EFFECTS OF CAPACITIES VARIATIONS ON MAXIMUM FLOWS, MINIMUM CUTS AND EDGE SATURATION

BARTH D / BERTHOME P / DIALLO M

Unité Mixte de Recherche 8623
CNRS-Université Paris Sud – LRI

10/2004

Rapport de Recherche N° 1395

Effects of capacities variations on maximum flows, minimum cuts and edge saturation

D. Barth[1], P. Berthomé[2] and M. Diallo[3]

[1] Laboratoire PRiSM, UMR 8144, CNRS, Université de Versailles, 45 Av. des Etats-Unis, 78035 Versailles-Cedex, France, Dominique.Barth@prism.uvsq.fr
[2] Laboratoire de Recherche en Informatique (LRI), UMR 8623, CNRS, Université Paris-Sud, 91405, Orsay-Cedex, France, Pascal.Berthome@lri.fr
[3] CNRS, LIMOS UMR 6158, Université Clermont 2, ISIMA, Campus des Cézeaux - BP 10125, 63173 Aubière CEDEX, FRANCE, diallo@isima.fr

Abstract In a recent work, we showed that the computation of the all pairs maximum flow values, when k edge capacities are allowed to vary, can be efficiently done with only 2^k Gomory-Hu cut-tree constructions. In this paper, we show the cut counterpart of the problem, i.e., with the variation of k capacities, we determine for each vertex pair, a minimum cut separating the two vertices.
Moreover, we provide two applications of the studied parametric flows and cuts problems. We first show how to reuse information of a computed cut-tree in order to construct a next one. Second, we introduce an interesting practical problem: Given an edge in an undirected network, it consists in determining the set of vertex pairs for which any maximum flow saturates the chosen edge.

Keywords: Gomory-Hu cut-tree, sensitivity analysis, parametrized capacities, algorithms.

Résumé Récemment, nous avons établi que le calcul du flot maximum multi-terminal en présence de k capacités d'arêtes paramétriques peut être effectué de manière efficace à l'aide de seulement 2^k arbres dits de Gomory-Hu. Dans cet article, nous donnons la contrepartie de ce théorème pour les coupes minimum. En d'autres termes, nous déterminons en présence de k arêtes paramétriques une coupe minimum séparant toute paire de sommets.
De plus, nous apportons deux applications à cette étude sur les flots et coupes paramétrés. Nous montrons tout d'abord comment utiliser les informations contenues dans un arbre de Gomory-Hu afin d'en calculer un second pour un réseau proche. Dans un second temps, nous introduisons un nouveau problème pratique: étant donnée une arête dans un réseau, nous recherchons l'ensemble des paires source-puits pour lesquelles n'importe quel flot maximum saturera cette arête.

Mots-Clés: Arbres de Gomory-Hu, analyse de sensitivité, capacités paramétriques, algorithmique.

1 Introduction

Given an undirected network, the multi-terminal network flows analysis consists in determining the all pairs maximum flow values. Since its introduction, in 1961, by Gomory and Hu [9], many applications and variants of the problem were presented in the literature. In [5], Diallo presented an exhaustive survey on the subject. The multi-terminal network flows problem has many known applications in the fields of transports, energy and telecommunications (see for example [4–6] and references therein). Most of the algorithms proposed to solve this problem use the concept of cut-trees. Given an undirected network $G = (V, E)$ with edge capacities, a cut-tree of G is an edge-weighted tree with vertex set V and in which, for any pair of vertices u and v, the minimum weight of an edge on the path between u and v in the tree is the maximum flow value between u and v in G. Moreover, the former edge with minimal weight between u and v in the tree decomposes the vertex set into two subsets forming a minimal cut between u and v in G. The construction of such a cut-tree costs only $(n - 1)$ maximum flow computations.

In this paper, we focus on some sensitivity analysis aspects of the multi-terminal flows problem, i.e., given an edge e, the impact of e on the all pairs maximum flows and the effect of the edge capacity variation on their values (we talk about *parametric multi-terminal network flows*). In 1964, Elmaghraby [6] was the first to introduce such a sensitivity analysis. He supposes that a single edge capacity is allowed to vary and examines the impact of this variation on the all pairs maximum flow values. However, Elmaghraby's method needs to compute as many cut-trees as the number of critical capacities. A critical capacity (or breakpoint) with respect to a given vertex pair is the value of the parameterized capacity for which the maximum flow value between the vertices of the pair stops or begins to vary with regard to the parametric capacity. However, Diallo [5] showed that Elmaghraby's method does not work in all cases (a counter-example and an improvement are there given). Note that parametric multi-terminal network flows problem reflects an application of problems including link breakdown, capacity increase and investments in expansion [5, 6].

This paper tackles different points in the network flow theory.

– In a recent work [2], we showed a very efficient method to compute the all-pairs maximum flow values for all possible capacities of k edges, by constructing 2^k cut-trees. In the same context, the problem of finding, for each vertex pair, a minimum cut separating the two vertices remained open. Recall that, in the constant capacity case, a cut-tree reflects at least a minimum cut separating each two vertices of the network.
  In this paper, we provide an efficient way to determine a minimum cut for any pair of vertices, when k capacities are allowed to vary.

– Solving the above problem, it appears that two cut-trees are needed. Both of them are related since they consider similar networks. The second aim of this paper is to show how to compute a new cut-tree, by using the most as possible information in other cut-trees.

– Finally, we study the *All pairs Saturating Flows*, called *ASF* problem [5]: given an undirected network and an edge e of the network, the problem is to determine the set $ASF[e]$ of vertex pairs for which any maximum flow computed between the two vertices saturates e.
  As for the *ASF* problem, we may cite interesting applications such as adding or removing edge problems [14] and the building evacuation problem [8,11,12] that is a particular case of the dynamic flow problem: Given the minimum evacuation time, the problem consists in detecting all the bottlenecks that may cause delay. Such bottlenecks correspond exactly to edges (saturated by maximum flows) belonging to minimum cuts.

The remainder of this paper is organized as follows. In Section 2, we give the basic definitions and briefly describe the results of our previous work. Section 3 is devoted to the computation of minimum cuts when capacities are varying in the network. In Section 4, we address the problem of constructing cut-trees by reusing information of previous constructed cut-trees. In Section 5, we introduce the *ASF* problem. We close the paper with a conclusion and some perspectives.

