Available online at www.sciencedirect.com
SCIENCE DIRECT®
Discrete Optimization 3 (2006) 195–205
DISCRETE OPTIMIZATION
www.elsevier.com/locate/disopt

Revisiting parametric multi-terminal problems: Maximum flows, minimum cuts and cut-tree computations

D. Barth[a], P. Berthomé[b,*], M. Diallo[c,1], A. Ferreira[d,2]

[a] *Laboratoire PRiSM, UMR 8144, CNRS, Université de Versailles, 45 Av. des Etats-Unis, 78035 Versailles-Cedex, France*
[b] *Laboratoire de Recherche en Informatique (LRI), UMR 8623, CNRS, Université Paris-Sud, 91405, Orsay-Cedex, France*
[c] *Depto. Eng. Industrial, Pontifícia Universidade Católica (PUC-RIO) Rua Marquês de São Vicente, 225, Gávea Rio de Janeiro, RJ - 22453-900, Brazil*
[d] *COST Office, Avenue Louise 149, 1050 Brussels, Belgium*

Received 6 October 2004; received in revised form 1 February 2006; accepted 1 May 2006
Available online 14 July 2006

Abstract

Given an undirected network, the multi-terminal network flows analysis consists in determining the all pairs maximum flow values. In this paper, we consider an undirected network in which some edge capacities are allowed to vary and we analyze the impact of such variations on the all pairs maximum flow values. We first provide an efficient algorithm for the single parametric capacity case, and then propose a generalization to the case of multiple parametric capacities. Moreover, we provide a study on Gomory–Hu cut-tree relationships.

Keywords: Multi-terminal network flows; Cut-trees; Minimum cuts; Sensitivity analysis

1. Introduction

Given an undirected network with n vertices, the multi-terminal network flows analysis consists in determining the $\frac{n(n-1)}{2}$ all pairs maximum flow values. In 1964, Gomory and Hu [1] showed that there exist only $(n-1)$ distinct such values. Moreover, the authors provided an algorithm that outputs a cut-tree that allows one to get all the possible maximum flow values after $(n-1)$ maximum flow computations. Furthermore, this cut-tree compactly represents a minimum cut for each pair of vertices in the network.

To date, many variants of the problem were studied (an exhaustive survey on the subject is given in [2]), with applications being identified in several areas, like transport, energy, finance, and telecommunications ([2–4] and references therein). To the best of our knowledge, all problems dealing with the multi-terminal network flows problem

* Corresponding author. Tel.: +33 1 69 15 42 26; fax: +33 1 69 15 65 86.
E-mail addresses: barth@prism.uvsq.fr (D. Barth), berthome@lri.fr (P. Berthomé), diallo@ind.puc-rio.br (M. Diallo), Afonso.Ferreira@cost.esf.org (A. Ferreira).

[1] This work is a postdoc research funded by the French CNRS ACI-Sécurité Project.

[2] On leave from CNRS, France. External collaborator of the MASCOTTE team at INRIA Sophia Antipolis.

1572-5286/$ - see front matter
doi:10.1016/j.disopt.2006.05.003

in undirected networks use the concept of cut-tree with one of the two existing algorithms for such trees [1,5]. Please note that such algorithms are based on the classic computation of maximum flows.

We notice that the Gomory and Hu results only apply to the case of undirected networks. Extensions to the directed case were first attempted in [6,7], but they were proved wrong in [8]. We refer the reader who is interested in related problems in directed networks to [9].

In this paper, we focus on some sensitivity analysis aspects of the multi-terminal flows problem. Hereafter, the term *network* indicates an *undirected network*. We seek to analyze the impact of edge capacity variations on the all pairs maximum flow values. An edge with a varying (parametric) capacity will be denoted as a *parametric* edge. The problem we deal with will be called *parametric multi-terminal network flows*.

In 1964, Elmaghraby [3] was the first to study the effect of a single edge capacity variation on the all pairs maximum flow values. Given a network in which a single edge capacity is allowed to decrease to zero, the author showed that for each pair of vertices, with respect to such a variation, either the maximum flow value remains constant or there exists a value of the varying capacity referred to as the *critical capacity* for which the maximum flow value begins to decrease linearly with the parametric capacity. Thus, in order to obtain the all pairs maximum flow values, the author proposed computing all critical capacities. Note that this algorithm needs to compute one cut-tree for the determination of each critical capacity, and more than one maximum flow value may have the same critical capacity.

Our first contribution in this paper is to show that if one single capacity varies, then just two cut-trees are sufficient to solve the former Elmaghraby sensitivity analysis. We then turn to providing an analysis for the generalized parameterization case. Our main result, first presented in [10], states that in the case where k edge capacities vary in the network, obtaining the all pairs maximum flows for any value of the k parameters is polynomial whenever $k = O(\text{polylog } n)$, since we show that it can be solved with the computation of only 2^k cut-trees. We also adapt these results and methods to provide, in the same setting, an efficient way to determine a minimum cut for each vertex pair.

Finally, the results in [11] highlight the importance of re-studying cut-tree relationships for the sake of reducing computation complexity. In seeking efficient ways to compute minimum cuts without using the saturating path concept of Menger's theorem or its extension of Max-Flow/Min-Cut, the authors focused on the concept of vertex degree domination and established several properties with respect to network vertices, edges, and minimum cuts that help to set up cut-tree relationships. Our last contribution in this paper focuses on a special relationship: the one between cut-trees that are computed in sequence. Aiming at decreasing the complexity of cut-tree computations in any process that needs more than one cut-tree computation, we use some results from [11] and show how to reuse information from an already computed cut-tree in order to efficiently construct the next one in the sequence.

