51 links with a very sparse requirement matrix (composed of only 19 individual requirements). Exact optimal solutions are only reported for a very special set of values of the input data.

Here we do not assume sparsity in the requirement matrix, i.e., for an n node instance, the multicommodity flow requirements to be satisfied include one requirement for each pair of nodes (thus the number of commodities is $n(n-1)/2$).

LP relaxations have also been investigated in Gabrel and Minoux [6] leading to lower bounds improving the natural bounds derived from the convexified problem. Computational results are provided for instances up to 50 nodes and about 90 links.

In the present paper, it is shown that, with an appropriate implementation of the constraint generation approach (a specialization of the well-known BENDERS procedure) standard LP software (such as CPLEX) can be used to obtain exact optimal solutions up to about 20 nodes and 37 links. As far as we know, this is the first systematic computational study aimed at solving exactly this class of hard network optimization problems.

The paper is organized as follows. The multicommodity network optimization problem with general step cost functions is formulated in Section 2. Constraint generation procedures, together with details on their implementations, are provided in Sections 3 and 4. Computational results are presented and discussed in Section 5.

2. Problem formulation

The basic network structure is given as an undirected graph $G=[\mathcal{N},\mathcal{U}]$ where $\mathcal{N}$ is the set of nodes ($|\mathcal{N}|=N$) and $\mathcal{U}$ the set of the edges (links) ($|\mathcal{U}|=M$).

The problem to be considered is to decide the amount of capacity $x_u \geqslant 0$ 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 given upper bound constraints:

$$\forall u \in \mathcal{U}\colon\ 0 \leqslant x_u \leqslant \beta_u;$$

- minimize the total cost of the network which, in terms of given individual link cost functions $\phi_u(x_u)$ $(u=1,\ldots,M)$, 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 e.g. [11,1]), linear with fixed cost or nonlinear concave but continuous and differentiable (see e.g. [18]).

We address here the minimum cost multicommodity flow problem in the case of general discontinuous step-increasing cost functions.

Thus, for each edge $u \in \mathcal{U}$ in the network, we assume that we are given a cost function $\phi_u(x_u)$ defined as follows. Let $V_u = \{v_u^0, v_u^1, \ldots, v_u^{q(u)}\}$ be a finite set of values representing the discontinuity points of the ϕ_u function and denote

$$\gamma_u^0 = \phi_u(v_u^0), \quad \gamma_u^1 = \phi_u(v_u^1),$$

$$\gamma_u^2 = \phi_u(v_u^2), \ \ldots\ \ldots\ \gamma_u^{q(u)} = \phi_u(v_u^{q(u)}),$$

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

With this notation we have

$$\phi_u(x_u) = 0 \quad \text{if } x_u = 0 \text{ and, } \forall i = 1,\ldots,q(u)\colon$$

$$\phi_u(x_u) = \gamma_u^i \quad \text{for all } x_u \in\,]v_u^{i-1}, v_u^i].$$

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: $0 \leqslant x_u \leqslant \beta_u$ either explicitly or implicitly.

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 kth flow to be routed between $s(k)$ and

$t(k)$), we denote by $X \subset \mathbf{R}_+^M$ the polyhedron representing the set of all feasible multicommodity flows. Thus $x = (x_u)_{u\in\mathscr{U}} \geqslant 0$ 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. With this notation, the minimum cost multicommodity flow problem to be solved may be formulated as

$$(\mathrm{P}) \begin{cases} \min & \displaystyle\sum_{u\in\mathscr{U}} \phi_u(x_u) \\ \text{s.t.} & x \in X \\ & x_u \in V_u \ (\forall u \in \mathscr{U}). \end{cases}$$

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 [11,18]). Later in the paper we will use the following representation of X involving the x variables only. For any $\lambda = (\lambda_1, \ldots, \lambda_M) \in \mathbf{R}_+^M$, 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 $u \in \mathscr{U}$ is given length $\lambda_u \geqslant 0$.

Then $x = (x_u)_{u\in\mathscr{U}}$ belongs to X if and only if, for all $\lambda \in \mathbf{R}_+^M$, we have

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

(see e.g. [8, Chapter 6]).