2 Previous work on parametric multi-terminal network flows

In this section, we provide some basic definitions and briefly describe the main results of our previous work [2]. Throughout this paper, we assume that the reader is familiar with general concepts of graph theory and network flows. For example, we refer to [1, 7, 13]. In the remaining of the paper, unless specified "network" stands for "undirected network".

2.1 Definitions

Let $G = (V, E)$ be an undirected connected network with vertex set V and edge set E. Throughout this paper, we denote n as $|V|$ and m as $|E|$. To 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 [7]. Thus, we denote by $\overrightarrow{f}_{s,t}$ a maximum flow between s and t in G, and by $f_{s,t} = f_{t,s}$ the value of $\overrightarrow{f}_{s,t}$. 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).

A proper subset X of V ($\emptyset \subsetneq X \subsetneq V$) is called a *cut separating two vertices u and v* if $u \in X$ and $v \in V \setminus X$. The capacity of a cut X is defined as $c(X) = \sum_{x \in X, y \in V \setminus X, [x,y] \in E} c[x,y]$. Such an edge $[x,y]$ *belongs* to the cut X.

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

Definition 1 (Cut-tree). *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' verifying the two following properties:*

a) Equivalent flow tree: *for any pair of vertices s and t, $f_{s,t}$ in G is equal to $f_{s,t}$ in T, i.e., the smallest capacity (with c') of the edges on the path between s and t in T;*

b) Cut property: *a minimum cut $C_{s,t}$ in T is also a minimum cut in G.*

As shown in [9], $(n-1)$ minimum cut computations are sufficient to construct 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 [9] and in [10]. An experimental study of minimum cut algorithms and a comparison of algorithms producing cut-trees are provided in [3].

Parametric multi-terminal flow problems: 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 capacities variations.

Throughout this paper, when one capacity $c(e)$ is allowed to vary ($c(e) = \lambda$), we will note $f^{\lambda}_{s,t}$ (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 Previous results

Effects of a single varying capacity: Let G be a network in which only a single edge $e = [i,j]$ possesses a capacity $\lambda \geq 0$. Note that, when $\{s,t\} \neq \{i,j\}$, $\lim_{\lambda \to \infty} f^{\lambda}_{s,t}$ is finite; this value is denoted hereafter $f^{\infty}_{s,t}$. We can then consider that the capacity $c(e)$ can be set to infinity. In practice, $c(e) = \infty$ can be obtained by setting $c(e)$ to the sum of the capacities of th edges adjacent to e.

In [2], we showed that the maximum flow value $f^{\lambda}_{i,j}$ between the extremities of e is continuously increasing with λ, and for the maximum flow value $f^{\lambda}_{p,q}$ between any two vertices $\{p,q\} \neq \{i,j\}$, there are two possibilities, either $f^{\lambda}_{p,q}$ is always constant or it is a piecewise-linear function of λ: it increases until $\lambda = \lambda^*_{p,q}$, and then becomes constant as follows. The value $\lambda^*_{p,q}$ is, called *critical capacity* for the pair $\{p,q\}$, $\{p,q\} \neq \{i,j\}$, is given by $f^{\infty}_{p,q} - f^{0}_{p,q}$. We have:

$$f^{\lambda}_{p,q} = \begin{cases} f^0_{p,q} + \lambda & \text{if } \lambda < \lambda^*_{p,q} \\ f^{\infty}_{p,q} & \text{otherwise} \end{cases} \tag{1}$$

Thus, for any vertex pair $\{p, q\}$ and any value of λ, follows the general form of $f^{\lambda}_{p,q}$.

$$f^{\lambda}_{p,q} = \min(f^0_{p,q} + \lambda, f^{\infty}_{p,q}). \tag{2}$$

Let GH^{λ} denote a cut-tree when $c(e) = \lambda$, $0 \leq \lambda \leq \infty$.

Theorem 1. *[2] Let G be a network with $e = [i, j] \in G$ and $c(e) = \lambda$. Then, given GH^0 and GH^{∞}, the maximum flow value $f^{\lambda}_{s,t}$ between any s and t for any value of $\lambda \in [0, \infty]$ can be computed in $O(n)$ time.*

The above theorem states that solving the sensitivity analysis of the multi-terminal network flows (the all pairs network flows) needs only two cut-tree constructions and some arithmetic computations. What improves considerably the Elmaghraby's method [6] that computes a cut-tree for each critical value. Considering the time complexity, the computation of the critical capacities can be performed in time $O(n^3)$, i.e., $O(n)$ time per pair once both cut-trees are known.

Effects of multiple varying capacities: A generalization of the previous theorem is the following.

Theorem 2. *[2] Let $G = (V, E)$ be a network with k different edges e_1, e_2, $\dots$, e_k with respective capacities λ_1, λ_2, $\dots$, λ_k. The all pairs maximum flow values, with regard to all parameters values, can be obtained by constructing only 2^k cut-trees.*

For the particular case of $k = 2$, as regard to the maximum flow value for a single vertex pair $\{s, t\}$, we have shown the following.

Theorem 3. *[2] Let $G = (V, E)$ be a network, e_1 and e_2 edges of E with respective capacities λ_1 and λ_2. Consider s and t two vertices of V. Then, the maximum flow value ($f^{\lambda_1,\lambda_2}_{s,t}$) can be directly obtained from the four maximum flow values $f^{0,0}_{s,t}$, $f^{0,\infty}_{s,t}$, $f^{\infty,0}_{s,t}$ and $f^{\infty,\infty}_{s,t}$. The maximum flow value ($f^{\lambda_1,\lambda_2}_{s,t}$) can be computed as follows:*

$$f^{\lambda_1,\lambda_2}_{s,t} = \min(f^{0,0}_{s,t} + \lambda_1 + \lambda_2, f^{0,\infty}_{s,t} + \lambda_1, f^{\infty,0}_{s,t} + \lambda_2, f^{\infty,\infty}_{s,t}). \tag{3}$$

In this section, we briefly recalled, based on cut-trees, how to compute the all pairs maximum flow values with regard to capacities varying in an undirected network. One can notice that in the presented results, we do not show how one could, in such a situation, obtain a minimum cut for each of the $\frac{n(n-1)}{2}$ vertex pairs of G in at most the same complexity as the one used for the maximum flow values. In the next section, we show how to determine for each vertex pair a minimum cut.

3 Computing minimum cuts with regard to varying capacities

This section is devoted to the computation of minimum cuts in the presence of varying capacities. We highlight a minimum cut for each vertex pair.

