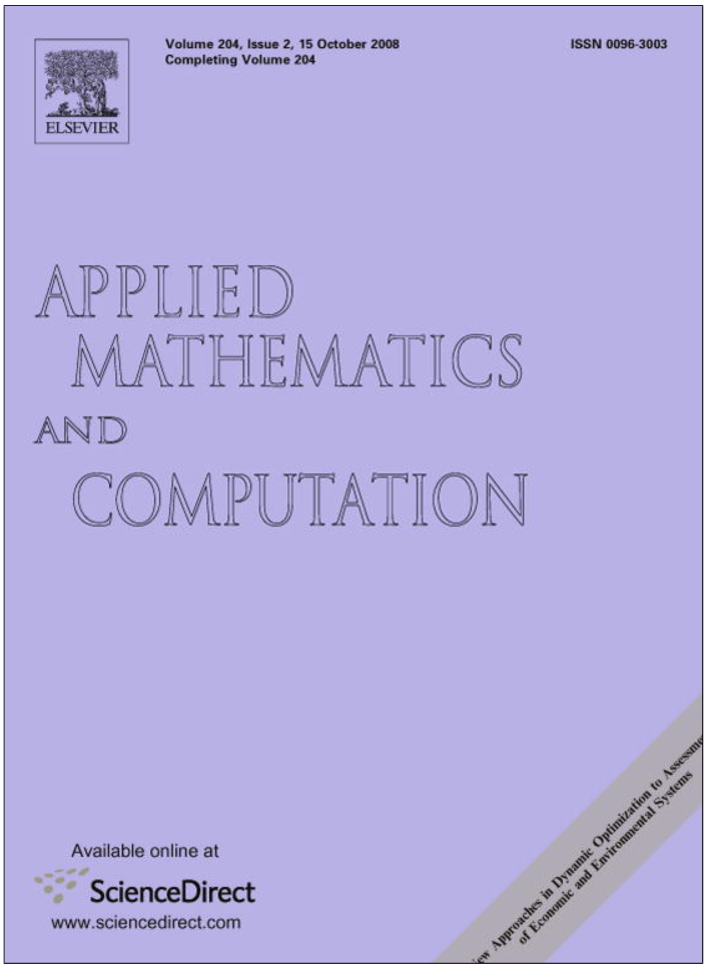
Volume 204, Issue 2, 15 October 2008
Completing Volume 204
ISSN 0096-3003
ELSEVIER
APPLIED MATHEMATICS AND COMPUTATION
Available online at
ScienceDirect
www.sciencedirect.com

Applied Mathematics and Computation 204 (2008) 745–753

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Optimal solution of the discrete cost multicommodity network design problem

Mehdi Mrad [a], Mohamed Haouari [b,*]

[a] ROI – Combinatorial Optimization Research Group, Ecole Polytechnique de Tunisie, BP 743, 2078 La Marsa, Tunisia
[b] Faculty of Economics and Administrative Sciences, Özyegin University, Istanbul, Turkey

A R T I C L E I N F O

Keywords:
Network design
Synthesis of capacitated networks
Multicommodity flows
Constraint generation

A B S T R A C T

We investigate a multicommodity network design problem where a discrete set of technologies with step-increasing cost and capacity functions should be installed on the edges. This problem is a fundamental network design problem having many important applications in contemporary telecommunication networks. We describe an exact constraint generation approach and we show that the conjunctive use of valid inequalities, bipartition inequalities that are generated using max-flow computations, as well as an exact separation algorithm of metric inequalities makes it feasible to solve to optimality instances with up to 50 nodes and 100 edges.

1. Introduction

We address the *Discrete Cost Multicommodity Network Design* problem (DCMND) which is defined as follows. We are given a connected undirected graph $G = (V, E)$, with $|V| = n$ and $|E| = m$, a set of L potential facilities to be installed on each edge, and a set of multicommodity flow requirements corresponding to $K \equiv n(n-1)/2$ distinct point-to-point commodity demand flows. Associated with each commodity k $(k = 1, \ldots, K)$ is a requested flow value d_k that must be routed between a specified source s_k and a specified sink t_k. Each facility l $(l = 1, \ldots, L)$ is characterized by a variable cost κ_l (per unit of length) and a capacity u_l which represents an upper bound on total flow that may pass through it. The installation of facility l on edge $e \in E$ incurs a fixed cost $f_e^l \equiv \theta_e \kappa_l$ where θ_e is the length of edge e. The problem is to design a minimum-cost network by installing at most one facility on each edge in such a way that the installed capacities permit the prescribed K point-to-point commodity demand flows to be routed simultaneously between their respective endpoints.