Constraints (1) are sometimes referred to as "metric inequalities" (see [2]). Note that testing whether a given $\bar{x} \in \mathbf{R}^M$ belongs to X can be done in polynomial time, since this amounts to solving a linear program.

3. Solving (P) through constraint generation (Benders)

The description of the multicommodity flow polyhedron X as a large set of metric inequalities of type (1) suggests a constraint generation approach, starting from an initial relaxation which is progressively refined by adding new inequalities violated by the current solution. The process stops when the (exact) optimal solution $\bar{x} = (\bar{x}_u)_{u\in\mathscr{U}}$ to the current relaxed problem satisfies all the metric inequalities, i.e. when $\bar{x} \in X$. With respect to problem (P), such a procedure may be viewed as a specialization of the well-known Benders approach [4] which has long been recognized as a useful basic tool for solving other types of optimum network design problems (see e.g. [7,16,10,17]).

At the current iteration k of the constraint generation approach, let J^k be the index set of metric inequalities generated so far. Solve (exactly) the current relaxed subproblem:

$$(\mathrm{R}_k) \begin{cases} \min & \displaystyle\sum_{u\in\mathscr{U}} \phi_u(x_u) \\ \text{s.t.} & \displaystyle\sum_{u\in\mathscr{U}} \lambda_u^j x_u \geqslant \theta(\lambda^j) \quad \forall j \in J^k. \\ & x_u \in V_u, \quad \forall u \in \mathscr{U}. \end{cases}$$

Let $\bar{x} = (\bar{x}_u)$ be the exact optimal solution obtained.

Metric inequalities violated by the current $\bar{x}$ are then looked for. If one (or several) can be found, add it (add them) to (R_k) to form the augmented relaxed subproblem (R_{k+1}), and start a new iteration $k+1$. If no violated inequality can be found then terminate: $\bar{x} \in X$ and $\bar{x}$ is an optimal solution to (P).

To implement the above, the current relaxed subproblem (R_k) is reformulated as a pure 0–1 integer linear program by introducing, for each link $u, q(u)$ 0–1 variables $\mu_u^1, \mu_u^2, \ldots, \mu_u^{q(u)}$ satisfying:

$$\forall t = 2, \ldots, q(u): \ \mu_u^t \leqslant \mu_u^{t-1}$$

and expressing the x_u variables as

$$\forall u \in \mathscr{U}: \ x_u = \sum_{t=1}^{q(u)} \mu_u^t (v_u^t - v_u^{t-1}) \tag{2}$$

and the objective function as

$$z = \sum_{u\in\mathscr{U}} \sum_{t=1}^{q(u)} \mu_u^t (\gamma_u^t - \gamma_u^{t-1}). \tag{3}$$

Thus, (R_k) reduces to the following 0–1 integer linear programming problem (ILP_k):

$$(ILP_k)\begin{cases}\min & z=\sum_{u\in\mathscr{U}}\sum_{t=1}^{q(u)}\mu_u^t(\gamma_u^t-\gamma_u^{t-1})\\ \text{s.t.} & \sum_{u\in\mathscr{U}}\lambda_u^j\left(\sum_{t=1}^{q(u)}\mu_u^t(v_u^t-v_u^{t-1})\right)\geqslant\theta(\lambda_j)\\ & \forall j\in J^k\\ & \forall u\in\mathscr{U},\ \forall t=2,\dots,q(u):\ \mu_u^t\leqslant\mu_u^{t-1}\\ & \forall t=1,\dots q(u):\ \mu_u^t\in\{0,1\}.\end{cases}$$

(ILP_k) could be solved either by using some of the various available standard LP software, either by developing a specialized algorithm. In this paper we chose to investigate the capabilities of standard LP software and we used CPLEX 4.0 in MIP mode to solve the relaxed problems (R_k) in all our computational experiments (see Section 5).

We now discuss the implementation of the constraint generation process at each iteration.

4. Single and multiple constraint generation procedures

To implement constraint generation, we first tried the standard way consisting in generating, at each iteration, a single "most violated" metric inequality, as explained in Section 4.1 below.