3.1 Single parametric capacity in the network

Extending the previous results Let $e \in E$ be the single edge having a parametric capacity $c(e) = \lambda \in [0, \infty]$ in the network G. Given two distinct values α and β of $c(e)$, $\alpha < \beta$. Equation 2 that gives, for an arbitrary vertex pair $\{s, t\}$, the maximum flow value $f^{\lambda}_{s,t}$ can be extended as follows.

$$\forall \lambda \in [\alpha, \beta] \quad f^{\lambda}_{s,t} = \min(f^{\alpha}_{s,t} + \lambda - \alpha, f^{\beta}_{s,t}). \tag{4}$$

Consequently, Theorem 1 receives the following extension.

Theorem 4. *Let G be a network with $e = [i, j] \in E$ and $C(e) = \lambda$. Given two distinct positive values α and β, $\alpha < \beta$, and cut-trees GH^{α} and GH^{β}, $O(n)$ arithmetic computations are sufficient in order to determine $f^{\lambda}_{s,t}$, where s and t vertices and $\lambda \in [\alpha, \beta]$.*

As far as the computation of minimum cuts is concerned, follows the first result.

Lemma 1. *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 $0 \leq \alpha < \lambda^*_{s,t} < \beta$. 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 $c(e) = \lambda \geq \lambda^*_{s,t}$. Considering, Equation 1, 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 $c(e) \geq \lambda^*_{s,t}$.

Let consider now the first case, i.e., $\lambda \leq \lambda^*_{s,t}$. From Equation 1, 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

$$c(C^{\alpha}_{s,t}) \geq f^{\lambda}_{s,t}$$

By Equation 1, we have:

$$f^{\alpha}_{s,t} \geq f^{\alpha}_{s,t} + \lambda - \alpha$$

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

$$c(C^{\alpha}_{s,t}) = \left(\sum_{x \in X, y \in V \setminus X, [x,y] \in E \setminus \{e\}} c(x,y) \right) + \lambda$$

From Equation 4, 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 corollary.

Corollary 1. *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 minimum cuts 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 1. Three cases are to be considered based on the position of the critical capacity $\lambda^{*}_{s,t}$ compared to α and β.

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

□

Theorem 5. *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 is direct application of Theorem 4 and Corollary 1. One can note that the exploration of the cut trees is linear in their size. □

3.2 Multiple parametric capacities

In this section, we generalize the previous result to more than one parametric capacity. This generalization follows the same idea as in the maximum flows setting.

Theorem 6. *Let $G = (V, E)$ be a network and e_1, e_2 two edges of G with respective parametric capacities λ_1, λ_2. The construction of 4 cut-trees is sufficient to determine, for any two vertices and any parameter value, a minimum cut that separates them in linear time.*

Sketch of Proof. The key is to extend Lemma 1 to the two dimensional case. Let s, t and λ_1, λ_2 fixed. From Lemma 1, we know that a minimum cut between s and t can be obtained by considering one of the two minimum cuts $C_{s,t}^{\lambda_1,0}$ and $C_{s,t}^{\lambda_1,\infty}$ (obtained when e_2 is removed and when its capacity is set to ∞). The choice depends on the position of λ_2 considering the critical value $\lambda_2^* = f_{s,t}^{\lambda_1,\infty} - f_{s,t}^{\lambda_1,0}$ as follows:

1. $\lambda_2 \leq \lambda_2^*$: use $C_{s,t}^{\lambda_1,0}$. It corresponds to the case where the minimum in Equation 2 is obtained by the term in λ_2;
2. $\lambda_2 \geq \lambda_2^*$: use $C_{s,t}^{\lambda_1,\infty}$.

Now, these two minimum cuts can be obtained by applying Lemma 1 twice. Consider first that we are in the first case in the above enumeration. Then, by applying the same paradigm, $C_{s,t}^{0,0}$ will be the desired cut when the term in $\lambda_1 + \lambda_2$ minimizes Equation 3.

The same work can be done considering the other values of λ_1 and λ_2.

To summarize, a minimum cut between s and t can be obtained only by considering the minimum cuts and maximum flow values, for the extremal values of the parameters.

In order to conclude this proof, the 4 extremal Gomory-Hu cut trees determine all the maximum flow values and all the minimum cuts (at least one per s and t). □

In Fig. 1, we show the typical behavior of minimum cuts influence zones. In each zone, a minimum cut is determined by the extremal one given in one of the Gomory-Hu cut trees. At the intersection lines, both cuts can be used.

Corollary 2. *Let $G = (V, E)$ be a network and e_1, e_2, ..., e_k be edges of G with respective parametric capacities λ_1, λ_2, ..., λ_k. The construction of 2^k cut-trees is sufficient to determine, for any two vertices, a minimum cut that separates them in $O(2^k n)$ time.*

Proof. We follow the same framework as for Theorem 2, i.e., an induction on the number of parametric capacities. The basic cases are given by Theorems 5 and 6. The proof of Theorem 6 provides the way to derive the result. □

For the sake of simplicity, in the multiple parametric settings, we consider that the parameters vary from 0 to ∞. The extension to other bounds is straightforward but complicates a lot the notations.

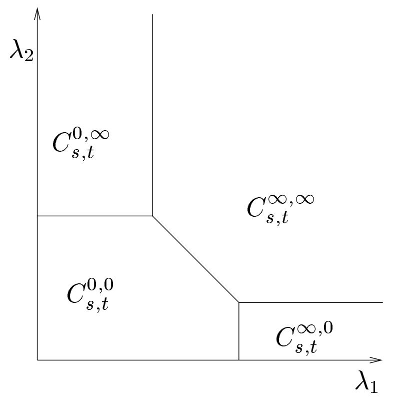

Figure 1. Influence zone of each extremal minimum cut

4 Efficiently recovering other Gomory-Hu cut trees

In this section, we show an application of Theorem 5. The problem studied here is divided into two parts. First, we want to construct as efficiently as possible a new cut-tree GH^λ from two extremal ones, *i.e.*, GH^α and GH^β, where $0 \leq \alpha < \lambda < \beta \leq \infty$. Second, we study the information of GH^α that can be efficiently used to compute GH^λ, $\lambda \in [\alpha, \infty]$.

This second part is important since it leads to average improvement on the construction of the sequence of cut-trees needed for the sensitivity analysis in Theorem 6 and Corollary 2. It also introduces concepts that will be developed in Section 5.