The remainder of this paper is organized as follows. In Section 2, we give the flow theory definitions we use throughout the paper. Section 3 is devoted to the resolution of the sensitivity analysis problem in terms of maximum flow and of minimum cut computations when just a single capacity varies. Section 4 provides a generalization of the analysis to more than one capacity variation. Section 5 details our study on cut-tree relationships. We close the paper by providing some open problems.

2. Definitions

2.1. Gomory–Hu cut-trees

Let $G = (V, E)$ be an undirected connected network with vertex set V and edge set E. Throughout this paper, we denote $|V|$ as n and $|E|$ as m. With each edge $e \in E$ is associated a positive capacity $c(e)$. Let us consider the symmetric digraph $G^* = (V, A)$ obtained from G by replacing each edge e by two opposite arcs with the same capacity $c(e)$. A *flow between two vertices s and t in G* is a flow from s to t (or the opposite one from t to s) in G^* as defined by Ford and Fulkerson [12]. Thus, we denote by $f_{s,t} (= f_{t,s})$ the value of the maximum flow between s and t in G. Vertices s and t are called the *sinks* of the flow.

We consider here a multi-terminal network, i.e., we consider all the possible pairs of sinks in G, but we do not consider simultaneous flows between different pairs (we do not deal with a multi-commodity flow problem).

Considering a graph $G = (V, E)$ with a capacity function c, a *cut separating two vertices u and v* in G is a proper subset $C_{s,t}(G)$ of V ($\emptyset \subsetneq X \subsetneq V$) such that $u \in X$ and $v \in V \setminus X$. Such a cut induces the set of edges

$$\{[x, y] \in E : x \in C_{s,t}(G), y \notin C_{s,t}(G)\}.$$

We say that such an edge *belongs* to the cut.

The capacity of such a cut $C_{s,t}(G)$ is defined by

$$c(C_{s,t}(G)) = \sum_{x \in C_{s,t}(G), y \in V \setminus C_{s,t}(G), [x,y] \in E} c[x, y],$$

i.e., the capacity of the cut is given by the set of edges $[x, y]$ induced by it.

A *minimum cut* separating vertices u and v, denoted hereafter as $C_{u,v}$, is a cut with minimal capacity among all the cuts separating u and v.

Definition 1. Given a network $G = (V, E)$ with a capacity function c, a *(Gomory–Hu) cut-tree* $T = (V, F)$ obtained from G is a tree having the same set of vertices V and an edge set F with a capacity function c' having the following two properties:

(a) *Equivalent flow tree*: for any pair of vertices s and t, $f_{s,t}$ in G is equal to the smallest capacity (with c') of the edges on the path between s and t in T, i.e., to the value of the maximum flow between s and t in T;
(b) *Cut property*: if a proper subset of V is a minimum cut separating s and t in T, it is also a minimum cut separating s and t in G.

As shown in [1], $(n - 1)$ minimum cut computations are sufficient for constructing a cut-tree. We notice that cut-trees are generally not unique. This fact will be used in our work. Two different ways of computing a cut-tree are provided in [1] and in [5]. An experimental study of minimum cut algorithms and a comparison of algorithms producing cut-trees are provided in [13].

Definition 2. Given a network in which some capacities are allowed to vary, *the parametric multi-terminal network flows problem* consists in obtaining the all pairs maximum flow values with regard to the capacity variations. When there is only one parametric edge capacity, this problem is also known as sensitivity analysis on multi-terminal flows [3].

Note that Diallo [2] shows that Elmaghraby's method given in [3] does not work in all cases; he provides an improvement of the technique that does not reduce the overall complexity.

In the sequel, when one capacity $c(e)$ is allowed to vary ($c(e) = \lambda$), we will denote as $f^{\lambda}_{s,t}$ (resp. $C^{\lambda}_{s,t}$) the maximum flow value (resp. a minimum cut). When k capacities are allowed to vary, $f^{\lambda_1,\lambda_2,\cdots}_{s,t}$ (resp. $C^{\lambda_1,\lambda_2,\cdots}_{s,t}$) will denote the maximum flow value (resp. a minimum cut) with $c(e_i) = \lambda_i$, $1 \leq i \leq k$.

2.2. Critical capacity

Let us consider the behavior of the maximum flow value function $\lambda \mapsto f^{\lambda}_{s,t}$, for a given pair of sinks $\{s, t\} \neq \{i, j\}$. In the case where this function varies with λ, as shown in Fig. 1, it consists of two distinct parts:

- As long as the capacity λ of the edge e increases, the maximum flow value increases in the same way, i.e., with slope 1. During this stage, the parameterized edge occurs in any (s, t)-minimum cut.
- At some value $\lambda^*_{s,t}$ of λ, namely the *critical capacity*, the maximum flow becomes saturated. During this stage, the parameterized edge is out of all (s, t)-minimum cuts.

If we restrict the variation of the capacity to being between two values α and β, $0 \leq \alpha < \beta \leq \infty$, the behavior of the function $\lambda \mapsto f^{\lambda}_{s,t}$ clearly corresponds to Fig. 1, restricted to the interval $[\alpha, \beta]$.

Consequently, when $\{s, t\} \neq \{i, j\}$, the maximum flow value has a finite limit. In this sense, we define $f^{\infty}_{s,t}$ to be such a limit. However, a maximum flow value $f_{s,t}$ may never depend on the parameter: $f^{0}_{s,t} = f^{\infty}_{s,t}$. In this case, we say by convention that there is no critical value in the interval $[0, \infty[$. Notice that $f_{i,j} \to \infty$ as $\lambda \to \infty$; thus by convention we also admit that $f^{\infty}_{i,j} = \infty$ and $\lambda^{\infty}_{i,j} = \infty$.