4.1. Single constraint generation (SCG)

Let $\bar{x}$ denote the (exact) optimal solution to the current relaxed subproblem (R_k). There are many possible criteria for selecting a "most violated inequality".

Based on some preliminary computational testing, the criterion chosen was to select a metric inequality maximizing the ratio between the right-hand side and left-hand side. It is easily seen that such an inequality is obtained as an optimal solution to the following auxiliary problem

$$(AP)\begin{cases}\max & \theta(\lambda)\\ \text{s.t.} & \sum\lambda_u\bar{x}_u=1\\ & \lambda_u\geqslant 0\quad\forall u\in\mathscr{U}.\end{cases}$$

In our experiments we solved (AP) using the subgradient algorithm described in [6, Section 4.1]. With this subgradient algorithm we can only approximate the exact optimal solution to (AP), but our experiments have confirmed that exact optimality in (AP) is not needed in intermediate steps, good approximate solutions to (AP) are sufficient. However, whenever our subgradient algorithm fails to produce a λ with $\theta(\lambda)>1$ (therefore suggesting that $\bar{x}\in X$ is likely to occur) an *exact feasibility test* is carried out by solving a continuous feasible multicommodity flow problem (an ordinary continuous LP problem easily solved by standard LP software).

The computational experiments reported in Table 1 show that, even for very small-sized problems ($\approx$ 8–12 nodes), constraint generation with (SCG) not only requires a significant number of iterations, but this number seems to increase quite rapidly with problem size (average # of iterations is 13 for $N=8$; 18 for $N=10$ and 37 for $N=12$).

In order to improve the efficiency of the algorithm (i.e. to reduce the total number of main iterations), we investigated a different approach where several violated inequalities are systematically generated at each iteration.

4.2. Multiple constraint generation (MCG)

The MCG procedure described here is based on two main ideas:

(i) before considering general metric inequalities, *bipartition inequalities* (i.e. metric inequalities corresponding to bipartitions of the node set X) are generated first;

(ii) at each step a significant number (O(N) in our experiments) of candidate bipartition inequalities is computed, all the violated inequalities in this set being actually added to the current relaxed subproblem.

For any subset $S\subset\mathscr{N}$, we denote $\bar{S}=\mathscr{N}\setminus S$, $\omega(S)$ the subset of edges having one endpoint in S, and the other in $\bar{S}$, and $d(S,\bar{S})$ the total sum of requirements d_k such that either $s(k)\in S$ and $t(k)\in\bar{S}$ or $s(k)\in\bar{S}$ and $t(k)\in S$.

The bipartition inequality induced by S is a metric inequality, which reads:

$$\sum_{u\in\omega(S)}\bar{x}_u\geqslant d(S,\bar{S}).\tag{4}$$

For a given $\bar{x}$, finding a most violated bipartition inequality can be done according to various possible criteria. Several such criteria were tested, among which:

- maximizing the difference between the right- and left-hand sides in (4);
- maximizing the ratio

$$\rho(S) = \frac{d(S, \bar{S})}{\sum_{u \in \omega(S)} \bar{x}_u}$$

between the right- and left-hand sides in (4).

Based on preliminary computational experiments, the second criterion was found to be the best choice for (MCG). With this criterion, the problem is to determine $S^* \subseteq \mathcal{N}$ such that

$$\rho(S^*) = \max_{S \subseteq \mathcal{N}} \{\rho(S)\}. \tag{5}$$

Since (5) is an NP-hard problem (MAX-CUT is easily seen to be a special case), (5) will be solved only approximately via a variable-depth local search heuristic of Kernighan-Lin type [12]. This is implemented through the procedure MAX-RATIO-CUT (i_0, j_0) below, which, for any given pair of nodes (i_0, j_0) in $\mathcal{N}$, returns a near-optimal subset S such that $i_0 \in S$, $j_0 \in \bar{S}$.

Procedure MAX-RATIO-CUT (i_0, j_0)

(a) initialization.

Randomly choose $S \subset \mathcal{N}$ satisfying $i_0 \in S,\ j_0 \notin S$. Set $S^* \leftarrow S,\ t \leftarrow 1$.