4.1 Computing a cut-tree from two initial cut-trees

In this section, we assume that two cut-trees have already been constructed for two different values of the parametric capacity. The goal is to efficiently construct a new cut-tree for another value of the parametric capacity.

To do so, we need to recall the guidelines of Gusfield algorithm to compute a cut-tree. The details and correctness of this algorithm can be found in [10]. Here, we call a star tree with n vertices the tree having $n-1$ leaves and a single root labeled 1.

Algorithm 1.1: **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 ver-
       tices 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, but Step 4, are linear in the number of vertices of G. This imply the classical $(n-1)MF(n)$ time complexity for the cut-tree construction, where $MF(n)$ is the time complexity of the maximum flow (minimum cut) computation.

Enlightened with the results of the previous section, we can state the following lemma.

Lemma 2. *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 $O(n^2)$ time complexity, for any $\lambda \in]\alpha, \beta[$.*

Proof. In order to obtain this time complexity, we slightly modify Gusfield algorithm. The main difference resides in the way a minimum cut in Step 4 is obtained. By applying Theorem 5, 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 $O(n^2)$. □

4.2 Computing a cut-tree from one initial cut-tree

In this subsection, we assume that only one cut-tree is known. The goal is the same as before, i.e., compute a new cut-tree for a different value of the edge capacity.

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

Lemma 3. *Let G be a network having n vertices and $e = [i, j]$ a single edge of G having a parametric capacity $c(e) = \lambda$. Let s and t be two vertices of G. If the path $P_{s,t}$ in GH^{α} has no common edge with $P_{i,j}$, then $f_{s,t}(\lambda) = f_{s,t}(\alpha)$, $\forall \lambda > \alpha \geq 0$.*

Proof. By using the cut property of the cut-tree, there exists a minimum cut $C^{\alpha}_{s,t}$ separating s and t such that both vertices i and j ($e = [i, j]$) are in 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 Equation 4:

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

Note that both $C'^{\alpha}_{s,t}$ and $C^{\alpha}_{s,t}$ are min 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 $\lambda > \alpha$, thus

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

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

Let G be a network with an edge $e = [i, j]$ having parametric capacity λ. Let GH^{α} be a cut-tree when $c(e) = \alpha$. Let $P_{i,j}$ be the path in GH^{α} connecting i and j such that $P_{i,j} = (x_0 = i, x_1, \ldots, x_l = j)$. Let $P^{-}_{i,j}$ be the forest obtained from GH^{α} by removing the edges of $P_{i,j}$. Let T_a be the tree of $P^{-}_{i,j}$ containing x_a. Fig. 2 illustrates all these definitions. Using these definitions, Lemma 3 says that the maximum flow value between s and t within the same subtree T_a is insensitive to the variation of the capacity of e for $\lambda \geq \alpha$.

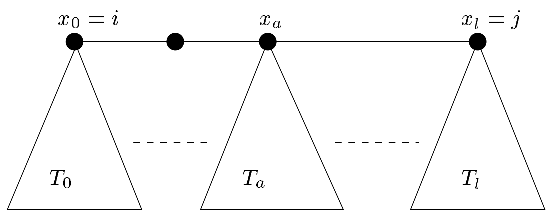

Figure 2. Identifying subtrees in a cut-tree

Lemma 4. *Using the above definitions, given GH^{α}, for each a, the construction of GH^{λ}, $\lambda > \alpha$ can be performed in $n - n_a$ minimum cut computations, where n_a is the number of vertices of T_a.*

Proof. Consider Gusfield Algorithm, and especially Step 5 in which the intermediate tree is rebuilt in order to take into account the newly discovered minimum cut. Using Lemma 3, all the minimum cuts defined in GH^{α} (in T_a) remain valid in G whatever the value of $\lambda > \alpha$ is, when the sources and sinks of the maximum flows are taken within T_a.

We must note that the correctness of Gusfield algorithm does not require a specific labeling of the vertices. Two different permutations of the labels may lead to two different cut trees. However, both trees will preserve the fundamental properties of the cut-tree, given in Definition 1.

The idea now is to find an order for examining all the vertices such that the structure of T_a is maintained. Let π be a permutation of $[1..n]$ such that $\pi^{-1}(1), \ldots \pi^{-1}(n_a)$ is a DFS exploration of T_a.

Then, using Gusfield algorithm will start by reconstructing exactly T_a, since all the minimum cuts computed in the first $n_a - 1$ steps are given by GH^{α}. Thus, all these computations can be replaced by the assignment of T_a. Afterwards, the usual Gusfield algorithm can proceed. □

Using this paradigm, we can go further in preserving maximum flow computations. Consider now a step in Gusfield algorithm in which a vertex x_b in the path $P_{i,j}$ has in its neighborhood in the intermediate tree all the vertices of its pending subtree T_b in GH^α. Using the same argument as before, we can replace all these vertices by the whole subtree T_b, avoiding again $n_b - 1$ minimum cut computations. The trick here is to define the permutation of the vertices at running time in such a way that after having examined x_b, we will consider all the vertices in T_b in a DFS order.

From these remarks, we can derive a new algorithm that efficiently computes a second cut-tree from an initial one.

Algorithm 1.2: **Recovering-Gusfield(G,** e, GH^α, λ**)**

```
    ▷ $G$ is a network with $n$ vertices, with an edge $e = [i,j]$ having parametric capacity.
    ▷ $GH^\alpha$ is a cut-tree of $G$ when $\lambda = \alpha$; $P_{i,j}$ is the path in $GH^\alpha$ from $i$ to $j$.
    ▷ $\lambda$ is the value of the parameter for which we want to compute $GH^\lambda$, $\lambda > \alpha$.
1   Let $\mathcal{X} = \{x_0 = i, x_1 \ldots, x_l = j\}$ and $\mathcal{Y} = \{1, 2, \ldots, n\}$
2   For each $x_a \in \mathcal{X}$, compute $T_a$, and let $a$ be such that $T_a$ has the greatest size.
3   Compute $T$ a rooted tree in $x_a$ having $n$ vertices such that $T_a$ is a subtree and all
    the other vertices are leaves connected to $x_a$.
4   $\mathcal{X} \leftarrow \mathcal{X} \setminus \{x_a\}$ and $\mathcal{Y} \leftarrow \mathcal{Y} \setminus V(T_a)$
5   while the whole cut-tree is not constructed ($\mathcal{Y} \neq \emptyset$) do
6       if there exists a vertex $x_b \in P_{i,j}$ such that all the vertices in $T_b$ are connected
        to $x_b$ then
7           Replace in $T$ all these vertices by $T_b$
8           $\mathcal{Y} \leftarrow \mathcal{Y} \setminus V(T_b)$
9       else
10          if $\mathcal{X} \neq \emptyset$ then let $s \in \mathcal{X}$ else let $s \in \mathcal{Y}$ end if
11          Let $t$ be the neighbor of $s$ in the current tree $T$
12          Compute a minimum cut $C_{s,t}$ between $s$ and $t$ in $G$.
13          Update 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
            as in Gusfield initial algorithm (Algorithm 1.1).
14          $\mathcal{X} \leftarrow \mathcal{X} \setminus \{s\}$ and $\mathcal{Y} \leftarrow \mathcal{Y} \setminus \{s\}$
15      end if
16  end while
```