The DCMND is a fundamental network design problem having many important applications in telecommunication networks where we have available a discrete set of technologies with discrete step-increasing cost and capacity functions. It is noteworthy that a seemingly different multicommodity network design problem with discrete *node* costs has recently attracted the attention of researchers [4]. Actually, this latter problem can be restated as a DCMND by splitting nodes in two and considering them as edges of the network. For an excellent survey of network design problems, the reader in referred to Minoux [12]. Interestingly, although this latter paper has been published about two decades ago, it still constitutes a valuable reference paper on network design problems. In particular, the author describes a simpler version of the DCMND and referred to as the *Optimum Rented Lines Network Problem*. However, during the last decade, the DCMND, in the foregoing general form, has been addressed by several authors. Stoer and Dahl [16] address a slightly more general version with survivability constraints. They derive valid inequalities from the knapsack substructure of the problem and describe a branch-and-cut approach to obtain lower bounds. Approximate solutions are obtained by forcing fractional variables to integral

* Corresponding author.
E-mail address: mh6368@yahoo.com (M. Haouari).

0096-3003/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2008.07.031

values. Gabrel et al. [7] present an integer programming formulation of the DCMND and describe an exact constraint generation algorithm for solving it. They report the solution of instances with up to 20 nodes and 37 links. Minoux [13] presents a general survey of the DCMND and some closely related problems. These latter include the so-called single-facility and two-facility network loading problems. While the former problem refers to the case where the available facility capacities on any edge are multiples of a given basic facility [5], the latter problem refers to the case where capacity expansion can be achieved by means of two types of facilities [6]. In Agarwal [2], a local search heuristic is described. A neighboring solution is obtained by solving a multiple choice knapsack problem using dynamic programming. The author reports that the heuristic produces solutions of instances of up to 20 nodes and 3 facilities within about 5% of lower bound on the average. Finally, Gabrel et al. [8] describe several heuristic approaches for DCMND. These approaches include greedy heuristics as well as a heuristic implementation of the exact algorithm which is presented in Gabrel et al. [7]. This latter approach has been found to perform well on graphs having up to 50 nodes and 90 edges.

In this paper, we are concerned with the exact solution of the DCMND. We describe a constraint (or, row) generation algorithm that can be cast in the same general framework as the approach proposed by Gabrel et al. [7] but exhibits several distinctive features with this latter algorithm. Our algorithm has produced proven optimal solutions for a number of randomly generated instances with up to 50 nodes and 100 edges.

The remainder of this paper is organized as follows. In Section 2, we present a valid mathematical programming formulation. In Section 3, we present a detailed description of our algorithm. In Section 4, we report the results of our computational experiments. Finally, some concluding remarks are provided in the last section.

Throughout the paper, we assume w.l.o.g. that $u_1 < u_2 < \ldots < u_L$ and $f_e^1 < f_e^2 < \ldots < f_e^L\ \forall e \in E$. Moreover, we shall conform with the following notation. For a node subset $W \subset V$, we define $\bar{W} = V \setminus W$ and the cutset $\delta(W) = \{e = \{i,j\} \in E : i \in W$ and $j \in \bar{W}\}$. Also, we define $v(W) = \{j \in \bar{W} : e = \{i,j\} \in \delta(W)\}$.

2. A 0-1 programming formulation of the DCMND

The decision variables are the following

$$y_e^l : \begin{cases} 1, \text{if facility } l \text{ is installed on edge } e, & l = 1,\ldots,L, \quad e \in E \\ 0, \text{otherwise} \end{cases}$$

z_e : capacity assigned to edge e, $e \in E$.

Moreover, we denote for each $\lambda \in \mathscr{R}_+^m$ by $\mu_k^*(\lambda)$ $(k = 1,\ldots,K)$ the value of the shortest path between s_k and t_k in G with respect to the distance matrix $(\lambda_e)_{e\in E}$.

This yields the following model

$$\textbf{DCMND}: \quad \text{Minimize} \sum_{e\in E} \sum_{l=1}^{L} f_e^l y_e^l \tag{1}$$

subject to:

$$\sum_{l=1}^{L} y_e^l \leqslant 1, \quad \forall e \in E, \tag{2}$$

$$\sum_{l=1}^{L} u_l y_e^l - z_e = 0, \quad \forall e \in E, \tag{3}$$

$$\sum_{e\in E} \lambda_e z_e \geqslant \sum_{k=1}^{K} d_k \mu_k^*(\lambda), \quad \forall \lambda \in \mathscr{R}_+^m, \tag{4}$$

$$y_e^l \in \{0,1\}, \quad \forall e \in E, \quad l = 1,\ldots,L. \tag{5}$$

The objective (1) is to minimize the total installation cost. Constraints (2) require that at most one facility should be installed on each edge $e \in E$. Constraints (3) enforce the z_e $(e \in E)$ variables to coincide with the installed capacity on edge e. Constraints (4) require that the capacity vector z is feasible (i.e. it enables the K flows to be routed simultaneously). These constraints, which are often referred to as *metric inequalities*, are necessary because if a linear design cost λ_e is associated with each edge $e \in E$, then a lower bound on the cost for routing flow k $(k = 1,\ldots,K)$ is equal to the shortest path length between source s_k and sink t_k multiplied with the flow value d_k. Hence, a valid lower bound on the total cost for routing the K flows simultaneously is $\sum_{k=1}^{K} d_k \mu_k^*(\lambda)$. Obviously, for any $\lambda \in \mathscr{R}_+^m$ this latter value cannot exceed the total design cost $\sum_{e\in E}\lambda_e z_e$. Moreover, Constraints (4) are sufficient conditions over the space of all possible nonnegative λ-vectors for ensuring the feasibility of the capacity vector. This latter result is a consequence of Farkas Lemma for linear programming duality [14].