3. Effect of a parameterized edge on several classical problems

In this section, we consider a graph $G = (V, E)$ and $e = [i, j] \in E$ as the parametric edge investigated.

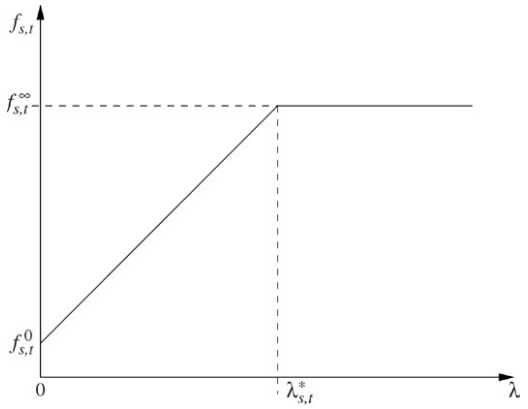

Fig. 1. Behavior of a sensitive maximum flow value $f^{\lambda}_{s,t}$.

3.1. Critical capacity with respect to a single vertex pair

Lemma 3. *Let $G = (V, E)$ be a network, $e = [i, j] \in E$ and $c(e) = \lambda \in [\alpha, \beta]$, $0 \leq \alpha < \beta \leq \infty$. Let p and q be two vertices of G. The critical capacity $\lambda^{*}_{p,q}$ exists if $\{p, q\} \neq \{i, j\}$ and $f^{\alpha}_{p,q} \neq f^{\beta}_{p,q}$, and it satisfies:*

$$\lambda^{*}_{p,q} = \alpha + f^{\beta}_{p,q} - f^{\alpha}_{p,q}. \tag{1}$$

Proof. This is a direct consequence of the behavior of the maximum flow value function. If $f^{\alpha}_{p,q} \neq f^{\beta}_{p,q}$, it grows linearly from $f^{\alpha}_{p,q}$, when $\lambda = \alpha$, up to $f^{\beta}_{p,q}$, then stagnates. Thus the breakpoint corresponds to the capacity: $\alpha + f^{\beta}_{p,q} - f^{\alpha}_{p,q}$. □

Proposition 4. *The critical capacity $\lambda^{*}_{s,t}$ of a parametric capacity for an arbitrary vertex pair s, t can be computed using only two maximum flow computations.*

Proof. Using Lemma 3, we deduce that only $f^{0}_{s,t}$ and $f^{\infty}_{s,t}$ are necessary for computing $\lambda^{*}_{s,t}$. □

Proposition 5. *Let $G = (V, E)$ be a network and $e = [i, j] \in E$ with $c(e) = \lambda$. Let s and t be two vertices of G, α and β two positive values such that $0 \leq \alpha < \beta \leq \infty$. The maximum flow value $f^{\lambda}_{s,t}$ verifies for $\lambda \in [\alpha, \beta]$:*

$$f^{\lambda}_{s,t} = \begin{cases} f^{\alpha}_{s,t} - \alpha + \lambda, & \text{if } \lambda < \lambda^{*}_{s,t} \\ f^{\beta}_{s,t}, & \text{otherwise} \end{cases} \tag{2}$$

or more simply:

$$f^{\lambda}_{s,t} = \min(f^{\alpha}_{s,t} + \lambda - \alpha, f^{\beta}_{s,t}). \tag{3}$$

Note that, in the case of a network for which the edge investigated is a cut-edge, i.e., an edge whose removal disconnects the graph, the previous formula is also valid, since when the edge investigated is removed, the maximum flow between two vertices, one in each side of the cut-edge, is zero due to disconnectivity, and the critical capacity is simply $f^{\infty}_{s,t}$.

3.2. All pairs maximum flow values with respect to a single parametric edge

Theorem 6. *Let $G = (V, E)$ be a network with n nodes. If only a single capacity is allowed to vary, then the set of all critical capacities can be computed using two cut-trees in $O(n^2 \log n)$.*

Proof. Let us make two remarks. First, for a given single pair of vertices s and t of the network, Lemma 3 gives a way to compute the unique critical capacity in constant time given the maximum flow values $f^0_{s,t}$ and $f^\infty_{s,t}$. Second, a cut-tree provides the all pairs maximum flow values.

Thus, the desired result can be obtained with the computation of a cut-tree where the edge investigated e is removed in order to obtain all the $f^0_{s,t}$, $\forall s, t \in V(G)$, and the computation of a second cut-tree where the capacity of the edge e is set to ∞. This latter computation provides all the $f^\infty_{s,t}$, $\forall s, t \in V(G)$. By Lemma 3, from these two maximum flow values, we get all critical capacities by computing and sorting increasingly the differences

$$f^\infty_{s,t} - f^0_{s,t}, \quad \forall s, t \in V(G).$$

This step considers $n(n-1)/2$ pairs of vertices, and thus it can be performed in $O(n^2 \log n)$. □

An algorithm can be directly obtained from this proof.

Theorem 7. *Let $G = (V, E)$ be a network with $e = [i, j]$ and $c(e) = \lambda$. Let α and β be two positive values such that $0 \le \alpha < \beta \le \infty$. Let GH^α and GH^β denote two cut-trees respectively when $c(e) = \alpha$ and $c(e) = \beta$. For $\lambda \in [\alpha, \beta]$, and any pair of nodes s and t in G, the value $f^\lambda_{s,t}$ can be computed in linear time.*

Proof. Using Eq. (3), both $f^\alpha_{s,t}$ and $f^\beta_{s,t}$ are required to compute $f^\lambda_{s,t}$. Both of them can be retrieved, respectively from GH^α and GH^β, in linear time. □

3.3. All pairs min-cuts with respect to a single parametric edge

Lemma 8. *Let $G = (V, E)$ be a network with $e = [i, j] \in E$ and $c(e) = \lambda$. Let s and t be two vertices of G, and α, β two values such that $0 \le \alpha < \lambda^*_{s,t} < \beta$. Then the following holds:*