The correctness of this new algorithm is a direct consequence of Lemma 4 and its extension. It is obtained by computing an equivalent ordering of the vertices that will be used by Gusfield algorithm.

Lemma 5. *Algorithm 1.2 constructs a cut-tree using at least* $|P_{i,j}| - 1$ *computations of minimum cuts in* G *and at most* $n - n_a$ *such computations, where* n_a *is the size of the largest subtree in the forest* $P_{i,j}^-$.

Proof. The upper bound is given by Lemma 4. For the lower bound, we should note that for any s and t in $P_{i,j}$, the cut $C_{s,t}$ contains e for $0 < \lambda < \lambda^*_{s,t}$. Thus, since the final cut does not contain e anymore (especially, when $\lambda = \infty$), all these cuts should be recomputed. This gives a core of the cut tree computation concerning $|P_{i,j}|$ elements, inducing $|P_{i,j}|-1$ minimum cut computations in the final graph. □

This lemma shows the impact of the first cut-tree in the construction of the second one. The drawback of this method is to give only a lower bound, since there is no guaranty that all the subtrees of the form T_a can be taken into account in Step 7. In the following subsection, we fill this gap by considering the other well-known algorithm for computing cut-trees, the procedure given by Gomory and Hu themselves [9].

4.3 Complexity of the recovering process

Using the structure of the algorithm due to Gomory and Hu to compute a cut-tree [9], we show that the lower bound given in Lemma 5 is also the upper bound, reducing the average time complexity of the problem discussed in this section.

Let first recall the framework of Gomory and Hu algorithm. For this, we use the notion of supernode. A supernode is simply a group of vertices. In the following algorithm, we will consider graphs in which supernodes are contracted in a single vertex.

Algorithm 1.3: **Gomory-Hu(G)**

```
    ▷ G is a network having n vertices.
    ▷ Returns a cut-tree T of G
1   Create one supernode with all the vertices.
2   while there exists one supernode $SN$ with more than one vertex do
3       Choose $s$ and $t$ in $SN$
4       Consider the network $G'$ composed by all the supernodes except $SN$ that has
        been expanded.
5       Compute a minimum cut $C'_{s,t}$ in $G'$.
6       Separate $SN$ into two supernodes $SN_1$ and $SN_2$ connected by the capacity of
        the previous cut, such that the vertices in $SN_1$ are in one part one the cut and
        $SN_2$ is in the other part. Connect the other supernodes to either $SN_1$ or $SN_2$
        depending on their position in $C'_{s,t}$.
7   end while
```

Note that this algorithm maintains during each loop a tree, called intermediate cut-tree, composed by supernodes. 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 supernode. Second, it may exists 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. It implies that this algorithm computes a sequence of non-crossing cuts. From all these remarks and based on Lemma 3, many information for the final Gomory-Hu computation can be taken from GH^{α}. This can be summarized in the following lemma.

Lemma 6. *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}. Let x_a be a vertex in $P_{i,j}$. Assume that there exists y_a^b a vertex connected to x_a not in $P_{i,j}$. Let T_a^b be the maximal subtree of GH^{α} rooted at y_a^b not containing x_a. Then, there exists a cut-tree of G with $c(e) = \lambda > \alpha$ that contains T_a^b as subtree.*

Proof. Fig. 3 illustrates the subtree decomposition of GH^{α}. Let us consider a DFS exploration of T_a^b rooted in y_b, i.e., the index of y_b is 1. We then perform Gomory-Hu algorithm in the following order. First, let $s_1 = x_a$ and $t_1 = y_a^b$, belonging to the same supernode. We can compute a minimum cut between s_1 and t_1. Using Lemma 3, this minimum cut can be taken in GH^{α}. From these choices, the resulting supernodes are (1) SN_1: all the elements of T_a^b and (2) SN_2: the other elements of the network.

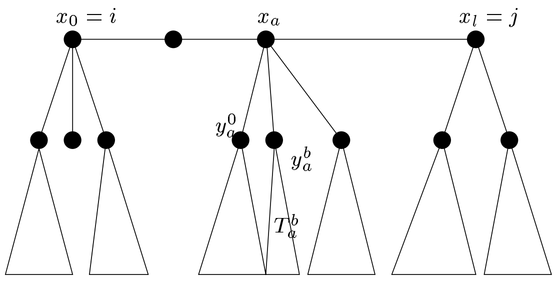

Figure 3. Two-level decomposition of a Gomory-Hu cut tree

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 supernode. Thus, we can perform a step in Gomory-Hu algorithm by seeking for a minimum cut between s_k and t_k. Using Lemma 3, such a minimum cut is given by GH^{α}. From Step 6, we obtain an edge between two supernodes, 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 supernode SN_1 has been transformed into a single tree. The Gomory-Hu algorithm can continue by considering SN_2. In this latter phase, no vertex of SN_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. □

Note that this proof can be performed whenever all the vertices of T_a^b remain in the same supernode as x_a. This remark is the basis of the proof of the following theorem.

Theorem 7. *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 if GH^{α} between i and j. For $\lambda > \alpha$ it is sufficient to compute $|P_{i,j}| - 1$ minimum cuts in order to obtain a cut-tree GH^{λ}.*

Proof. This proof is based on Gomory and Hu initial algorithm. The main point again is to find 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 many computations.