It is noteworthy that an alternative equivalent formulation has been derived by Gabrel et al. [7]. The decision variables of this formulation are the binary variables b_e^t $(e \in E, t = 1,\ldots,L)$ where b_e^l takes value 1 if the facility loaded on edge e is $t \geqslant l$, and 0 otherwise. Hence, these variables satisfy

$$b_e^l \leqslant b_e^{l-1}, \quad \forall l = 2,\ldots,L. \tag{6}$$

Clearly, we can express the z_e variables as

$$z_e = \sum_{t=1}^{L} b_e^l (f_e^l - f_e^{l-1}), \quad \forall e \in E \tag{7}$$

with $f_e^0 \equiv 0$, and the y variables as

$$y_e^l = b_e^l - b_e^{l+1}, \quad \forall l = 1, \ldots, L-1, \quad \forall e \in E \tag{8}$$

$$y_e^L = b_e^L \forall e \in E. \tag{9}$$

Remark 1. Model (1)–(5) could be easily extended to accommodate several DCMND variants. In particular, in case where the facilities are modular then it suffices to relax Constraint (2). Also, if multiple identical facilities could be installed on the same link, then the decision variables y_e^l $(e \in E,\ l = 1, \ldots, L)$ should be required to be nonnegative integers instead of binary. Furthermore, the extension to edge-dependent capacities is straight forward.

3. Separation of metric inequalities

Although, it can be shown that it suffices to consider the metric inequalities defined by vectors $\lambda \in \mathscr{R}_+^m$ in a set of extreme rays of a well-defined cone [3], the number of these inequalities is exponential. It is therefore out of question to solve Formulation (1)–(5) without resorting to a constraint generation approach where violated cuts are iteratively generated "on the fly" and appended to a relaxed master program. Given a solution $(\bar{y}, \bar{z})$ to the relaxed master program that is defined by (1), (2), (3), (5) and possibly a restricted subset of metric inequalities (4), we seek for violated metric inequalities by solving the following separation problem.

Find $\lambda \in \mathscr{R}_+^m$, such that

$$\sum_{e \in E} \lambda_e \bar{z}_e < \sum_{k=1}^{K} d_k \mu_k^*(\lambda) \tag{10}$$

or prove that no such vector exists.

In this section, we successively describe two procedures for solving this separation problem. The first approach is an *inexact* nondifferentiable optimization-based separation algorithm, and the second one is an *exact* LP-based separation algorithm.

3.1. Approximate separation of metric inequalities using a deflected subgradient algorithm

Define

$$f(\lambda) = \sum_{k=1}^{K} d_k \mu_k^*(\lambda) - \sum_{e \in E} \lambda_e \bar{z}_e \tag{11}$$

and

$$f^* = \max_{\lambda \geqslant 0} f(\lambda). \tag{12}$$

Clearly, we have

$$f^* \leqslant 0 \iff \text{there is no } \lambda \geqslant 0 \text{ violating } (4).$$

Moreover, since $f(a\lambda) = af(\lambda)$, $\forall a > 0$, then if there exists $\lambda \geqslant 0$ such that $f(\lambda) > 0$ then f^* is unbounded. Consequently, for solving the separation problem, we need to exhibit a vector $\lambda \geqslant 0$ such that $f(\lambda) > 0$ or verify that $f^* \leqslant 0$.

Following Gondran and Minoux [9], we can observe that

$$\sum_{k=1}^{K} d_k \mu_k^*(\lambda) = \sum_{k=1}^{K} d_k \min_{\pi_k \in \Pi_k} \lambda^t \pi_k, \tag{13}$$

where π_k is the incidence vector of a path in G between s_k and t_k, and Π_k is the polytope of the $s_k - t_k$ paths in G. Now, for a given $\lambda \in \mathscr{R}_+^m$, let $\pi_k^*(\lambda)$ denote the incidence vector of a shortest $s_k - t_k$ path in G with respect to the distance matrix $(\lambda_e)_{e \in E}$. Thus, we have

$$\sum_{k=1}^{K} d_k \mu_k^*(\lambda) = \sum_{k=1}^{K} d_k \lambda^t \pi_k^*(\lambda). \tag{14}$$

Hence, we get

$$f(\lambda) = \lambda^t \left(\sum_{k=1}^{K} d_k \pi_k^*(\lambda) - \bar{z} \right). \tag{15}$$

It is easy to check that the vector

$$g \equiv \sum_{k=1}^{K} d_k \pi_k^*(\lambda) - \bar{z} \tag{16}$$

is a subgradient of $f(.)$ at λ.