(1) *Any minimum cut $C^\alpha_{s,t}$ remains a minimum cut when $\lambda \in [0, \lambda^*_{s,t}]$.*

(2) *Any minimum cut $C^\beta_{s,t}$ remains a minimum cut when $\lambda \in [\lambda^*_{s,t}, \infty]$.*

Proof. Let us prove the second case first. Assume that $c(e) = \lambda \ge \lambda^*_{s,t}$. Considering Eq. (3), we have

$$f^\lambda_{s,t} = \min(f^0_{s,t} + \lambda, f^\infty_{s,t}) = f^\infty_{s,t}.$$

Since $\beta > \lambda^*_{s,t}$, we also have

$$f^\beta_{s,t} = c(C^\beta_{s,t}) = f^\infty_{s,t} = f^\lambda_{s,t}.$$

Thus, by the Max-Flow/Min-Cut Theorem, $C^\beta_{s,t}$ is also a minimum cut when $c(e) \ge \lambda^*_{s,t}$.

Let us consider now the first case, i.e., $\lambda \le \lambda^*_{s,t}$. From Eq. (3), we have $f^\lambda_{s,t} = f^0_{s,t} + \lambda$.

In this case, e belongs to $C^\alpha_{s,t}$. If not, when $\lambda > \alpha$, $C^\alpha_{s,t}$ remains a cut (not necessarily minimum) separating s and t; thus

$$c(C^\alpha_{s,t}) \ge f^\lambda_{s,t}.$$

By Eq. (3), we have

$$f^\alpha_{s,t} \ge f^\alpha_{s,t} + \lambda - \alpha$$

leading to a contradiction. Thus e belongs to $C^\alpha_{s,t}$.

Consequently, if $c(e) = \lambda$, the capacity of $X = C^\alpha_{s,t}$ is given by

$$c(X) = \left(\sum_{x \in X, y \in V \setminus X, [x,y] \in E \setminus \{e\}} c[x, y] \right) + \lambda.$$

From Eq. (3), this shows that $C^\alpha_{s,t}$ is a minimum cut for $\lambda = 0$ (where e does not exist) and

$$f^0_{s,t} = \sum_{x \in X, y \in V \setminus X, [x,y] \in E \setminus \{e\}} c[x, y].$$

Thus, by the Max-Flow/Min-Cut Theorem, $C^\alpha_{s,t}$ is a minimum cut in $[0, \lambda^*_{s,t}]$. □

This lemma implies the following proposition.

Proposition 9. *Let $G = (V, E)$ be a network with $e = [i, j] \in E$ and $c(e) = \lambda$. Given two distinct positive values α and β, $0 \leq \alpha < \beta \leq \infty$, let s and t be two arbitrary vertices of G and $C^{\alpha}_{s,t}$ (resp. $C^{\beta}_{s,t}$) be a minimum cut separating s and t when $\lambda = \alpha$ (resp. β).*

For any $\lambda \in [\alpha, \beta]$, at least one of $C^{\alpha}_{s,t}$ and $C^{\beta}_{s,t}$ is a minimum cut that separates s and t.

Proof. This is a direct consequence of Lemma 8. Three cases are to be considered based on the position of the critical capacity $\lambda^*_{s,t}$ compared to α and β.

(1) Lemma 8 deals with the case $\lambda^*_{s,t} \in [\alpha, \beta]$.
(2) If $\alpha > \lambda^*_{s,t}$ Lemma 8 implies that $C^{\alpha}_{s,t}$ and $C^{\beta}_{s,t}$ have the same capacity, and thus both remain minimum cuts over the whole interval.
(3) The case where $\beta < \lambda^*_{s,t}$ is obtained similarly. □

Theorem 10. *Let $G = (V, E)$ be a network with $e = [i, j] \in E$ and $c(e) = \lambda$. Let GH^{α} and GH^{β} be two cut-trees for two distinct values of the parameter, $0 \leq \alpha < \beta \leq \infty$. Let s and t be two vertices of G and $\lambda \in [\alpha, \beta]$. Then a minimum cut between s and t can be determined in linear time $O(n)$.*

Proof. This directly follows from Theorem 7 and Proposition 9. The explorations of their cut are linear in their size. □

4. Applications

In this section, we give two direct applications of the previous theorems. In Section 4.1, we investigate the case where several capacities are subject to variations. In Section 4.2, given some cut-trees for two values α and β of the parameter, we consider the problem of computing a new cut-tree for an intermediate value.

4.1. Extension to multiple parametric edges

We examine in detail the case where the capacities of two edges vary independently. The main result of this section is that the algorithm given by Theorem 6 can also be applied in the current case, and only four (actually, $2^{\mathbf{2}}$) maximum flow computations are needed to compute any maximum flow value, whatever the value of the capacities of the selected edges are.

The general problem in this section can be formally stated as follows. Given a network $G = (V, E)$ and two distinct edges e_1 and e_2, we want to determine the maximum values of flow between all pairs of vertices when $c(e_i) = \lambda_i$, $i = 1, 2$.

We first consider a pair of vertices s and t in the network and provide a way to compute $f^{\lambda_1,\lambda_2}_{s,t}$. Before stating our results, one can remark that all the partial functions $\lambda_1 \mapsto f^{\lambda_1,\alpha_2}_{s,t}$, with α_2 fixed, have the same profile as illustrated in Fig. 1: the maximum flow value first increases up to a saturation step (critical capacity), and then stagnates. The partial functions $\lambda_2 \mapsto f^{\alpha_1,\lambda_2}_{s,t}$, with α_1 fixed, behave analogously.