Let us go into details and define the order in which the vertices of G must be explored. Based on Lemma 6, we can begin the Gomory Hu algorithm by exploring all the subtrees of the form T_a^b. The final order would follow a DFS numbering of the successive subtrees.

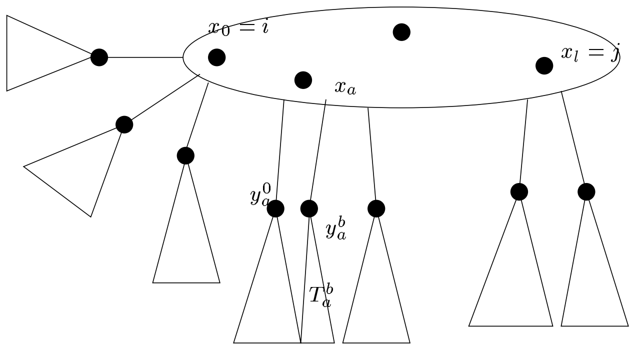

Figure 4. State of the intermediate tree before the second phase of the algorithm

This leads to the following state of intermediate tree: one supernode SN composed by all the elements of $P_{i,j}$ and all the subtrees of the form T_a^b connected to SN by y_a^b as shown in Fig. 4. Thus, we can start Gomory-Hu algorithm at this step. Since it remains $|P_{i,j}|$ element in the supernode, we only need to compute $|P_{i,j}| - 1$ minimum cuts. This new bound matches the lower bound found in Lemma 5. □

This theorem provides a great improvement on the average time complexity of computing two 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.

5 The ASF Problem

In some practical problems, determining bottlenecks is relevant. For instance, the evacuation problem introduced in Section 1 is one of them. When a flow goes from its source to its sink, the saturated edges are bottlenecks for this flow. In this section, we are interested in a particular application. Given an edge in an undirected network, we seek the set of all vertex pairs for which any maximum flow saturates the edge. Such a set may help decision makers to know, for instance, in telecommunications, the communications that would take benefit of a Quality of Service improvement on the edge. Another relevant information is that the set gives exactly the vertex pairs that would suffer from the edge removal. For more applications, we refer to [5].

Definition 2. ASF *Let $G = (V, E)$ be network and e an edge of G. The set of the* all pairs Always Saturating Flows *for e is the set of vertex pairs $\{s, t\}$ for which any maximum flow $\overrightarrow{f}_{s,t}$ saturates e. By denoting such a set $ASF[e]$, formally,*

$$\mathbf{A}SF[e] = \{\{s, t\} \in V2 \mid \forall \overrightarrow{f}_{st}, \quad \overrightarrow{f}_{st}(e) = c(e)\}.$$

The objective in this section is to show that cut-trees can be used to solve the problem. Let $G = (V, E)$ be a network with constant edge capacities, and e an edge of G with capacity $c(e) = c_0$. In the following, we show that computing a cut-tree with respect to G does not allow to determine exactly the set $ASF[e]$. But, if the capacity $c(e)$ of e is slightly decreased, assume $c(e) = \mu_0 < c_0$, where μ_0 is fixed, and let G_{μ_0} be the new network, then, computing cut-tree for G_{μ_0} and making a pairwise comparison of the maximum flow values, determines $ASF[e]$. The set of vertex pairs for which the maximum flow value decreases with respect to the decrease on $c(e)$ is ASF.

5.1 Towards $ASF[e]$ approximations

Let us consider $e = [i, j] \in E$ as the investigated edge in the network $G = (V, E)$. With respect to the extremities of e, define $P_{i,j}$ as the unique path with end points i and j in a cut-tree T of G. Recall that from the cut property of a cut-tree, any edge belonging to $P_{i,j}$ reflects a minimum cut in G that contains e, since any edge removal in $P_{i,j}$ separates i and j in T and thus in G.

Lemma 7 stands for the relationship between $P_{i,j}$ and $ASF[e]$.

Lemma 7. *Let $G = (V, E)$ be a network and $e = [i, j] \in E$. Consider a cut-tree T of G and the path $P_{i,j}$.*

$$P_{i,j} \times P_{i,j} \subset ASF[e].$$

Proof. Let $P_{i,j} = (v_0 = i, v_1, \ldots, v_k = j)$ as shown in Fig. 5. Consider two arbitrary vertices s and t in $P_{i,j}$, thus $s = v_a$ and $t = v_b$ for some a and b between 0 and k.

Let $[x, y]$ be an edge labeled with the minimum weight in $P_{s,t}$. By definition of the cut-tree, $[x, y]$ reflects a minimum cut $(X, \bar{X})$ in G that separates s and t, with $s \in X$. The removal of $[x, y]$ from T defines X as the set of all the vertices that are on the same connected component as x in T. Thus, i belongs to X and j to $\bar{X}$.

Consequently, the edge $[i, j]$ is in a minimum cut that separates s and t. From the Max-Flow/Min-Cut theorem [7], the edge $[i, j]$ is saturated by any maximum flow between s and t. Then, $[s, t]$ is $\in ASF[e]$. □

Lemma 8 extends the results of Lemma 7 to a larger subset of $ASF[e]$.

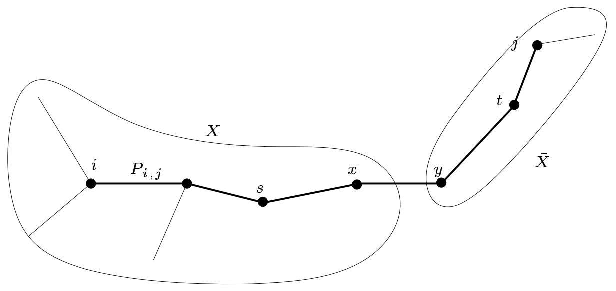

Figure 5. A cut-tree T of G and Lemma 7.

Lemma 8. *Let $e = [i, j] \in E$ be an edge of $G = (V, E)$, T a cut-tree of G and $P_{i,j}$ as before. For any two vertices s and t in G, if an edge labeled with the minimal weight in the unique path $P_{s,t}$ between s and t in T belongs also to the path $P_{i,j}$, then $[s, t]$ is in $ASF[e]$.*

Proof. This proof is based on the same idea as the previous one. By hypothesis, suppose that an edge $[x, y]$ with the minimum weight on $P_{s,t}$ lies on $P_{s,t} \cap P_{i,j}$ as shown in Fig. 6.