Consequently, the separation problem defined by (10) might be solved for a given $\bar{z}$, by maximizing $f(.)$. An *approximate* solution to the problem defined by (12) is derived through using a deflected subgradient algorithm which is called the *Average Direction Strategy* (ADS), introduced by Sherali and Ulular [15]. We have chosen to implement ADS because we found that this latter performs extremely well (despite its simplicity) and consistently outperforms other nondifferentiable optimization algorithms [10]. In this procedure, at the qth iteration, a subgradient g^q is computed after solving the shortest path problem between each pair of nodes. This is accomplished using Floyd–Warshall all-pairs shortest-paths algorithm that runs in $O(n^3)$-time. Next, this current subgradient is used to compose a search direction γ^q in order to update the vector λ^q. This direction is computed in the following way:

$$\gamma^q = g^q + \frac{\|g^q\|}{\|\gamma^{q-1}\|}\gamma^{q-1} \quad \text{for } q \geqslant 1 \tag{17}$$

and $\gamma^0 = g^0$.

A formal statement of ADS is given below.

Step 0: Initialization – choose $\lambda^0 = (1, \ldots, 1)$ as a starting Lagrangian multiplier vector and set $q = 0$.
Step 1: Subgradient computation – given λ^q, compute $f(\lambda^q)$ and the current subgradient g^q. If $f(\lambda^q) > 0$ or $g^q = 0$ (practically, if $\|g^q\| < 10^{-6}$), then stop. Otherwise, go to Step 2.
Step 2: Search direction computation – select a search direction γ^q using (17). If $\|\gamma^q\| < 10^{-6}$, then set $\gamma^q = g^q$.
Step 3: Lagrangian multipliers update – set $\lambda^{q+1} = \lambda^q + \epsilon_q \gamma^q$ for some step-length $\epsilon_q = -\beta_q \frac{f(\lambda^q)}{\|\gamma^q\|^2}$, where $\beta_q \in (0, 2]$ is a step-length parameter.
Step 4: Termination test – If ($\|g^q\| < 10^{-6}$) or (the algorithm performs 20 consecutive non-improving iterations) then stop; otherwise, Set $q \leftarrow q + 1$ and go to Step 1.

In the foregoing scheme, the step-length parameter β_q is computed according to the following rule: $\beta_0 = 2$ and β_q is halved from its previous value whenever $f(\lambda^q)$ has failed to increase for five consecutive iterations.

If at some iteration q, a vector satisfying $f(\lambda^q) > 0$ is obtained, then the cut

$$\sum_{e \in E} \lambda_e^q z_e \geqslant \sum_{k=1}^{K} d_k \mu_k^*(\lambda^q) \tag{18}$$

is appended to the relaxed master program.

3.2. Exact separation of metric inequalities using linear programming duality

Given a solution $(\bar{y}, \bar{z})$ to the relaxed master program, we can check whether the installed capacities permit the simultaneous flow of the K distinct point-to-point commodity demand flows by exactly solving a feasibility problem. This is accomplished through finding a feasible multicommodity flow vector x in the digraph $\vec{G} = (V, A)$ that is derived from G by replacing each edge $e \in E$ by two arcs (i, j) and (j, i) having opposite directions. The total flow that may pass through these two arcs should not exceed the capacity of the facility loaded on edge e. Hence, the feasibility problem amounts to solving

$$g(\bar{y}, \bar{z}) \equiv \text{Minimum} \sum_{e \in E} \xi_e \tag{19}$$

subject to:

$$\sum_{j:(i,j) \in A} x_{ij}^k - \sum_{j:(j,i) \in A} x_{ji}^k = \begin{cases} d_k & \text{if } i = s_k \\ 0 & \text{if } i \in V \setminus \{s_k, t_k\}, \forall k = 1, \ldots, K, \\ -d_k & \text{if } i = t_k \end{cases} \tag{20}$$

$$\sum_{k=1}^{K} x_{ij}^k + \sum_{k=1}^{K} x_{ji}^k - \xi_e \leqslant \bar{z}_e, \quad \forall e = \{i, j\} \in E, \tag{21}$$

$$x_{ij}^k \geqslant 0, \quad \forall (i, j) \in A, \quad k = 1, \ldots, K, \tag{22}$$

$$\xi_e \geqslant 0, \quad \forall e \in E, \tag{23}$$

where ξ_e ($e \in E$) is a nonnegative artificial variable associated with edge e.