(b) Current phase t.

Set:

$\hat{\rho} \leftarrow \rho(S),\ \hat{S} \leftarrow S,\ T \leftarrow \{i_0, j_0\}$

While $(T \neq \mathcal{N})$ do

For each $i \in \mathcal{N} \setminus T$ compute:

$\delta_i = \rho\ (S \cup \{i\}) - \rho(S)$ if $i \notin S$

$\delta_i = \rho(S \setminus \{i\}) - \rho(S)$ if $i \in S$

and determine $\delta_r = \max_{i \in \mathcal{N} \setminus T} \{\delta_i\}$

$T \leftarrow T \cup \{r\}$

if $r \notin S$ set $S \leftarrow S \cup \{r\}$

if $r \in S$ set $S \leftarrow S \setminus \{r\}$

if $\rho\ (S) > \hat{\rho}$ set: $\hat{\rho} \leftarrow \rho\ (S)$, $\hat{S} \leftarrow S$

endWhile

If $\rho\ (\hat{S}) \leqslant \rho(S^*)$ Terminate and output S^*.

Otherwise set $S^* \leftarrow \hat{S},\ S \leftarrow \hat{S},\ t \leftarrow t + 1$ and return to (b)

In our experiments, each time the procedure MAX-RATIO-CUT is called for, ten distinct random initial subsets S are tried, and the final result is taken to be the best of the 10 locally optimal solutions found. The observed number of phases needed for reaching a local optimum from a given initial subset typically lies between 2 and 4, and very rarely exceeds 4.

We now describe the MCG procedure. The first step of this procedure consists in calling MAX-RATIO-CUT (i, j) for all $(i, j) \in \mathcal{U}$. This is done in order to ensure that each variable x_u is involved in at least one of the candidate bipartition inequalities. Of course only those candidate inequalities which are violated by the current $\bar{x}$ are actually appended to (R_k).

In practice, it was observed that the number of distinct cuts found at each step is usually close to $N - 1$ (note that this is consistent with the result due to Cheng and Hu [5]).

(MCG) multiple constraint generation procedure

Input: $\bar{x} = (\bar{x}_u)_{u \in \mathcal{U}}$

Step 1: For all $u = (i, j) \in \mathcal{U}$ do: call MAX-RATIO-CUT (i, j), and let S^u be the (near-optimal) subset returned by the procedure.

If $\rho(S^u) \leqslant 1$ for all $u \in \mathcal{U}$, go to step 2.

Otherwise, for each u such that $\rho(S^u) > 1$, add to the current relaxed subproblem the new constraint:

$$\sum_{v \in \omega(S^u)} x_v \geqslant d(S^u, \bar{S}^u),$$

end of (MCG).

Step 2: It essentially consists in applying (SCG) as described in Section 4.1. Determine $\bar{\lambda}$, an (approximate) optimal solution to the auxiliary problem (AP) as explained in Section 4.1 to find a most violated metric inequality.

If $\theta\ (\bar{\lambda}) > 1$ add to the current relaxed subproblem the metric inequality:

$$\sum_{u \in \mathcal{U}} \bar{\lambda}_u x_u \geqslant \theta(\bar{\lambda})$$

end of (MCG).

If $\theta\ (\bar{\lambda}) \leqslant 1$ no violated metric inequality has been found. End of (MCG).

In all our computational experiments, it was observed that when (MCG) terminates without producing

any new violated inequality, then the current $\bar{x}$ was indeed feasible. Whenever this situation occurs, an *exact feasibility check* is carried out by solving to optimality (using CPLEX) the (continuous) linear program obtained from a node-arc formulation of the feasible multicommodity flow problem (where each link u is assigned capacity $\bar{x}_u$).