Then, using the definition and properties of a cut tree, the edge $[x, y]$ reflects a minimum cut (X) in G that separates s and t. Thus, this minimum cut clearly separates i and j with $i \in X$ and $j \in \bar{X} = V \setminus X$.

Consequently, using the Max-Flow/Min-Cut theorem [7], any maximum flow between s and t in G saturates $e = [i, j]$, what implies $[s, t] \in ASF[e]$. □

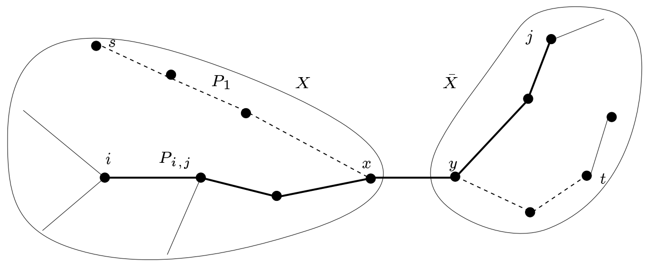

Figure 6. A cut tree and Lemma 8.

Using only the information provided by the Lemma does not always allow to explicit the set $ASF[e]$. For instance, let in Fig. 7, the top simple path of length 3, with capacities equal to 2, be an arbitrary cut-tree. Such a cut-tree may be constructed from the two networks pointed by the former path: one is the network (T) being the path itself and the other is the network (S) represented by the undirected cycle of length four with all capacities equal to 1.

In both networks S and T, let $e = [3, 4]$ be the investigated edge. It is clear from the definition of $ASF[e]$ that:

1. in T, $ASF[e] = \{(1, 4), (2, 4), (3, 4)\}$
2. in S, $ASF[e] = \{(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)\}$.

But, if we use only the information provided by Lemma 8, we would get the following sets:

– in T, the set is $\{\{1, 4\}, \{2, 4\}, \{3, 4\}\}$, what would correspond exactly to the set $ASF[e]$ described in Point 1,
– in S, the set is $\{\{1, 4\}, \{2, 4\}, \{3, 4\}\}$, what does not correspond to the set $ASF[e]$ described in Point 2.

In seeking to determine the exact set $ASF[e]$ with respect to the network S, we arise to Lemma 9.

Lemma 9. *Let $G = (V, E)$ be a network, $e = [i, j]$ an edge in E and T a cut-tree of G. Define w as the minimum edge weight on the unique path $P_{i,j}$ between i and j in T. Let K be the subtree of T containing i and j, and resulting from the removal of all edges of T labeled with a weight strictly less than w. Then,*

$$ASF[e] \subseteq V(K) \times V(K),$$

where, $V(K)$ is the set of all vertex of V that are in K.

Proof. Assume that there exists a pair s, t in $ASF[e]$ that does not belong to $V(K) \times V(K)$. Then, a minimum cut that separates s and t will be such that i and j belongs to its same component. Thus, edge e does not belong to any minimum cut separating s and t. What implies the existence of a maximum flow $\overrightarrow{f}_{s,t}$ such that $|\overrightarrow{f}_{s,t}(e)| < c(e)$. But, this is a contradiction with our hypothesis and also with the definition of ASF. $\square$

With Lemma 8 and 9, we bound the set ASF. Given any network G and its edge e, we showed above that using only the information provided by Lemma 8 is not sufficient to determine $ASF[e]$. As for Lemma 9, we show that using only the information provided by the Lemma is also not sufficient to determine $ASF[e]$.

For instance, consider the same networks S and T of Fig. 7 and let in both networks, the edge $e = [3, 4]$ be the investigated one. Using only the information provided by Lemma 9, the sets described by the Lemma are:

– in T, the set is $\{\{1, 2\}, \{1, 3\}, \{(1, 4\}, \{(2, 3\}, \{2, 4\}, \{3, 4\}$, what clearly does not correspond to the set $ASF[e]$ described in Point 1.
– in S, the set is $\{\{1, 2\}, \{1, 3\}, \{(1, 4\}, \{(2, 3\}, \{2, 4\}, \{3, 4\}, \{1, 4\}, \{2, 4\}, \{3, 4\}\}$, what corresponds to the set $ASF[e]$ described in Point 2.

Thus, these two lemmas describe the set $ASF[e]$ for some cases and for other not. Consequently, we showed that, given a network G and its edge e, a single cut-tree T of G does not provide on its own enough information for the problem addressed here. In the next section, we show that one needs to make a slight perturbation on the capacity of e and compute a cut-tree of the new network with perturbed capacity. Comparing the pairwise maximum flow values provided by the two cut-trees would explicit the vertex pairs for which their maximum flow value changed.

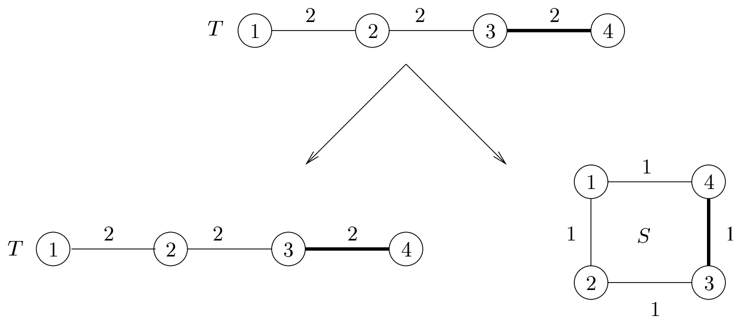

Figure 7. Approximations of $ASF[e]$

5.2 Characterization of $ASF[e]$

Given a network G, an edge e of G that is saturated by a maximum flow remains saturated even if the capacity of the edge is slightly decreased. Furthermore, consider the vertex pair $\{s,t\}$, if e is belongs to a minimum cut induced by a maximum flow $\overrightarrow{f}_{s,t}$, then the maximum flow value $f_{s,t}$ will also decrease by the same quantity [2].

Theorem 8. *Let $G = (V, E)$ be a network, $e = [i, j] \in E$ with capacity $c(e) = c_0$, and T a cut-tree of G. Consider, a fixed slight decrease $\delta_0 > 0$ on the capacity $c(e)$, i.e., $c(e) = c_0 - \delta_0 > 0$. Let G_{δ_0} be the new network and let T_{δ_0} be a cut-tree of G_{δ_0}.*

For any two vertices s and t, $\{s,t\} \in ASF[e]$ if and only if $f_{s,t}$ in T is different (by δ_0) from $f_{s,t}$ in T_{δ_0}.

