Central European Journal of Operations Research (2020) 28:229–249
https://doi.org/10.1007/s10100-018-0579-y

ORIGINAL PAPER

# Parametric analysis on cut-trees and its application on a protein clustering problem

**João Paulo de Freitas Araujo[1] · Madiagne Diallo[1] · Fernanda Maria Pereira Raupp[2] · Pascal Berthomé[3] · José Eugenio Leal[1]**

Published online: 29 August 2018


**Abstract**
Computing the maximum flow value between a pair of nodes in a given network is a classic problem in the context of network flows. Its extension, namely, the *multi-terminal maximum flow problem*, comprises finding the maximum flow values between all pairs of nodes in a given undirected network. In this work, we provide an overview of the recent theory of sensitivity analysis, which examines the influence of a single edge capacity variation on the multi-terminal maximum flows, and we make remarks about extending some theoretical results to the case where more than one edge has their capacities changed. Based on these extensions, we present algorithms to construct the Gomory and Hu cut-trees dynamically, considering capacity variations in more than one edge in the network. Finally, the presented theory is applied on a clustering problem, in the field of biology, in order to improve an existing algorithm that identifies protein complexes in protein–protein interaction networks. In this application, a new result in the sensitivity analysis theory is introduced.

**Keywords** Multi-terminal network flows · Parametric analysis · Cut-tree · Clustering · Protein complexes

## 1 Introduction

Computing the maximum flow value between a pair of nodes in a given network is a classic problem in the context of network flows. Its extension, namely, the *multi-terminal maximum flow problem*, comprises finding the maximum flow values between all pairs of nodes in a given undirected network. These problems have several applications, particularly in the fields of transport, logistics, telecommunications and energy. More applications can be found, for example, in Agarwal and Arora (1976), Liu et al. (2011) and Diallo (2011).

✉ João Paulo de Freitas Araujo
johnpa@terra.com.br

Extended author information available on the last page of the article

It is noteworthy to mention that the multi-terminal maximum flow problem differs from the multi-commodity flow problem. In the latter, flows between multiple origins and multiple destinations are calculated simultaneously in the network, while in the former, although we compute the all pairs maximum flow values in the network, we consider, at each time, a single source and a single destination.

In the 1950s, Ford and Fulkerson (1973) popularized the maximum flow problem, which they sophisticatedly solved by demonstrating the relationship between the maximum flow value and the minimum cut capacity. At that moment, the multi-terminal maximum flow problem, in turn, could be solved by simply applying Ford and Fulkerson's algorithm $n(n-1)/2$ times to determine the maximum flows between all pairs of nodes in an undirected network of $n$ nodes.

Some years later, Gomory and Hu (1961) developed a method to compute all the maximum flow values of an undirected network by just solving $n-1$ times the maximum flow problem. Their result is resumed in a generated tree, called *cut-tree*, which encodes all the maximum flow values and identifies a minimum cut between any pair of the $n$ nodes.

Afterward, Gusfield (1990) presented a simpler procedure to obtain the same cut-tree but also requiring $n-1$ calls to a maximum flow algorithm. Goldberg and Tsioutsiouliklis (2001) then conducted a study comparing computationally three variations of the Gomory and Hu algorithm and the Gusfield algorithm. Those authors concluded that the Gomory and Hu algorithm, with some heuristics, is more robust than the one of Gusfield. In the case of a network with unitary capacity in each edge (the unweighted case), Bhalgat et al. (2007) showed a faster algorithm that does not use a maximum flow algorithm as an internal procedure.

The theory of sensitivity analysis on multi-terminal maximum flows started with the seminal work of Elmaghraby (1964), who studied the effects on maximum flow values of a network under the variation of the capacity of a single edge, called a *parametric edge*. Recently, Barth et al. (2006) improved the results of Elmaghraby for the case of a single parametric edge and generalized it to the case of a network with $k$ parametric edges, noting that $2^k$ cut-trees are sufficient to calculate all the maximum flows for any parameters values.

To reduce the complexity of the algorithms that need to compute two or more cut-trees in a sequence, considering an increasing variation of the capacity of a single edge in the network, Barth et al. (2006) showed how to use the information of an already computed cut-tree for building the next one. Hartmann and Wagner (2013) demonstrated the same for the case of a decreasing variation of the capacity of a single edge.

In this work, we aim to provide an overview of the theory of sensitivity analysis on multi-terminal maximum flows and extend the theoretical results of Barth et al. (2006) and Hartmann and Wagner (2013) in this context, covering not only the case of a single parametric edge but also more than one parametric edge. Additionally, we also introduce a property that relates cut nodes and cut-trees.