To initialize the constraint generation process, (MCG) is applied with an input vector $\bar{x}$ constructed as follows. For all $k \in [1, K]$, let P_k denote the edge set of the shortest chain between $s(k)$ and $t(k)$ in terms of number of edges, and let $\psi_u = \sum_{k|u\in P_k} d_k$ (thus ψ_u is recognized as the total flow through link u, when all the commodities are routed on a single chain having minimum number of edges between source and sink). Then, the initial $\bar{x}$ vector used is defined by

$$\forall u \in \mathscr{U}: \bar{x}_u = \tfrac{1}{2}\psi_u. \tag{6}$$

The initial restricted problem (R_1), at the first iteration of the constraint generation process, is therefore composed of all the violated bipartition inequalities identified by (MCG) with the $\bar{x}$ vector defined by (6).

5. Computational results

We present two series of results, shown in Tables 1 and 2.

Table 1 compares the two implementations of the constraint generation process obtained by using either the (SCG) procedure described in Section 4.1 or the (MCG) procedure in Section 4.2. In order to obtain a fair comparison between the two approaches, the initial restricted problem for (SCG) has been taken to be the same as for (MCG) (for details refer to Section 4.2 above).

For each method Table 1 displays:

- the total number of iterations needed until exact optimality is reached (for (SCG), since exactly one metric inequality is generated at each iteration, this is also the total number of generated constraints);
- the total running time (in seconds) of the procedure on a SPARC 20 workstation;

The last column shows the factor of improvement in terms of computing time between (SCG) and (MCG).

Due to the long time taken by (SCG) these experiments have been limited to small size problems not exceeding 12 nodes.

The results from Table 1 clearly confirm the superiority of (MCG) over (SCG), both in terms of number of iterations and computing time.

The second series of results, found in Table 2, illustrates the behaviour of the (MCG) procedure on a full set of 50 test problems of size up to 20 nodes and 37 links. The corresponding data have been obtained by applying the random generator described in Appendix 2 of Gabrel and Minoux [6]. All the requirement matrices are fully dense. Also the link cost functions feature an average number of six steps. For each problem, Table 2 shows:

- the number of nodes N and the number of links M;
- NV, the number of 0–1 variables in the relaxed subproblem;
- NC_1, the number of constraints in the initial relaxed subproblem (R_1);
- z_1, the optimal integer solution value of the initial relaxed subproblem;
- iter, the total number of iterations necessary to reach an exact optimal solution to (P);
- NAP the total number of metric inequalities generated by solving the auxiliary problem (AP);
- NC, the total number of constraints in the final relaxed subproblem;
- z^*, the exact optimum solution value to problem (P);
- T(Total) the total running time in seconds on a SPARC 20 workstation;
- T(CG), the time taken by the process of generating violated constraints.

The main observations which can be drawn from Table 2 are the following:

(a) Step 2 of (MCG) is almost never processed (NAP $>$ 0 in only 2 cases over 50). So, for most instances, bipartition inequalities are sufficient to obtain a feasible multicommodity flow solution within a limited number of main iterations.

(b) The average number of main iterations (iter) increases rather moderately with problem size : iter $\approx$ 5 for $N = 8$; iter $\approx$ 9 for $N = 10$; iter $\approx$ 10 for $N = 12$; iter $\approx$ 12 for $N = 15$; iter $\approx$ 13 for $N = 20$. These figures illustrate the relevance of the multiple constraint generation approach as implemented in (MCG).

(c) The last column in Table 2 illustrates the computational efficiency of the process of generating

Table 1
Comparison between single constraint generation (SCG) and multiple constraint generation (MCG)

No. of nodes	(SCG)		(MCG)		Time ratio (SCG)/(MCG)
	No. of iterations	Time (s)	No. of iterations	Time (s)	
8	13	21	6	5.2	4
8	12	23	6	8.2	2.8
8	13	112	6	14.3	7.8
8	13	10	4	1.5	6.6
8	13	4	4	1.8	2.2
8	14	66	4	23.9	2.8
8	13	38	6	13.5	2.8
8	12	11	4	4.3	2.5
8	14	43	6	5.3	8.1
8	13	34	5	11.3	3
10	20	421	9	138	3
10	19	584	8	109	5.3
10	16	242	5	32	7.5
10	30	565	9	47	12
10	12	354	7	123	2.9
10	17	234	8	62	3.8
10	20	637	7	116	5.5
10	14	90	7	26	3.5
10	30	986	9	171	5.8
10	11	157	6	48	3.3
12	58	23 423	11	1 471	15.9
12	10	220	7	150	1.5
12	34	2 594	12	361	7.2
12	46	8 799	9	322	27.3
12	39	12 755	10	1 353	9.4

constraints. It is seen that, in most cases, the time taken (TCG) is negligible as compared with the total computation time (typically less than 1–2%).