For small- and medium-sized sparse graphs, Model (19)–(23) might be easily solved using a state-of-the-art LP solver. Obviously, we have

$$g(\bar{y},\bar{z}) > 0 \iff (\bar{y},\bar{z}) \text{ violates a metric inequality}.$$

Now, we describe how to generate such a violated metric inequality. To that aim, we associate with each balance constraint (20) a dual variable α_{ik} $(i \in V,\ k = 1,\ldots,K)$ and with each capacity constraint (21) a nonnegative dual variable β_e $(\forall e = \{i,j\} \in E)$. Since for each $k = 1,\ldots,K$, there is one redundant balance constraint, then we can set

$$\alpha_{t_k,k} = 0 \quad \text{for } k = 1,\ldots,K. \tag{24}$$

The dual of the feasibility problem is

$$\text{Maximize} \sum_{k=1}^{K} d_k \alpha_{s_k,k} - \sum_{e\in E} \bar{z}_e \beta_e \tag{25}$$

subject to:

$$\alpha_{ik} - \alpha_{jk} - \beta_{\{i,j\}} \leqslant 0, \quad \forall (i,j) \in A, \quad k = 1,\ldots,K, \tag{26}$$
$$\beta_e \leqslant 1, \quad \forall e \in E, \tag{27}$$
$$\beta_e \geqslant 0, \quad \forall e \in E. \tag{28}$$

Let (α^*,β^*) denote an optimal dual solution. By duality, we have

$$g(\bar{y},\bar{z}) = \sum_{k=1}^{K} d_k \alpha^*_{s_k,k} - \sum_{e\in E} \bar{z}_e \beta^*_e \tag{29}$$

Thus, $(\bar{y},\bar{z})$ is feasible if and only if the inequality

$$\sum_{e\in E} \beta^*_e z_e \geqslant \sum_{k=1}^{K} d_k \alpha^*_{s_k,k} \tag{30}$$

holds.

Lemma 1. *The inequality defined by (30) is a metric inequality.*

Proof. It suffices to prove that $\alpha^*_{s_k,k}$ is equal to the value of the shortest path in $\vec{G}$ between s_k and t_k $(k = 1,\ldots,K)$.

We observe that at optimality, if $\beta^* \geqslant 0$ is an optimal dual vector, then we can find a corresponding optimal dual vector α^* by solving K independent subproblems, where problem k $(k = 1,\ldots,K)$ is defined by

$$\text{Maximize } d_k \alpha_{s_k,k} \tag{31}$$

subject to:

$$\alpha_{ik} - \alpha_{jk} \leqslant \beta^*_{\{i,j\}}, \quad \forall (i,j) \in A. \tag{32}$$

Now, consider the problem defined by

$$\text{Maximize } \alpha_{s_k,k} \tag{33}$$

subject to (32).

The dual of (32), (33) is

$$\text{Minimize} \sum_{(i,j)\in A} \beta^*_{\{i,j\}} x_{ij} \tag{34}$$

subject to:

$$\sum_{j:(i,j)\in A} x^k_{ij} - \sum_{j:(j,i)\in A} x^k_{ji} = \begin{cases} 1 & \text{if } i = s_k \\ 0 & \text{if } i \in V \setminus \{s_k,t_k\}, \end{cases} \tag{35}$$
$$x^k_{ij} \geqslant 0, \quad \forall (i,j) \in A. \tag{36}$$

Clearly, this problem amounts to finding a shortest path in $\vec{G}$ between s_k and t_k with the arc costs $\beta^*_{\{i,j\}}$. Hence, using the previously introduced notation, we have $\alpha^*_{s_k,k} = \mu^*_k(\beta^*)$. □

Consequently, the exact separation of a metric inequality can be achieved through the exact solution of Model (19)–(23).

Remark 2. The complexity of each iteration of ADS is $O(n^3)$ (which corresponds to the complexity of the Floyd–Warshall all-pairs shortest-paths algorithm). Hence, the complexity of ADS is $O(\theta n^3)$ where θ is the (unknown) number of iterations upon convergence. On the other hand, the exact approach requires solving a linear program having $O(mn^2)$ variables and $O(n^3)$ constraints.

4. Enhancements of the constraint generation algorithm

Actually, a constraint generation algorithm that is solely based on the separation of the foregoing metric inequalities would exhibit an extremely slow convergence [7]. In order to improve the efficiency of the solution approach, we propose several enhancements that are described below.

4.1. Initialization of the constraint generation algorithm

Given $W \subset V$, a *bipartition inequality* has the following form

$$\sum_{e \in \delta(W)} z_e \geqslant d(W), \tag{37}$$

where $d(W)$ refers to the cumulative demand that must pass through the cut set $\delta(W)$. Hence,

$$d(W) = \sum_{k:|W \cap \{s_k, t_k\}|=1} d_k. \tag{38}$$

It is easy to check that a bipartition inequality induced by a subset $W \subset V$ corresponds to a metric inequality that is obtained by setting the λ-vector equal to the incidence vector of the cutset $\delta(W)$.