According to the extensions introduced in this work to the sensitivity analysis theory, we present two algorithms to construct the Gomory and Hu cut-trees dynamically, considering capacity variations in more than one edge in the network. For the case of increasing variations, the first proposed algorithm is straightforwardly obtained with a

modification on the Gomory and Hu algorithm, while for the case of decreasing variations, the second proposed algorithm is a generalization of the algorithm in Hartmann and Wagner (2013).

Finally, we apply the theoretical results obtained in this work to improve the algorithm in Mitrofanova et al. (2009), which identifies protein complexes in protein–protein interaction (PPI) networks. Those authors developed an algorithm that uses the computation of successive Gomory and Hu cut-trees in the identification of the protein complexes. In this way, we seek to reduce the computational effort performed by the existing algorithm, when we consider more than one capacity variations and reusable cuts in the PPI networks. A new result on reusable cuts is introduced after the exclusion of a minimum cut of the cut-tree.

Summarizing, the problem we are dealing with in this work can be formalized as below.

**Problem 1** Let $G$ be a network and $CT$ its cut-tree. Consider that some edges of $G$ have their capacities changed, generating $G'$. The goal is to determine the new cut-tree $CT'$ performing as few maximum flow computations as possible.

The paper is structured as follows. Section 2 presents the used notations and basic concepts related to graph theory and maximum flow problems. Section 3 presents the method of Gomory and Hu (1961), which solves efficiently the multi-terminal maximum flow problem. In Sect. 4, we study the sensitivity analysis theory of Gomory and Hu cut-trees, and we extend it to the case of more than one parametric edge in the network. Section 5 introduces the proposed algorithms that compute cut-trees dynamically upon changes of capacity in one or more edges. The problem of identifying protein complexes in PPI networks solved by Mitrofanova et al.'s algorithm is addressed in Sect. 6. Additionally, we introduce a procedure to reduce its computational effort. Concluding remarks and perspectives are offered in Sect. 7.

## 2 Notation and basic concepts

From now on, we assume that the reader has previous knowledge of graph theory and network problems. Reference books include Ford and Fulkerson (1973) as well as Hu and Shing (2002).

Let $G = (V, E)$ be a connected undirected graph, comprising a set $V$ of $n$ nodes $v$, and a set $E$ of $m$ edges $e$, where each edge is an unordered pair $[i, j]$ of nodes in $V$. A *network* is a graph $G$ associated with a *capacity* function over the edges $c: E \rightarrow R_+$. A flow from a source node $s$ to a terminal node $t$ in $G$ is a function $f: E \rightarrow R_+$ with the conservation property at each node $v$, except for $s$ and $t$, that is,

$$\sum_{i \in V} f[i, v] = \sum_{j \in V} f[v, j] \quad \forall v \in V \backslash \{s, t\},$$

and the capacity constraint

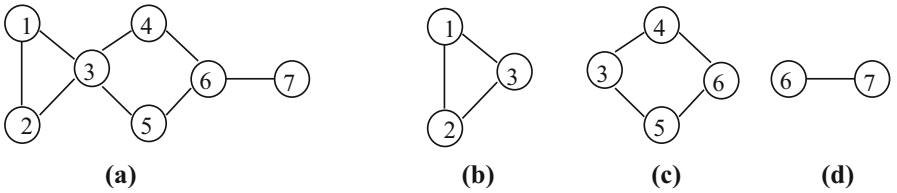


**Fig. 1** The graph (**a**) and its three biconnected components (**b**, **c, d**)

$$\forall i, \quad j \in V, \ \ f[i, j] \leq c[i, j].$$

Observe that an edge can be represented by two arcs of opposite directions. Therefore, unlike an arc, an edge has no direction; that is, it has two opposite directions at once. Thus, an undirected network has the same structure as a directed symmetric network, where each arc has the same capacity of the original edge, allowing equal capacity to either direction. It is important to note that, when a flow passes through a capacitated edge, it uses only one arc, never both. Hereafter, we will deal only with undirected networks.

We denote by $(s - t)$ a cut separating the nodes $s$ and $t$, by $c(s - t)$ the capacity of the cut (cut value), and by $\left(X, \ \bar{X}\right)$ a cut separating the nodes of a graph into two complementary subsets $X$ and $\bar{X}$. Among all possible cuts separating $s$ and $t$, one with the smallest capacity is called a *minimum cut*, and its capacity is the maximum flow value between the pair of nodes, which will be represented by $f_{s,t}$.

We say that a node is a *cut node* when its deletion disconnects the graph, whereas a *biconnected component* is a maximal connected subgraph containing no cut nodes. The term "maximal" refers to the state that any inclusion of a node in the biconnected component creates a cut node in it. In addition, an edge is a *bridge* when its deletion disconnects the graph. These concepts are exemplified in Fig. 1: in (a) nodes 3 and 6 are cut nodes and edge [6, 7] is a bridge; (b), (c) and (d) are the three biconnected components of the graph in (a).