(d) For a given problem size N, a significant variability of the results in terms of computing time is observed, the ratios between the longest and the shortest computing time typically range from 6.5 (for $N = 10$) to 66 (for $N = 12$). This suggests that our test problem generator indeed provides a fairly wide sampling of the problem instances, including both easier ones and harder ones.

As far as we know, the above results are the first systematic computational study providing exact optimal solutions to this class of hard network optimization problems. They confirm the practical applicability of an approach based on the use of standard LP software (CPLEX) to solve moderate size instances in this class of hard network optimization problems. Since most of the computation time is spent in running CPLEX (in MIP mode) for solving the restricted problems, the main criterion of efficiency, in our implementation of constraint generation, was to reduce the number of main iterations as much as possible. The (MCG) procedure described in this paper appears to be practically efficient according to this criterion.

We finally mention that possible improvements in computational efficiency might be obtained by further investigating:

- better criteria to select the constraints in the initial restricted problem;
- reduction of the computational effort in solving the restricted problem (R_k) by making use of information gained during the solution of (R_{k-1});
- development of a specialized algorithm (hopefully more efficient than CPLEX) to solve the restricted problems.

Table 2
Results obtained by applying (MCG) on a series of randomly generated test problems

N	M	NV	NC_1	z_1	Iter	NAP	NC	z^*	T(total)	T(CG)
8	13	84	8	380	6	0	23	465	5.2	0.4
8	12	80	7	368	6	0	20	463	8.2	0.3
8	13	83	7	246	6	0	19	358	14.3	0.3
8	13	79	7	172	4	0	18	296	1.5	0.3
8	13	76	7	289	4	0	15	336	1.8	0.4
8	14	89	7	223	4	0	20	483	23.9	0.4
8	13	86	7	396	6	0	22	477	13.5	0.4
8	12	72	7	273	4	1	19	338	4.3	0.5
8	14	81	8	279	6	0	24	357	5.3	0.4
8	13	85	8	406	5	0	23	506	11.3	0.3
10	18	115	8	299	9	0	53	410	138	1.6
10	17	108	8	467	8	0	41	772	109	1.3
10	16	102	7	352	5	0	29	431	32	0.8
10	17	115	9	365	9	0	41	591	47	1.5
10	17	117	8	358	7	0	33	582	123	1.2
10	16	111	8	348	8	0	41	555	62	1.3
10	17	105	9	504	7	0	40	604	116	1.2
10	16	105	9	525	7	0	25	600	26	1.1
10	17	105	6	367	9	2	53	616	171	2.4
10	18	109	10	514	6	0	30	626	48	1.1
12	21	138	9	569	11	0	68	805	1 471	4
12	20	124	11	628	7	0	37	1 011	150	2.4
12	20	134	10	522	12	0	68	858	361	4.2
12	21	130	10	450	9	0	63	704	322	3.3
12	20	130	10	809	10	0	71	996	1 353	3.5
12	20	135	11	426	9	0	60	685	525	3.1
12	20	130	10	682	9	0	53	933	199	3.1
12	21	131	10	427	6	0	37	636	22	2.1
12	20	132	11	698	8	0	52	919	457	2.7
12	21	140	10	488	9	0	48	614	535	3.3
15	26	171	13	557	9	0	79	859	1 621	8
15	27	172	12	938	12	0	132	1 315	10 911	11.2
15	26	165	14	525	8	0	69	743	984	7.2
15	26	167	13	719	7	0	52	973	565	6.2
15	26	166	11	608	10	0	93	1 102	2 724	8.9
15	26	170	12	668	10	0	89	974	662	8.8
15	25	165	13	683	10	0	93	1 214	6 314	8.6
15	26	171	12	670	9	0	83	997	2 302	8
15	26	168	11	850	11	0	97	1 242	13 473	9.7
15	25	162	12	800	11	0	92	1 136	5 179	9.3
20	36	205	17	887	13	0	146	1 263	23 042	37
20	36	213	16	849	12	0	183	1 262	18 795	35
20	35	206	21	707	9	0	103	963	2 139	25
20	35	200	17	686	12	0	147	1 161	12 476	34
20	36	218	16	1 077	12	0	156	1 567	39 792	35
20	35	207	19	1 122	12	0	142	1 581	10 961	34
20	37	218	20	733	13	0	190	1 100	30 644	39
20	35	201	17	1 009	12	0	145	1 491	10 963	34
20	35	200	17	945	14	0	180	1 450	21 140	35
20	36	208	17	977	14	0	196	1 600	51 644	37