In our implementation, the initial relaxed master program (P_0) includes a set of bipartition inequalities that are induced by node subsets W_σ^k, $(k = 1, \ldots, K,\ \sigma = 1, \ldots, \tau_k)$ that are obtained using the following iterative procedure.

```
Generation of initial bipartition inequalities
1. Set $W_1^k = \{s_k\}$, $\sigma = 1$,
2. While $(v(W_\sigma^k) \neq \{t_k\})$
Begin
    2.1. $W_{\sigma+1}^k = (v(W_\sigma^k) \setminus \{t_k\}) \cup (W_\sigma^k)$
    2.2. $\sigma \leftarrow \sigma + 1$
End (While)
$\tau_k \leftarrow \sigma$
End.
```

It is noteworthy that a similar procedure has been proved very useful for generating initial cuts of a network design problem with non-simultaneous commodity flow requirements [11].

4.2. Appending violated bipartition inequalities

Given a solution $(\bar{y}, \bar{z})$ to the relaxed problem, we generate a set of violated bipartition inequalities in the following way. For each commodity k $(k = 1, \ldots, K)$, we determine the minimum cut separating s_k and t_k in G where each edge $e \in E$ is assigned a capacity $\bar{z}_e$. Clearly, this can be achieved by computing the maximum flow φ_k between s_k and t_k. Assume that the resulting minimum cut is $\delta(W_k)$. Then, we check if the following inequality

$$\sum_{e \in \delta(W_k)} \bar{z}_e < d(W_k) \tag{39}$$

holds. If the answer is "yes", then Constraint (37) is appended to the relaxed problem. This process is repeated for all commodities in turn. Hence, this procedure requires solving K maximum flow problems. Since, the maximum flow problem is solvable in $O(n^2\sqrt{m})$-time see [1] then the generation of violated bipartition inequalities can be carried out in $O(n^4\sqrt{m})$-time. Actually, this time complexity could be reduced to $O(n^3\sqrt{m})$-time using a special-purpose algorithm that computes the all-pairs minimum cut problem by invoking only $(n-1)$ applications of the single-pair minimum cut problem see for e.g. [1, p. 277-283]. Consequently, violated bipartition inequalities could be very efficiently generated using standard network flow computations. At this point, it noteworthy that Gabrel et al. [7] have implemented a significantly more complex strategy for generating violated bipartition inequalities. Indeed, given a solution $(\bar{y}, \bar{z})$ to the relaxed problem, they maximize the ratio of the right to the left hand sides of (39). Thus, they solve the following *maximum ratio cut* problem:

$$(\mathrm{BP}) : \text{Find } W^* = \arg\max_{W \subset V} \rho(W) \text{ where } \rho(W) \equiv \frac{d(W)}{\sum\limits_{e \in \delta(W)} z_e} \text{ for } W \subset V.$$

Clearly, if $\rho(W^*) > 1$ then the bipartition inequality induced by W^* is appended to the relaxed master program. However, *BP* is an $\mathcal{NP}$-hard problem and is therefore solved *approximately* via a variable-depth local search heuristic.

4.3. Appending valid inequalities

Given a feasible solution $\bar{y}$, the graph $G(\bar{y}) = (V, A(\bar{y}))$ that is obtained from G by removing all the edges $e \in E$ such that $\sum_{l=1}^{L} \bar{y}_e^l = 0$ should be connected. Hence, we include in the initial relaxed master program (P_0) the obvious cut

$$\sum_{e \in E} \sum_{l=1}^{L} y_e^l \geqslant n - 1. \tag{40}$$

Moreover, define q^* as the smallest possible sum of capacities that should be installed in any feasible solution. Clearly, this value can be obtained by solving a multicommodity flow problem with all capacities equal to u_L (the largest values) and unit flow costs. Hence, we have

$$q^* \equiv \quad \text{Minimum} \sum_{k=1}^{K} \sum_{(i,j) \in A} x_{ij}^k \tag{41}$$

subject to (20), (22), and

$$\sum_{k=1}^{K} x_{ij}^k + \sum_{k=1}^{K} x_{ji}^k \leqslant u_L, \quad \forall \{i, j\} \in A. \tag{42}$$

Consequently,

$$\sum_{e \in E} \sum_{l=1}^{L} u_l y_e^l \geqslant q^* \tag{43}$$

is a valid inequality. Therefore, this latter inequality is appended to the initial relaxed master program.

Furthermore, if at some iteration of the constraint generation process the current solution $\bar{y}$ is proven to violate some metric inequality, then we append to the relaxed master program an additional cut that is generated as follows. Let $\bar{E} = \{e \in E : \bar{y}_e^L = 0\}$ and $l(e)$ $(e \in \bar{E})$ be the index of the capacity chosen for edge e in the current solution. Clearly, in any feasible solution there is at least one edge that must be loaded with a higher capacity. Consequently, the cut

$$\sum_{e \in \bar{E}} \sum_{l=l(e)}^{L} y_e^l \geqslant 1 \tag{44}$$

should be appended to the current relaxed master program.