## 3 Basic procedure

In this section, we recall the definition of cut-tree and the method of Gomory and Hu (1961), which solves the multi-terminal maximum flow problem for undirected networks. For the reader interested in studying the case of directed networks, see Scutellà (2007).

After observing the existence of at most $n-1$ distinct values of maximum flow in an undirected network with $n$ nodes, Gomory and Hu (1961) developed an algorithm that obtains the $n(n-1)/2$ values of maximum flows using a node contraction scheme and running only $n - 1$ times the maximum flow algorithm. The final result is expressed by a cut-tree defined as follows.

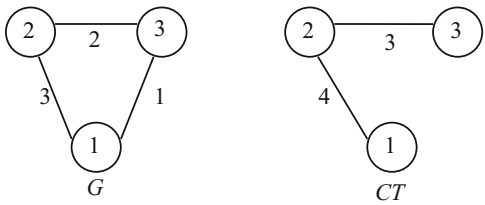


**Fig. 2** Example of a cut-tree *CT* of a network *G*

**Definition 1** A cut-tree of a network $G = (V, E)$ is a tree $CT = (V, E')$ obtained from $G$, with weighted edges and the same set of nodes $V$. A cut-tree $CT$ has the following properties:

1. *Equivalent flow tree* The value of the maximum flow between any $s$ and $t$ *in* $G$ is equal to the value of the maximum flow between $s$ and $t$ in $CT$, i.e., the smallest edge capacity on the unique path connecting $s$ to $t$ in $CT$; in this way, the maximum flows between all pairs of nodes in $G$ are represented in $CT$;
2. *Cut property* The removal of any edge $e$ with capacity $c(e)$ from $CT$ separates its nodes into two sets. This partition corresponds to a minimum cut in $G$ with capacity $c(e)$, separating the two extremities of the edge $e$.

Figure 2 illustrates an example of a cut-tree $CT$ constructed from a network $G$. As we can see, by the properties mentioned above, the minimum cut between nodes 2 and 3 and between nodes 1 and 2 are respectively reflected by the edges [2, 3] and [1, 2] of the cut-tree $CT$. Its maximum flow values are the capacities of these edges, that is, 3 and 4. The minimum cut between nodes 1 and 3 is reflected, in turn, by the edge [2, 3] in $CT$, with a maximum flow value equal to 3.

In general, there are several cut-trees for a given network. The cut-tree will be unique only if all the $(s - t)$ minimum cuts of the network are unique.

The algorithm created by Gomory and Hu induces, through nodes contractions, the construction of minimum cuts that do not intersect. Thus, they showed that the maximum flow between two noncontracted nodes computed on a contracted network is of equal value when the maximum flow is computed on the original network. The minimum cut found on the contracted network is also minimum cut in the original network by simply replacing the super-nodes (nodes in contraction) by the nodes that compose them. In this algorithm, the final cut-tree is achieved when all super-nodes have only one node each, which occurs after $(n - 1)$ resolutions of the maximum flow problem. Figure 3 presents the pseudo-code of the Gomory and Hu (GH) algorithm.

## 4 Sensitivity analysis on cut-trees

In this section, we study the theory of computing cut-trees dynamically from the works of Barth et al. (2006) and Hartmann and Wagner (2013), and we extend some theoretical results to the case where more than one edge have their capacities changed. Additionally, we introduce a property that relates cut nodes and cut-trees.

The theory of dynamically computing cut-trees comes up from the parametric version of the multi-terminal maximum flow problem, called a parametric problem, which

**Input:** $G$
1 Build $CT$ by creating a super-node with all nodes of $G$;
2 **while** *there exists in CT a super-node* $\bar{v}$ *with more than one node* **do**
3 Select two arbitrary nodes $s$ and $t$ of $\bar{v}$ ;
4 In $G$, contract the nodes of each connected component of $CT \setminus \bar{v}$ into one super-node, generating a contracted network $G'$;
5 Compute a minimum cut $(X, \bar{X})$ between $s$ and $t$ in $G'$;
6 Separate $\bar{v}$ into two super-nodes $\bar{v}_1$ and $\bar{v}_2$, such that the nodes of $\bar{v}_1$ are in $X$ and the nodes of $\bar{v}_2$ are in $\bar{X}$ ;
7 Connect $\bar{v}_1$ and $\bar{v}_2$ with an edge of capacity $c(X, \bar{X})$;
8 Connect the super-nodes previously adjacent to $\bar{v}$ to either $\bar{v}_1$ or $\bar{v}_2$ depending on their position in $(X, \bar{X})$;
9 **end while**
10 **return** $CT$;

**Fig. 3** Pseudo-code of Gomory and Hu (GH) algorithm

is based on the theory of sensitivity analysis of multi-terminal maximum flows. In general terms, this theory studies the behavior of the all pairs maximum flow values in a network under edge capacity variations.