Proposition 11. *Let $G = (V, E)$ be a network, e_1 and e_2 two different edges of E, and s and t two distinct vertices of V. Let α_1, α_2, β_1 and β_2 be values such that $0 \leq \alpha_i < \beta_i \leq \infty$, $i = 1, 2$. Then, if $\lambda_i \in [\alpha_i, \beta_i]$, $i = 1, 2$, the maximum flow value $f^{\lambda_1,\lambda_2}_{s,t}$ can be directly obtained from the four maximum flow values $f^{\alpha_1,\alpha_2}_{s,t}$, $f^{\alpha_1,\beta_2}_{s,t}$, $f^{\beta_1,\alpha_2}_{s,t}$ and $f^{\beta_1,\beta_2}_{s,t}$. The maximum flow value ($f^{\lambda_1,\lambda_2}_{s,t}$) can be computed as follows:*

$$f^{\lambda_1,\lambda_2}_{s,t} = \min \begin{pmatrix} f^{\alpha_1,\alpha_2}_{s,t} + \lambda_1 - \alpha_1 + \lambda_2 - \alpha_2, \\ f^{\alpha_1,\beta_2}_{s,t} + \lambda_1 - \alpha_1, \\ f^{\beta_1,\alpha_2}_{s,t} + \lambda_2 - \alpha_2, \\ f^{\beta_1,\beta_2}_{s,t} \end{pmatrix}. \tag{4}$$

Proof. The main point of this proof is to decompose the computation of the general maximum flow into several computations of simple maximum flows and use Proposition 5 to obtain the desired values.

Thus, let us consider that λ_2 is fixed. As noted previously, the partial function $\lambda_1 \mapsto f_{s,t}^{\lambda_1,\lambda_2}$ can be obtained if both maximum flows $f_{s,t}^{\alpha_1,\lambda_2}$ and $f_{s,t}^{\beta_1,\lambda_2}$ are known by using Proposition 5 or its closed form given in Eq. (3):

$$f_{s,t}^{\lambda_1,\lambda_2} = \min(f_{s,t}^{\alpha_1,\lambda_2} + \lambda_1 - \alpha_1, f_{s,t}^{\beta_1,\lambda_2}). \tag{5}$$

For this step, it remains to compute $f_{s,t}^{\alpha_1,\lambda_2}$ and $f_{s,t}^{\beta_1,\lambda_2}$. For the former, we consider the partial function $\lambda_2 \mapsto f_{s,t}^{\alpha_1,\lambda_2}$. Again, this function can be described using Proposition 5:

$$f_{s,t}^{\alpha_1,\lambda_2} = \min(f_{s,t}^{\alpha_1,\alpha_2} + \lambda_2 - \alpha_2, f_{s,t}^{\alpha_1,\beta_2}). \tag{6}$$

Similarly, $f_{s,t}^{\beta_1,\lambda_2}$ can be obtained from the following equation:

$$f_{s,t}^{\beta_1,\lambda_2} = \min(f_{s,t}^{\beta_1,\alpha_2} + \lambda_2 - \alpha_2, f_{s,t}^{\beta_1,\beta_2}). \tag{7}$$

Consequently, we have

$$f_{s,t}(\lambda_1, \lambda_2) = \min \begin{pmatrix} \min(f_{s,t}^{\alpha_1,\alpha_2} + \lambda_2 - \alpha_2, f_{s,t}^{\alpha_1,\beta_2}) + \lambda_1 - \alpha_1 \\ \min(f_{s,t}^{\beta_1,\alpha_2} + \lambda_2 - \alpha_2, f_{s,t}^{\beta_1,\beta_2}) \end{pmatrix}. \tag{8}$$

The previous equation simplifies to the desired result. □

As previously, the proof yields an algorithm for computing the resulting maximum flow values.

From Proposition 11, we can deduce the all pairs maximum flow proposition for two parametric capacities:

Proposition 12. *Let $G = (V, E)$ be a network, and e_1 and e_2 be two different edges of E. Then, the all pairs parametric maximum flow problem and the all pairs parametric minimum cuts problem can be solved with the computation of four cut-trees if the capacities of both edges e_1 and e_2 vary.*

Proof. This is a direct consequence of Proposition 11. We need to compute all the $f_{s,t}^{0,0}$, $f_{s,t}^{0,\infty}$, $f_{s,t}^{\infty,0}$ and $f_{s,t}^{\infty,\infty}$ for all the pairs (s, t) of vertices. The set of $f_{s,t}^{0,0}$, for all the vertices s and t can be obtained by the computation of a cut-tree considering the network G in which both edges e_1 and e_2 have been removed. The three other cut-trees can be obtained similarly considering the presence and/or the removal of each tested edge with the infinite capacity.

Once the four cut-trees are computed, all values of maximum flows can be obtained using Proposition 11. The four cut-trees provide the four extremal maximum flow values and any other maximum flow value can be obtained using Eq. (4).

The minimum cuts can be directly deduced from the maximum flow values. As in the single parametric case, the minimum cut can be taken from the cut-tree that realizes the minimum in Eq. (4). □

In the general case, more than two edges may vary. Let $e_1, e_2, \ldots, e_k$ be the selected edge capacities that will be parameterized by $\lambda_1, \lambda_2, \ldots$ and λ_k, $k \leq m$, where m is the number of edges in the network.