4.4. Synthesis of the constraint generation procedure

Since bipartition inequalities can be efficiently generated, we initialize the constraint generation process by generating these cuts. If the algorithm fails to identify a violated bipartition inequality, then we invoke the ADS for generating a violated metric inequality. Although being inexact, this procedure often requires less computational effort than the exact LP-based approach. When ADS fails to identify a violated metric inequality, the exact separation algorithm is invoked. A synthesis of the proposed approach is given below.

Step 1: Initialization. Let P_0 be the problem defined by (1), (2), (3), (5), (40), (43) and the initial bipartition inequalities that are described in Section 4.1. Set $q = 0$.

Step 2: MIP-solver. Set $q \leftarrow q + 1$. Solve P_q using a MIP solver. Let $(\bar{y}, \bar{z})$ be an optimal solution to P_q.

Step 3: Bipartition cut identification. Generate violated bipartition inequalities by computing the minimum cut separating each pair $\{s_k, t_k\}$ $(k = 1, \ldots, K)$. If one or more violated bipartition constraints are found, then define P_{q+1} to be P_q amended by the violated bipartition constraints and go to Step 2. Else, go to Step 4.

Step 4: Approximate separation of a metric inequality. Invoke algorithm ADS for identifying a violated metric inequality. If a violated metric inequality is found, then define P_{q+1} to be P_q amended by the violated inequalities (18) and (44) and go to Step 2. Else, go to Step 5.

Step 5: Exact feasibility test. Invoke an LP solver to check if $(\bar{y}, \bar{z})$ is feasible. If a feasible multicommodity flow is found then output $(\bar{y}, \bar{z})$ as an optimum. Else go to Step 6.

Step 6: Exact separation of a metric inequality. Define P_{q+1} to be P_q amended by the violated inequalities (30) and (44) and go to Step 2.

5. Computational experiments

We have coded the proposed algorithm in Microsoft Visual C++ (version 6.0) in concert with the CPLEX 9.0 solver. All the computational experiments were carried out on a Pentium IV 2.4 GHz Personal Computer with 3.2 GB RAM. A maximum CPU time limit was set to 1 h.

The test-bed we have used consists of a set of 20 randomly generated instances. The numbers of nodes and edges range from 10 to 50, and 15 to 100, respectively, and the number of facility types is $L = 3$ for all instances. In order to account for the economy of scale phenomenon which is very common in the telecommunication setting, the cost of installing a capacity u_l on edge $e \in E$ is a concave function of its capacity and is computed as $f_e^l = \lceil \theta_e \sqrt{u_l} \rceil$ where θ_e is the length of edge e.

Results are displayed in Table 1. For each instance, we report n : number of nodes, m : number of edges, K : number of commodities, Time: total CPU time in seconds, Sol: value of the optimal solution, BC: number of generated bipartition cuts, MI-ADS : number of metric inequalities that are generated using ADS, MI-LP: number of metric inequalities that are generated using the LP approach.

We see from Table 1 that the proposed approach is able to solve to optimality all the instances, but only two, within a reasonable CPU time. However, we observe that the metric inequalities are scarcely generated. This observation is consistent with the results reported by Gabrel et al. [7]. On the contrary, the bipartition cuts, despite their simplicity, prove to be extremely useful for deriving proven optimal solutions.

Table 1
Results of the constraint generation approach

n	m	K	Time	Sol	BC	MI-ADS	MI-LP
10	15	45	0.078	1041	72	0	0
15	20	105	0.234	2101	268	0	0
15	25	105	0.39	2500	293	0	0
15	30	105	0.578	2447	408	0	0
20	35	190	0.797	3572	674	0	0
20	40	190	0.625	3537	599	0	0
20	45	190	0.75	3392	759	0	0
21	40	210	8.531	4245	2031	3	0
22	45	231	1.735	4961	1167	0	0
23	50	253	16.563	5395	2815	0	0
24	55	276	4.172	5912	1229	0	0
25	50	300	1103.845	6154	7593	0	8
25	60	300	149.957	5705	4998	0	0
30	60	435	108.643	8645	7259	0	0
30	65	435	155.941	8170	7538	0	7
35	70	595	>3600	[a]	14368	0	9
40	75	780	860.148	15505	17667	0	0
45	80	990	>3600	[a]	26464	0	4
50	90	1225	93.719	24254	11585	0	0
50	100	1225	26.344	23275	7273	0	0

[a] Indicates that the instance remained unsolved after 1 h CPU time.