Barth et al. (2006) examined this parametric problem considering the increasing variation of an edge capacity. The problem can be formulated as follows for a given pair of nodes.

**Problem 2** Let $G = (V, E)$ be a network with source node $s$ and terminal node $t$. Consider an edge $e = [i, j] \in E$ with non-negative capacity $c(e) = \lambda$. The goal is to determine the maximum flow value between $s$ and $t$ with the increasing variation of $\lambda$.

The same authors concluded that, if an edge with null capacity is not part of a $(s - t)$ minimum cut, then it is still not part of a $(s - t)$ minimum cut when its capacity increases. Therefore, for a network that has only one edge with parametric capacity, the variation of this capacity may not influence the values of maximum flows (and minimum cuts) for various pairs of nodes. Denoting by $f^{\lambda}_{s,t}$ the maximum flow value between $s$ and $t$ when the capacity of the edge $e$ is $\lambda$, this result is formalized as follows.

**Lemma 1** (Barth et al. 2006) *Let* $G = (V, E)$ *be a network with n nodes and* $e = [i, j] \in E$ *such that* $c(e) = \lambda$. *Let s and t be a pair of nodes of G and* $CT^{\alpha}$ *a cut-tree when* $c(e) = \alpha$. *If the path connecting s to t* $P_{s,t}$ *in* $CT^{\alpha}$ *has no edge in common with* $P_{i,j}$, *then* $f^{\lambda}_{s,t} = f^{\alpha}_{s,t}$, $\forall \lambda > \alpha \geq 0$.

***Proof*** Using the cut property of the cut-trees (Definition 1, item 2), one can show that there exists a minimum cut $C^{\alpha}_{s,t}$ separating $s$ and $t$ where both nodes $i$ and $j$ ($e = [i, j]$) are in the same side of the minimum cut. Therefore, the cut does not contain $e$ for $\lambda > \alpha$, and it is insensible to the variation of $\lambda$. □

Based on Lemma 1, we introduce the following theoretical extension regarding the case of several edges with parametric capacities.

**Corollary 1** *Let $G$ be a network with $k$ edges $e_x = [i_x, j_x]$, where $x = 1, 2, \ldots, k$, of parametric capacities $c(e_x) = \lambda_x$. Let $CT^{\alpha}$ be a cut-tree obtained when $c(e_x) = \alpha_x$ and $P^x_{i,j}$ the path between $i_x$ and $j_x$ in $CT^{\alpha}$. For $\lambda_x > \alpha_x$ the minimum cuts of $CT^{\alpha}$ not in $\bigcup_{x=1}^{k} P^x_{i,j}$ can be reused in the cut-tree $CT^{\lambda}$.*

***Proof*** According to Lemma 1, the increasing variations of the edges $e_x$ will not influence the cut-tree $CT^{\alpha}$, except for the union of its paths $P^x_{i,j}$. □

Next, we give a note on adding edges to a network.

- To affirm that a given edge is not contained in a network is equivalent to saying that this edge is contained in the network and has null capacity. Therefore, adding an edge to a network can be understood as varying positively its capacity from zero.

So far, the results in this section apply only to the cases of adding edges to a network or varying increasingly its capacities. For the decreasing variation case, and, in an analogous way, the exclusion of edges, Hartmann and Wagner (2013) stated:

**Lemma 2** (Hartmann and Wagner 2013) *Let $G$ be a network with an edge $e = [i, j]$ of parametric capacity $c(e) = \lambda$. Let $CT^{\alpha}$ be a cut-tree obtained when $c(e) = \alpha$ and $P_{i,j}$ the path between $i$ and $j$ in $CT^{\alpha}$. For $0 \leq \lambda < \alpha$ the minimum cuts in $P_{i,j}$ remain valid in $CT^{\lambda}$ with capacity decreased by $(\alpha - \lambda)$.*

***Proof*** The edge $e$ belongs to all minimums cuts in $P_{i,j}$, and only to them in $CT^{\alpha}$. As $c(e)$ decreases by $(\alpha - \lambda)$, all cuts of $G$ decreases by at most $(\alpha - \lambda)$, so the minimum cuts of $P_{i,j}$ remain valid in $CT^{\lambda}$ with capacity decreased by $(\alpha - \lambda)$. □

A theoretical extension to the result of Lemma 2 is introduced in Lemma 3.