Theorem 13. *Let $G = (V, E)$ be a network, k be an integer, and $e_1, e_2, \ldots, e_k$ be k different edges. The all pairs maximum flow values and all pairs minimum cuts for all values of the parameters can be computed by using 2^k cut-tree computations if the capacities of the edges vary independently.*

Proof. This result can be obtained by induction on the number of parameterized edges. The initial case $k = 1$ is solved in Sections 3.2 and 3.3. The sketch of the general case strictly follows the proof of Proposition 12. The main idea is to consider as fixed one of the parameters and use the recursion hypothesis for this case, leading to 2^{k-1} maximum flow computations. Then, it remains to develop each maximum flow computation in terms of the final dimension. Thus, for each computation, two maximum flows are necessary, by using Proposition 5, leading to 2^k maximum flow computations. The same paradigm can be used for minimum cuts.

The graphs on which the cut-trees have to be computed are the variations of the initial graph where either the edges considered are removed or their capacity is set to infinity. □

4.2. Computing a Gomory–Hu cut-tree from two extremal ones

In this section, we show another application of Theorem 10. The problem is computing as efficiently as possible a new cut-tree GH^{λ} from two extremal ones, i.e., GH^{α} and GH^{β}, where $0 \leq \alpha < \lambda < \beta \leq \infty$. In Section 5, we study the same problem under weaker assumptions, i.e., when only GH^{α} is given.

We first summarize the Gusfield algorithm for computing a cut-tree. The details and proof of correctness of this algorithm can be found in [5]. Here, we call a star tree with n vertices the tree having a root, labeled 1, and $n-1$ leaves, labeled from 2 to n.

Algorithm 1 Gusfield(G)
▹ *G is a network having n vertices.*
▹ *Returns a cut-tree T of G.*
1 Compute a star tree T with n vertices, labeled from 1 to n.
2 **for** $s = 2$ to n
3 Let t be the neighbor of s in the current tree T.
4 Compute a minimum cut $C_{s,t}$ between s and t in G.
5 Change the tree by labeling the edge $[s, t]$ with $c(C_{s,t})$, and rearrange the vertices such that this new tree reflects the newly computed minimum cut, while maintaining the validity of the previously computed ones.
6 **end for**

Considering the time complexity of this algorithm, all the steps, except Step 4, are linear in the number of vertices of G. This implies the classical $(n-1)MF(n)$ time complexity for the cut-tree construction, where $MF(n)$ is the time complexity of a maximum flow (minimum cut) computation.

Proposition 14. *Let G be a network having n vertices and e a single edge of G having a parametric capacity $c(e) = \lambda$. Let α and β be two numbers such that $0 \leq \alpha < \beta \leq \infty$. Given GH^{α} and GH^{β}, we can compute GH^{λ} in $O(n^2)$, for any $\lambda \in]\alpha, \beta[$.*

Proof. In order to obtain this time complexity, we slightly modify the Gusfield algorithm. The main difference resides in the way a minimum cut in Step 4 is obtained. From applying Theorem 10, any minimum cut required by the algorithm can be computed in linear time. Since the skeleton of the algorithm remains the same as in the original Gusfield algorithm, the resulting time complexity is $O(n^2)$. □

Note that in this problem no information is recovered from the network itself.

5. Computation of cut-trees

In this section, we study the same problem as in Section 4.2. However, we consider weaker assumptions. The problem turns into the computation of a cut-tree GH^{λ} from GH^{α} and G, where $\alpha < \lambda \leq \infty$. Note that in the previous section, the construction of the new cut-tree is a consequence of the sensitivity analysis.

For a network having only one varying edge capacity, the variation of this capacity may have no influence on the maximum flow value (and thus on the minimum cuts) for many pairs of vertices. The remainder of this section tries to identify such pairs, and the consequences on the construction for a new cut-tree.

Lemma 15. *Let G be a network with n vertices and $e = [i, j]$ a single edge of G such that $c(e) = \lambda$. Let s and t be a pair of vertices of G. Let GH^{α} be a cut-tree when $c(e) = \alpha$. If the path $P_{s,t}$ in GH^{α} has no common edge with $P_{i,j}$, then $f^{\lambda}_{s,t} = f^{\alpha}_{s,t}$, $\forall \lambda > \alpha \geq 0$.*

Proof. From using the cut property of the cut-tree, there exist a minimum cut $C^{\alpha}_{s,t}$ separating s and t such that both vertices i and j ($e = [i, j]$) are on the same side of the minimum cut. Consequently, it does not contain e for $\lambda > \alpha$, and is insensitive to the variation of λ. Assume that there exists another minimum cut $C'^{\alpha}_{s,t}$ sensitive to λ. Thus, using the Max-Flow/Min-Cut Theorem and Eq. (3):

$$\exists \lambda > \alpha \quad c(C'^{\lambda}_{s,t}) = c(C'^{\alpha}_{s,t}) + \lambda - \alpha.$$

Note that both $C'^{\alpha}_{s,t}$ and $C^{\alpha}_{s,t}$ are minimum cuts. Thus, we have

$$\exists \lambda > \alpha \quad c(C'^{\lambda}_{s,t}) = c(C^{\alpha}_{s,t}) + \lambda - \alpha.$$

As noted before, $C^{\alpha}_{s,t}$, remains a cut whenever $\lambda > \alpha$, and thus

$$\begin{aligned} \exists \lambda > \alpha \quad c(C^{\alpha}_{s,t}) &\geq c(C'^{\lambda}_{s,t}) \\ &\geq c(C^{\alpha}_{s,t}) + \lambda - \alpha. \end{aligned}$$

Consequently, this is only possible for $\lambda = \alpha$. Summarizing, the only possibility for e belonging to a minimum cut is for $\lambda = \alpha$. In this case, α must be strictly positive and there exists another minimum cut that does not contain edge e. Thus, the maximum flow value is constant for $\lambda \geq \alpha$. □

Let us now summarize the Gomory and Hu algorithm (proofs and details can be found in [1]). To do this, we use the notion of *supervertex*. A supervertex is a part in a partition of the vertex set of the graph. During the construction of the cut-tree, the partition induced by the supervertices is refined at each step. In the following algorithm, we will consider graphs in which supervertices are contracted to a single vertex.