Also, extensions of our aproach to cope with survivability constraints (see e.g. [9]) will have to be investigated. This is left for future research.

Acknowledgements

An anonymous referee is gratefully acknowledged for his/her constructive comments on a first version of the paper.

References

[1] R.K. Ahuja, T. Magnanti, J. Orlin, Network Flows: Theory, Algorithms and Applications, Prentice-Hall, Englewood Cliffs, NJ, 1993.
[2] D. Avis, On the extreme rays of the metric cone, Can. J. Math. 32 (1980) 126–144.
[3] F. Barahona, Network design using cut inequalities, SIAM J. Optim. 6 (3) (1996) 823–837.
[4] J.F. Benders, Partitioning procedures for solving mixed-variables programming problems, Numerische Mathematik 4 (1962) 238–252.
[5] C.K. Cheng, T.C. Hu, Ancestor tree for arbitrary multi-terminal cut functions, Ann. Oper. Res. 33 (1991) 199–213.
[6] V. Gabrel, M. Minoux, LP relaxations better than convexification for multicommodity network optimization problems with step-increasing cost functions, Acta Mathematica Vietnamica 22 (1997) 128–145.
[7] R.E. Gomory, T.C. Hu, An application of generalized linear programming to network flows, SIAM J. Appl. Math. 10 (2) (1962) 260–283.
[8] M. Gondran, M. Minoux, Graphes et Algorithmes, third ed., Paris, Eyrolles, 1995.
[9] M. Grötschel, C.L. Monma, M. Stoer, Design of survivable networks, in: Handbook in OR and MS, North-Holland, vol. 7, 1995, pp. 617–672.
[10] H.H. Hoang, Topological optimization of networks: a non-linear mixed integer model employing generalized Benders decomposition, IEEE Trans. Automat. Control AC-27 (1982) 164–169.
[11] J.L. Kennington, A survey of linear cost multicommodity network flows, Oper. Res. 26 (1978) 209–236.
[12] B.W. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs, Bell. Systems Tech. J. 49 (2) (1970) 291–307.
[13] T.L. Magnanti, P. Mirchandani, Shortest paths, single origin-destination network design and associated polyhedra, Networks 23 (1993) 103–121.
[14] T.L. Magnanti, P. Mirchandani, R. Vachani, Modeling and solving the two-facility network loading problem, Oper. Res. 43 (1) (1995) 142–157.
[15] T.L. Magnanti, P. Mireault, R.T. Wong, Tailoring Benders decomposition for uncapacitated network design, Math. Programming Study 26 (1986) 112–154.
[16] M. Minoux, Optimum synthesis of a network with nonsimultaneous multicommodity flow requirements, Ann. Discrete Math. 11 (1981) 269–277.
[17] M. Minoux, Subgradient optimization and Benders decomposition for large scale programming, in: R.W. Cottle, M.L. Kelmanson, B. Korte (Eds.), Mathematical Programming, North-Holland, Amsterdam, 1984, pp. 271–288.
[18] M. Minoux, Network synthesis and optimum network design problems: models, solution methods and applications, Networks 19 (1989) 313–360.
[19] M. Stoer, G. Dahl, A polyhedral approach to multicommodity survivable network design, Numerische Mathematik 68 (1994) 149–167.