**Lemma 3** *Let $G$ be a network with $k$ edges $e_x = [i_x, j_x]$, where $x = 1, 2, \ldots, k$, of parametric capacities $c(e_x) = \lambda_x$. Let $CT^{\alpha}$ be a cut-tree obtained when $c(e_x) = \alpha_x$ and $P^x_{i,j}$ the path between $i_x$ and $j_x$ in $CT^{\alpha}$. For $0 \leq \lambda_x < \alpha_x$ the minimum cuts in $\bigcap_{x=1}^{k} P^x_{i,j}$ remain valid in $CT^{\lambda}$ with capacity decreased by $\sum_{x=1}^{k} (\alpha_x - \lambda_x)$.*

***Proof*** Following the proof of Lemma 2, with the decreases of $c(e_x)$, all cuts of $G$ decreases by at most $\sum_{x=1}^{k} (\alpha_x - \lambda_x)$, so the minimum cuts of $\bigcap_{x=1}^{k} P^x_{i,j}$ remain valid in $CT^{\lambda}$ with capacity decreased by $\sum_{x=1}^{k} (\alpha_x - \lambda_x)$. □

Hartmann and Wagner (2013) also commented on other reusable cuts for the decreasing variation case.

**Lemma 4** (Hartmann and Wagner 2013) *Let $G$ be a network with an edge $e = [i, j]$ of parametric capacity $c(e) = \lambda$. Let $CT^{\alpha}$ be a cut-tree obtained when $c(e) = \alpha$ and $\left(X, \bar{X}\right)$ a $(s - t)$ minimum cut in $G^{\lambda}$ for $0 \leq \lambda < \alpha$. Let $i, j \in \bar{X}$ and edge $[g, h] \in CT^{\alpha}$ with $g, h \in X$. Then, $[g, h]$ is a $(g - h)$ minimum cut in $G^{\lambda}$.*

***Proof*** Suppose by contradiction that there exists a $(g - h)$ minimum cut in $G^{\lambda}$ that is cheaper than the cut represented by $[g, h]$. Note that the cut $[g, h]$ costs the same in $G^{\alpha}$ and $G^{\lambda}$. Such a cheaper $(g - h)$ minimum cut in $G^{\lambda}$ would separate $i$ and $j$ in $\bar{X}$. At the same time, since the Gomory and Hu algorithm induces minimum cuts that do not intersect, there is a $(g - h)$ minimum cut that does not separate $i$ and $j$. However, the latter cut is already cheaper in $G^{\alpha}$, contradicting the hypothesis. □

In Lemma 5, we extended the results of Lemma 4 to consider decreasing variations in more than one edge.

**Lemma 5** *Let $G$ be a network with $k$ edges $e_x = [i_x, j_x]$, where $x = 1, 2, \ldots, k$, of parametric capacities $c(e_x) = \lambda_x$. Let $CT^{\alpha}$ be a cut-tree obtained when $c(e_x) = \alpha_x$ and $P^x_{i,j}$ the path between $i_x$ and $j_x$ in $CT^{\alpha}$. Let $\left(X, \bar{X}\right)$ be a $(s-t)$ minimum cut in $G^{\lambda}$ for $0 \leq \lambda_x < \alpha_x$ and $y$ the number of parametric edges that has $i_x, j_x \in \bar{X}$. Let edge $[g, h] \in CT^{\alpha}$ with $g, h \in X$ and the set $L = \{x : [g, h] \in P^x_{i,j}\}$. If $(k-y) = |L|$, then $[g, h]$ is a $(g-h)$ minimum cut in $G^{\lambda}$ with capacity decreased by $\sum_{x \in L} (\alpha_x - \lambda_x)$.*

***Proof*** We can demonstrate Lemma 5 through the Gomory and Hu algorithm. Let $\left(X, \bar{X}\right)$ be the first $(s-t)$ minimum cut found by the algorithm and $Y$ the set of the parametric edges with extremities $i_x, j_x \in \bar{X}$. From the second iteration until the last one, for $s, t \in X$, all $(s-t)$ minimum cuts will be computed in contracted networks that do not contain any edge of $Y$. Therefore, based on the proof of Lemma 3, if a $(s-t)$ minimum cut $C$ in $CT^{\alpha}$ contains all parametric edges of $G$, except those in $Y$, then $C$ remains valid in $CT^{\lambda}$ with capacity decreased by $\sum_{x \in L} (\alpha_x - \lambda_x)$. □

Next, we introduce a remark on the relationship between cut nodes and cut-trees.