Algorithm 2 Gomory–Hu(G)

▷ *G is a network having n vertices.*

▷ *Returns a cut-tree T of G*

1 Create one supervertex with all the vertices.
2 **while** there exists one supervertex SV with more than one vertex **do**
3 Choose s and t in SV
4 Consider the network G' composed by all the supervertices except SV that has been expanded.
5 Compute a minimum cut $C'_{s,t}$ in G'.
6 Separate SV into two supervertices SV_1 and SV_2 connected by the capacity of the previous cut, such that the vertices in SV_1 are in one part of the cut and SV_2 is in the other part. Connect the other supervertices to either SV_1 or SV_2 depending on their position in $C'_{s,t}$.
7 **end while**

Note that this algorithm maintains a tree during each loop, called the intermediate cut-tree, connecting supervertices. This implies that the resulting graph is always a tree. As in Gusfield's algorithm, choices can be performed in order to obtain different cut-trees. These choices can be made at Step 3 at two different levels. First, the two vertices involved with the Max-Flow/Min-Cut computation are arbitrary within a same supervertex. Second, there may exist different minimum cuts, that may lead to different cut-trees. Finally, in order to prove the correctness of this algorithm, Gomory and Hu stated that a minimum cut obtained in the condensed graph G' in Step 5 between vertices s and t represents a minimum cut between s and t in the original graph. This implies that this algorithm computes a sequence of non-crossing cuts. From all these remarks and on the basis of Lemma 15, much information for the final computation of GH^{β} can be taken from GH^{α}.

In order to describe our final result, we need some further notation. Let G be a network having a parametric edge $e = [i, j]$. Let GH^{α} be a cut-tree when $c(e) = \alpha$. Let $P_{i,j}$ be the path connecting i and j in GH^{α}, $P_{i,j} = (x_0 = i, \ldots x_a, \ldots, x_l = j)$. Let x_a be a vertex in $P_{i,j}$, and y^b_a, $0 \leq b < k_a$, the neighbors of x_a not in $P_{i,j}$. For any y^b_a, let T^b_a be the maximal subtree of GH^{α} rooted at y^b_a not containing x_a. This situation is illustrated in Fig. 2. The set of trees $\mathcal{T}_{i,j}$ defined as above is called the (i, j) *forest decomposition of* GH^{α}.

With this definition, we obtain the following lemma.

Lemma 16. *Let G be a network having a parametric edge $e = [i, j]$. Let GH^{α} be a cut-tree when $c(e) = \alpha$. Let $\mathcal{T}_{i,j}$ be a (i, j) forest decomposition of GH^{α}. For each tree $T^b_a \in \mathcal{T}_{i,j}$, there exists a cut-tree of G with $c(e) = \lambda > \alpha$ that contains T^b_a as subtree.*

Proof. Let us consider a DFS exploration of T^b_a rooted in y^b_a, i.e., the index of y^b_a is 1. We perform the Gomory–Hu algorithm breaking the ties as follows. First, let $s_1 = x_a$ and $t_1 = y^b_a$, belonging to the same supervertex. We can compute a minimum cut between s_1 and t_1. The path P_{s_1,t_1} is reduced to the edge $[x_a, y^b_a]$ in GH^{α}; it does not have any edge in common with $P_{i,j}$. Thus, using Lemma 15, a minimum cut between s_1 and t_1 can be taken in GH^{α}. From

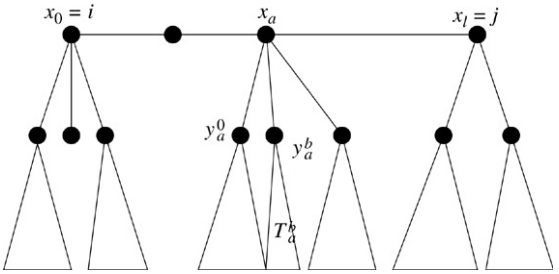

Fig. 2. The (i, j) forest decomposition of GH^α.

these choices, the resulting supervertices are (1) SV_1: all the elements of T_a^b and (2) SV_2: the other elements of the network.

Now, for the other steps, we consider s_k as the k-th vertex in the DFS order and t_k as its parent in T_a^b. As previously, we can see that they belong at step k to the same supervertex. Thus, we can perform a step in the Gomory–Hu algorithm by seeking for a minimum cut between s_k and t_k. Using Lemma 15, such a minimum cut is given by GH^α. From Step 6, we obtain an edge between two supervertices, exactly corresponding to the edge between s_k and t_k in T_a^b. Thus, step by step, we reconstruct all the structure of T_a^b.

Once all the elements of T_a^b have been considered, the whole supervertex SV_1 has been transformed into a single tree. The Gomory–Hu algorithm can continue by considering SV_2. In this latter phase, no vertex of SV_1 will be affected by the choices made for s and t and the structure of the subtree T_a^b will be preserved through the algorithm. □

In this proof, we always seek for minimum cuts between vertices that are neighbors in the original cut-tree (GH^α) and that do not belong to $P_{i,j}$. The order we have chosen for exploring the tree T_a^b ensures that this property is verified at any step of this algorithm. Note that there exist many orders for which this algorithm would have led to the reconstruction of T_a^b: the order obtained by DFS is one of them.