Table 2
Impact of removing the initial cuts

n	m	K	Time	Sol	BC	MI-ADS	MI-LP	Time-Ratio
10	15	45	0.171	1041	241	0	0	2.205
15	20	105	0.374	2101	597	0	0	1.602
15	25	105	1.077	2500	626	0	0	2.764
15	30	105	3.687	2447	938	0	0	6.38
20	35	190	2.296	3572	1388	0	0	2.882
20	40	190	2.625	3537	1208	0	0	4.2
20	45	190	7.453	3392	1856	0	0	9.938
21	40	210	58.514	4245	2641	0	0	6.859
22	45	231	23.904	4961	2675	0	0	13.778
23	50	253	1171.517	5395	5438	0	0	70.731
24	55	276	668.871	5912	6350	0	0	160.324
25	50	300	>3600	[a]	9347	0	0	[a]
25	60	300	>3600	[a]	7419	0	0	[a]
30	60	435	>3600	[a]	9538	0	0	[a]
30	65	435	>3600	[a]	9343	0	0	[a]
35	70	595	>3600	[a]	12742	0	0	[a]
40	75	780	>3600	[a]	16860	0	0	[a]
45	80	990	>3600	[a]	17405	0	0	[a]
50	90	1225	>3600	[a]	24007	0	0	[a]
50	100	1225	>3600	[a]	25162	0	0	[a]

[a] Indicates that the instance remained unsolved after 1 h CPU time.

In addition, we have investigated the impact of the initial cuts. To that aim, we have dropped these constraints and initialized the constraint generation process from scratch. The results are displayed in Table 2. In this Table, the column Time_-Ratio includes the ratio of the CPU time of the simplified variant to the original algorithm. Table 2 shows the advantage of including the initial cuts. Indeed, we observe that removing these cuts caused, a very significant increase of the required CPU time. Moreover, 9 instances remained unsolved after 1 h CPU time.

6. Conclusion

In this paper, we have described an exact approach for solving the discrete cost multicommodity network design problem. We have presented the results of computational experiments that provide evidence that the efficient combination of effective initial valid cuts, bipartition inequalities (very simply) generated through max-flow computations, and an LP-based exact separation algorithm of metric inequalities render the ability to optimally solve instances having up to 50 nodes and 100 edges.

As a topic for future research, we recommend the investigation of an alternative formulation of DCMND with decision variables representing paths between sources and sinks. This latter formulation could be solved using a column generation approach. We expect that a state-of-the-art branch-and-price algorithm would prove useful for solving large-scale DCMND instances but this needs to be investigated carefully.

Acknowledgement

The authors would like to thank an anonymous referee for kindly suggesting (30), (43) and (44) and for helpful comments which improved the presentation of this paper.

References

[1] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows: Theory, Algorithms and Applications, Prentice Hall, Upper River, New Jersey, 1993.
[2] Y.K. Agarwal, Design of capacitated multicommodity networks with multiple facilities, Operations Research 50 (2002) 333–344.
[3] D. Avis, On the extreme rays of the metric cone, Canadian Journal of Mathematics 32 (1980) 126–144.
[4] P. Belotti, L. Brunetta, F. Malucelli, Multicommodity network design with discrete node costs, Networks 49 (2007) 100–115.
[5] D. Bienstock, S. Chopra, O. Günlük, C.Y. Tsai, Minimum-cost capacity installation for multicommodity network flows, Mathematical Programming 81 (1998) 177–199.
[6] T.L. Magnanti, P. Mirchandani, R. Vachani, Modeling and solving the two-facility capacitated network loading problem, Operations Research 43 (1995) 142–157.
[7] V. Gabrel, A. Knippel, M. Minoux, Exact solution of multicommodity network optimization problems with general step cost functions, Operations Research Letters 25 (1999) 15–23.
[8] V. Gabrel, A. Knippel, M. Minoux, A comparison of heuristics for the discrete cost multicommodity network optimization problem, Journal of Heuristics 9 (2003) 429–445.
[9] M. Gondran, M. Minoux, Graphs and Algorithms, Wiley Interscience, New York, 1984.
[10] M. Haouari, S.B. Layeb, H.D. Sherali, The prize collecting Steiner tree problem: models and Lagrangian dual optimization approaches, Computational Optimization and Applications 40 (2008) 13–39.
[11] M. Haouari, M. Mrad, H.D. Sherali, Optimum synthesis of discrete capacitated networks with multi-terminal commodity flow requirements, Optimization Letters 1 (2007) 341–354.
[12] M. Minoux, Network synthesis and optimum network design problems: models, solution methods and applications, Networks 19 (1989) 313–360.
[13] M. Minoux, Discrete cost multicommodity network optimization problems and exact solution methods, Annals of Operations Research 106 (2001) 19–46.
[14] K. Onaga, O. Kakusho, On feasibility conditions of multicommodity flows in networks, IEEE Transactions on Circuit Theory 18 (1971) 425–429.
[15] H.D. Sherali, O. Ulular, Conjugate gradient methods using quasi-newton updates with inexact line searches, Journal of Mathematical Analysis and Applications 150 (1990) 359–377.
[16] M. Stoer, G. Dahl, A polyhedral approach to multicommodity survivable network design, Numerische Mathematik 68 (1994) 149–167.