**Lemma 6** *A cut-tree of a biconnected component can be computed independently of the rest of the graph.*

***Proof*** Let $s$ and $t$ be two nodes of a biconnected component $A$. Because there is no path between $s$ and $t$ that contains an edge not in $A$, a minimum cut between $s$ and $t$ can only contain edges from $A$. □

## 5 Algorithms for computing cut-trees dynamically

Upon the theory about sensitivity analysis on cut-trees, we propose here two algorithms to compute cut-trees dynamically when variations of capacity occur in the edges. For the case of increasing capacity variations, the first algorithm results from a modification in the Gomory and Hu algorithm. The proposed modification seeks to collect those edges that are not in $\bigcup_{x=1}^{k} P^x_{i,j}$ and to reuse them as minimum cuts between their extremities. After that, the Gomory and Hu algorithm proceeds as usual.

Before presenting the second proposed algorithm, we will study the algorithm proposed in Hartmann and Wagner (2013), which solves the multi-terminal maximum flow problem in terms of a cut-tree, considering a decreasing variation on a single edge capacity in the network. In the algorithm, illustrated in Fig. 4 and namely HW, the cuts of the current cut-tree are marked as valid for the generation of the next cut-tree. Observe that, as the HW algorithm evolves, the extremities of valid and non-valid edges can alter, which does not change the cut represented by them. The procedure ends when all the cuts, or edges, are valid.

The HW algorithm addresses bridges as we see in lines 1 and 9. The reader can understand these commands recalling Lemma 6, since a bridge is a biconnected component. The command in line 5 in the HW algorithm comes from Lemma 2, and the

**Input:** $G, e = [i, j], CT^{\alpha}, \Delta = (\alpha - \lambda)$

1 **if** $[i, j]$ *is a bridge* **then**
2 Replace, in $CT^{\alpha}$, $\alpha$ with $\lambda$ in $c(e)$;
3 **return** $CT^{\lambda}$;
4 **end if**
5 Decrease the capacity of the edges in $P_{i,j}$ by $\Delta$ and mark the edges as valid;
6 $Q \leftarrow$ non-valid edges non-increasingly ordered by their costs;
7 **while** $Q \neq \emptyset$ **do**
8 $[u, v] \leftarrow$ most expensive non-valid edge with $v$ on $P_{i,j}$;
9 **if** $[u, v]$ *is a bridge in* $G$ **then**
10 Mark $[u, v]$ as a valid edge;
11 Consider the subtree $U$ rooted at $u$ with $v \notin U$;
12 Mark all edges in $U$ valid, remove valid edges from $Q$;
13 **else**
14 $(U, \overline{U}) \leftarrow (u - v)$ minimum cut in $G^{\lambda}$ with $u \in U$;
15 Mark $[u, v]$ as a valid edge, remove $[u, v]$ from $Q$;
16 **if** $c[u, v] = c(U, \overline{U})$ **then**
17 Consider the subtree $U$ rooted at $u$ with $v \notin U$;
18 Mark all edges in $U$ valid, remove valid edges from $Q$;
19 **else**
20 $c[u, v] \leftarrow c(U, \overline{U})$;
21 $N \leftarrow$ neighbors of $v$;
22 **forall the** $g \in N$ **do**
23 **if** $g \in U$ **then** reconnect $g$ to $u$;
24 **end forall**
25 **end if**
26 **end if**
27 **end while**
28 **return** $CT^{\lambda}$;

**Fig. 4** Pseudo-code of Hartmann and Wagner (HW) algorithm

command in lines 16–18 from Lemma 4. For the explanation of selecting the most expensive non-valid edge in line 8, please refer to the work of Hartmann and Wagner (2013).

We remark that the operations in lines 21–23 of the HW algorithm are similar with the ones of the Gusfield algorithm, as they obtain minimum cuts that do not intersect through cuts computed in the (noncontracted) original network. The correctness of commands 21–23, regarding the reconnection of non-valid edges, is grounded in Theorem 1 of Hartmann and Wagner (2013), which demonstrates the existence of minimum cuts in the new generated network that do not intersect minimum cuts of the old network.