Theorem 17. *Let G be a network having an edge $e = [i, j]$ with parametric capacity $c(e) = \lambda$. Let GH^α be a cut-tree obtained when $c(e) = \alpha$. Let $P_{i,j}$ be the path in GH^α between i and j. For $\lambda > \alpha$ it is sufficient to compute $|P_{i,j}| - 1$ minimum cuts in G^λ in order to obtain a cut-tree GH^λ.*

Proof. This proof is based on the Gomory and Hu initial algorithm. The main point again is finding an order for exploring the graph in such a way that the interesting structures, i.e., parts of GH^α, are created first in the construction of the cut-tree. Since we show that these structures are created, we can start the algorithm in this new step, avoiding the corresponding computations.

More precisely, we define the order in which the vertices of G must be explored. On the basis of Lemma 16, we can begin the Gomory–Hu algorithm by exploring all the subtrees of the form T_a^b one by one. The final order would follow a DFS numbering of the successive subtrees.

This leads to the following state of an intermediate tree: one supervertex SV composed of all the elements of $P_{i,j}$ and all the subtrees of the form T_a^b connected to SV by y_a^b as shown in Fig. 3. Thus, we can start the Gomory–Hu algorithm at this step. Since there remain $|P_{i,j}|$ elements in the supervertex, we only need to compute $|P_{i,j}| - 1$ minimum cuts in G. □

This theorem provides an improvement on the number of maximum flow computations that have to be performed to compute the cut-trees of two networks that only differs in one single edge capacity. Using the original Gomory and Hu algorithm, we note that the maximum flow computations are made on smaller graphs than the original one. This would imply further improvement on the time complexity.

For these different algorithms used to compute cut-trees from previously known ones we can make the following remark. The study performed in this section only considers the case where capacities can increase. In the other case, the problem becomes more difficult. Indeed, in our case, we deeply use the fact that subtrees can be reused if they consider saturated flows, i.e., only when these flows will never be affected when the capacity is increasing. When the capacity is decreasing, a study *à la* Elmaghraby should be made; the number of intermediate cut-trees is then a function of the number of critical capacities between β and λ.

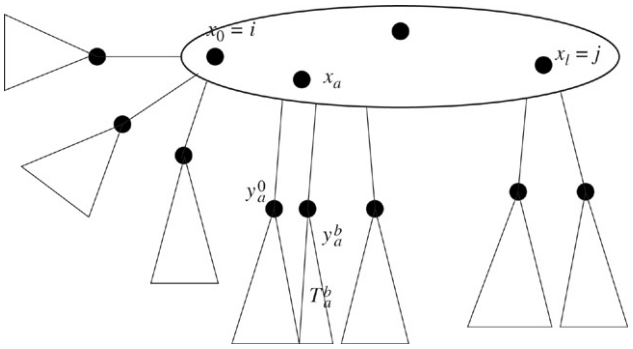

Fig. 3. State of the intermediate tree before the second phase of the algorithm.

6. Open problems

From these results, some open questions could be investigated. First, is there a polynomial algorithm for computing all pairs flow variations for any number of parametric edges with related varying capacities? Second, we have seen that the computation time for iterative Gomory–Hu cut-trees when capacities vary can be improved in many cases. What could be the impact of this improvement on the average computation time? Are there some other possible improvements using properties of the initial cut-tree?

Acknowledgments

We thank the anonymous referees for their many valuable comments that helped to improve the presentation of this paper. The third author's Postdoctoral Research was funded by the French CNRS within the French ACI Sécurité project (PRESTO) at LIMOS, UMR CNRS 6158. The fourth author's work for this article was partially supported by the European FET project CRESCCO and by the French ACI Security.

References

[1] R.E. Gomory, T.C. Hu, Multi-terminal network flows, SIAM Journal of Computing 9 (4) (1961) 551–570.
[2] M. Diallo, Réseaux de flots: Flots paramétrés et tarification, Ph.D. Thesis, Université de Versailles, France, (in French). Available at http://www.prism.uvsq.fr/~diallo, December 2003.
[3] S. Elmaghraby, Sensitivity analysis of multi-terminal network flows, Operations Research 12 (5) (1964) 680–688.
[4] E. Duarte Jr., J. Cohen, Fault tolerant routing of network management messages in the internet, in: IEEE Latin American Network Operations and Management Workshop, G. Proceedings of the 2nd Latin American Network Operations and Management Workshop, Belo Horizonte, MG, 2001, pp. 87–98.
[5] D. Gusfield, Very simple methods for all pairs network flow analysis, SIAM Journal of Computing 19 (1990) 143–155.
[6] D. Gusfield, D. Naor, Efficient algorithms for generalized cut trees, in: ACM Symposium On Discrete Algorithms, 1990, pp. 422–433.
[7] C.-P. Schnorr, Bottlenecks and edge connectivity in unsymmetrical networks, SIAM Journal of Computing 8 (2) (1979) 265–274.
[8] A. Benczúr, Counterexamples for directed and node capacitated cut-trees, SIAM Journal of Computing 24 (3) (1995) 505–510.
[9] M. Scutellà, A note on the parametric maximum flow problem and some related reoptimization issues, Annals of Operations Research (in press). Conference version appeared in INOC 2003, pp. 516–520.
[10] P. Berthomé, M. Diallo, A. Ferreira, Generalized parametric multi-terminal flows problem, in: H. Bodlaender (Ed.), Graph Theoretical Concepts in Computer Science (WG) 2003, in: Lecture Notes in Computer Science, vol. 2880, 2003, pp. 71–80.
[11] P. Tucker, T. Hu, M. Shing, Minimum cuts without path packing, Tech. Rep. CS 99-625, UCSD, June, 1999.
[12] L. Ford, D. Fulkerson, Flows in Networks, Princeton Univ. Press, Princeton, NJ, 1973.
[13] C. Chekuri, A. Goldberg, D. Karger, M. Levine, C. Stein, Experimental study of minimum cut algorithms, in: ACM Symposium on Discrete Algorithms, New Orleans, Louisiana, 1997, pp. 324–333.