Proof. Let $G = (V, E)$ be a network and $e \in E$. Suppose that an arbitrary $\{s,t\}$ is in $ASF[e]$, then any maximum flow $\overrightarrow{f}_{s,t}$ saturates e. In other terms, e belongs to a minimum cut that separates s and t. Thus, any decrease on $c(e)$ would affect the value $f_{s,t}$ since such a decrease does not avoid e to belong to a minimum cut separating s and t.

Reversely, if $f_{s,t}$ changes from a value of $c(e)$ to another, then e belongs to a minimum cut separating s and t, thus any maximum flow $\vec{f}_{s,t}$ saturates e implying that $\{s,t\} \in ASF[e]$. □

If one uses the Ford and Fulkerson [7] maximum flow algorithm, one could gain in complexity by using a capacity increase technique instead of decreasing the capacity of $c(e)$. In fact, if the capacity of $c(e)$ is increased the complexity may be improved by reusing some augmenting paths.

In Fig. 8, we illustrate how the example of Fig. 7 derives in two different cases. Let us consider the same networks S and T and $e = [3,4]$ as before. In both networks, we decrease $c(e)$ by $\delta_0 = 0.5$ and construct a cut-tree T_{δ_0}.

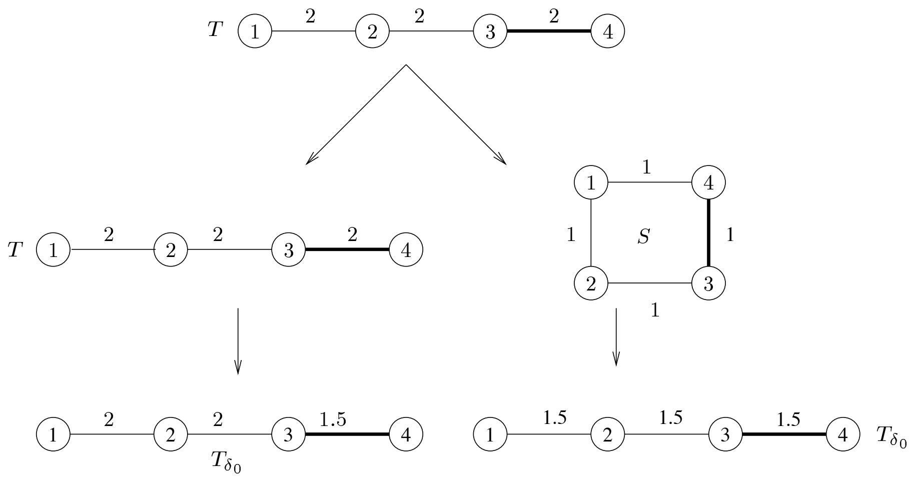

Figure 8. Characterization of $ASF[e]$

The application of Theorem 8 to each of the networks provides the respective sets $ASF[3,4]$. In the network (T),

$$ASF[3,4] = \{\{1,4\},\{2,4\},\{3,4\}\}.$$

As for network (S),

$$ASF[3,4] = \{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}\}.$$

6 Concluding remarks

In this paper, we have focused on two aspects of the sensitivity analysis of multi-terminal network flows problem. Theses two aspects deal with the use of cut-trees and parametric capacities. Some open questions could be investigated.

First, we have seen that the computation time of 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? Is there some other possible improvements using properties of the initial cut-tree?

Secondly, a pair of node s, t is in $ASF[e]$ for a given edge e of an undirected network iff all the maximum flows between s and t sautrate e (i.e., e is critical for s, t). One could also focus on the pairs s, t such that at least one maximum flow between s and t saturates e. All the edges verifying this property are globaly critical for s, t.

References

1. C. Berge. *Graphs and Hypergraphs.* North Holland, Amsterdam, 1973.
2. P. Berthomé, M. Diallo, and A. Ferreira. Generalized parametric multi-terminal flows problem. In H.L. Bodlaender, editor, *Graph Theoretical Concepts in Computer Science (WG)2003*, volume 2880 of *Lecture Notes in Computer Science*, pages 71–80, October 2003.
3. C.S. Chekuri, A.V. Goldberg, D.R. Karger, M.S. Levine, and C. Stein. Experimental study of minimum cut algorithms. In *ACM Symposium On Discrete Algorithms*, pages 324–333, New Orleans, Louisiana, 5–7 January 1997.
4. J. Cohen and E.P. Duarte Jr. Fault-tolerant routing of TCP/IP PDU's on general topology backbones. In *Design of Reliable Communication Networks*, Budapest, Hungary, October 2001.
5. M. Diallo. *Réseaux de Flots: Flots Paramétrés et Tarification.* PhD thesis, Université de Versailles, France, December 2003. In French. Available at `http://www.prism.uvsq.fr/~diallo`.
6. S.E. Elmaghraby. Sensitivity analysis of multi-terminal network flows. *J. ORSA*, 12:680–688, 1964.
7. L.R. Ford and D.R. Fulkerson. *Flows in Networks.* Princeton Univ. Press, Princeton, NJ, 1973.
8. R. L. Francis and P. B. Saunders. EVACNET: Prototype network optimization models for building evacuation. Technical Report NBSIR 79-1738, National Bureau of Standards, Washington, DC, 1979.
9. R. E. Gomory and T. C. Hu. Multi-terminal network flows. *SIAM Journal of Computing*, 9(4):551–570, December 1961.
10. D. Gusfield. Very simple methods for all pairs network flow analysis. *SIAM Journal of Computing*, 19:143–155, 1990.
11. H. W. Hamacher and S. Tjandra. Earliest arrival flow with time dependent capacity for solving evacuation problems. In M. Schreckenberg and S. D. Sharma, editors, *Pedestrian and Evacuation Dynamics*, pages 267–276. Springer, 2002.
12. H. W. Hamacher and S. Tjandra. Mathematical modelling of evacuation problems. In M Schreckenberg and S. D. Sharma, editors, *Pedestrian and Evacuation Dynamics*, pages 227–266. Springer, 2002.
13. T.C. Hu and M.T. Shing. *Combinatorial Algorithms, Enlarged 2nd Ed.* Dover Publications, INC, Mineola, New York, 2002.
14. D. M. Topkis. Monotone minimum node-cuts in capacitated networks. Technical Report ORC 70-39, University of Carlifornia, Berkley, CA, 1970.