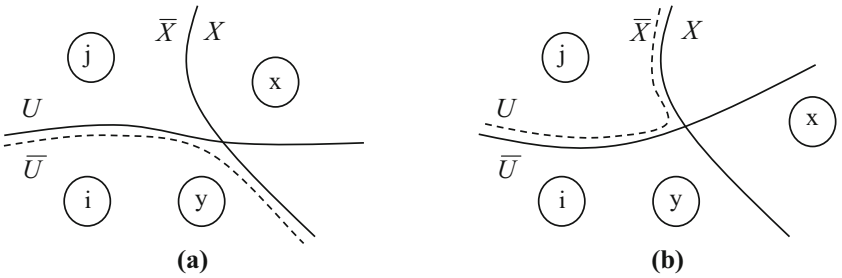


**Fig. 5** Situations of Theorem 1: (i) (**a**) and (ii) (**b**)

**Theorem 1** (Hartmann and Wagner 2013) *Let $G$ be a network with an edge $e = [i, j]$ of capacity $c(e) = \alpha$. Let $(X, \bar{X})$ be a $(x - y)$ minimum cut in $G$ with $x \in X$, $y \in \bar{X}$ and $\{i, j\} \subseteq \bar{X}$. Let further $(U, \bar{U})$ be a cut that separates $i$ and $j$. For $c(e) = \lambda < \alpha$:*

(i) *if $(U, \bar{U})$ separates $x$ and $y$ with $x \in U$, then $c(U \cup X, \overline{U \cup X}) \leq c(U, \bar{U})$;*
(ii) *if $(U, \bar{U})$ does not separate $x$ and $y$ with $x \in \bar{U}$, then $c(U \backslash X, \overline{U \backslash X}) \leq c(U, \bar{U})$.*

***Proof*** Please consult the cited reference.

Figure 5 illustrates the two situations mentioned in Theorem 1.

We still note that we omitted some commands in the HW algorithm related to a property that will not be considered in the second proposed algorithm. As the number of parametric edges increases, the verification of the property becomes burdensome. Further research can verify if this property is still worthy for a certain number of parametric edges.

For the case of decreasing capacity variations, the second algorithm to be proposed in this work results from a generalization of the HW algorithm. In the general version of the algorithm, we will propose to address biconnected components, instead of bridges, since a network can have two or more biconnected components without containing a bridge. To compute a cut-tree dynamically when decreasing capacity variations occur in more than one edge, we must apply the results of Lemmas 3 and 5 on the HW algorithm, leading to the Generalized Hartmann and Wagner (GHW) algorithm, which is illustrated in Fig. 6.

Next, we show a small example of the application of GHW in the network $G$ and its cut-tree $CT$, both illustrated in Fig. 7. During this demonstration, non-valid edges will be represented by thin edges in the cut-trees, while valid edges will be represented by fat edges.

Suppose that the capacities of the edges $e_1 = [3, 4]$ and $e_2 = [4, 5]$ of $G$ have been reduced. The new network $G^{\lambda}$ is shown in Fig. 8a. Figure 8b illustrates the intermediate cut-tree after the execution of lines 1, 2, 3 and 4 of GHW.

In the first execution of the function **while**, the algorithm selects the edge [1, 2], where $u = 2$ and $v = 1$ (line 7). Since there is no cut node in $G$, the condition of line 8 is never true; therefore, the algorithm proceeds and executes the command in line 13, resulting in $U = \{2, 3\}$ and $c(U, \bar{U}) = 4$. In the following commands, the condition

**Input:** $G, k, e_x = [i_x, j_x],\ CT^{\alpha}, \Delta_x = (\alpha_x - \lambda_x)$

1 **if** $\bigcap_{x=1}^{k} P_{i,j}^{x} = \emptyset$ **then return** $CT^{\lambda}$ computed by the GH algorithm;
2 $B \leftarrow$ biconnected component of $G$ that contains the edges $e_x$;
3 **for** $x = 1, \ldots, k$ **do** decrease the capacity of the edges in $P_{i,j}^{x}$ by $\Delta_x$;
4 Mark the edges in $\bigcap_{x=1}^{k} P_{i,j}^{x}$ as valid;
5 $Q \leftarrow$ non-valid edges non-increasingly ordered by their costs;
6 **while** $Q \neq \emptyset$ **do**
7 $[u, v] \leftarrow$ most expensive non-valid edge with $v$ being an extremity of a valid edge;
8 **if** $u \notin B$ **then**
9 Mark $[u, v]$ as a valid edge;
10 Consider the subtree $U$ rooted at $u$ with $v \notin U$;
11 Mark all edges in $U$ valid, remove valid edges from $Q$;
12 **else**
13 $(U, \overline{U}) \leftarrow (u\text{-}v)$ minimum cut in $G^{\lambda}$ with $u \in U$;
14 Mark $[u, v]$ as a valid edge, remove $[u, v]$ from $Q$;
15 **if** $c[u, v] = c(U, \overline{U})$ **then**
16 Consider the subtree $U$ rooted at $u$ with $v \notin U$;
17 $L_{[u,v]} \leftarrow \{x : [u,v] \in P_{i,j}^{x}\}$;
18 Mark as valid all edges $e$ of $U$ in which $L_e = L_{[u,v]}$, remove valid edges from $Q$;
19 **else**
20 $c[u, v] \leftarrow c(U, \overline{U})$;
21 $N \leftarrow$ neighbors of $v$;
22 **forall the** $g \in N$ **do**
23 **if** $g \in U$ **then** reconnect $g$ to $u$;
24 **end forall**
25 $L_{[u,v]} \leftarrow \{x : [u,v] \in P_{i,j}^{x}\}$;
26 **if** $|L_{[u,v]}| < k$ **then**
27 Consider the subtree $U$ rooted at $u$ with $v \notin U$;
28 $M_U \leftarrow \{x : (i_x \in U) \wedge (j_x \in U)\}$;
29 Mark as valid all edges $e$ of $U$ in which $|L_e| = |L_{[u,v]}| + |M_U|$, remove valid edges from $Q$;
30 Consider the subtree $\overline{U}$ rooted at $v$ with $u \notin \overline{U}$;
31 Mark as valid all edges $e$ of $\overline{U}$ in which $|L_e| = k - |M_U|$, remove valid edges from $Q$;
32 **end if**
33 **end if**
34 **end if**
35 **end while**
36 **return** $CT^{\lambda}$;

**Fig. 6** Pseudo-code of the Generalized Hartmann and Wagner (GHW) algorithm

in line 15 is not true, node 3 is reconnected to node 2 in the intermediate cut-tree (lines 21, 22 and 23), and from line 25–31, the new edge [2, 3] (old edge [1, 3]) is marked valid, once $L_{[u,v]} = \{1\}, k = 2$, subtree $U = \{[2, 3]\}, M_U = \emptyset$ and $L_{[2,3]} = \{1\}$.

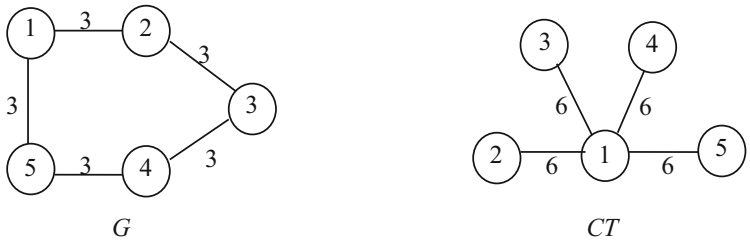


**Fig. 7** Network $G$ and its cut-tree $CT$

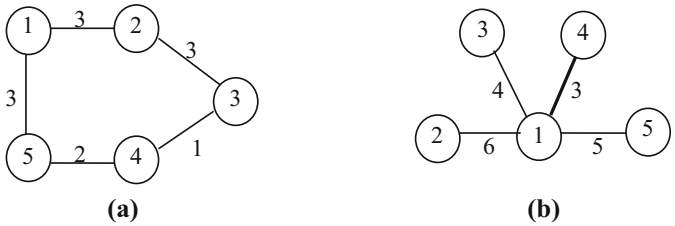


**Fig. 8** New network $G^{\lambda}$ (**a**) and intermediate cut-tree in GHW after execution of line 4 (**b**)

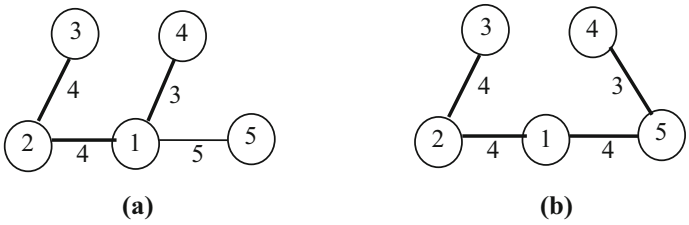


**Fig. 9** Intermediate cut-tree in GHW after the first (**a**) and the second (**b**) execution of the function **while**

Figure 9a illustrates the intermediate cut-tree after the first execution of the function **while**.

In the second execution of the function **while**, the results are: edge [1, 5] selected, with $u = 5$ and $v = 1$ (line 7); $U = \{4, 5\}$ and $c\left(U, \bar{U}\right) = 4$ (line 13); not true condition (line 15); node 4 reconnected to node 5 (lines 21, 22 and 23); subtrees $U = \{[4, 5]\}$ and $\bar{U} = \{[1, 2], [2, 3]\}$ do not contain non-valid edges (lines 25–31). Figure 9b illustrates the final cut-tree after the second execution of the function **while**.

Observe that, in the GHW algorithm, the condition $\bigcap_{x=1}^{k} P_{i,j}^{x} \neq \emptyset$ must be satisfied (line 1). Based on Lemma 6, we can affirm that all the edges of a minimum cut belong to the same biconnected component. Therefore, since the GHW algorithm requires $\bigcap_{x=1}^{k} P_{i,j}^{x} \neq \emptyset$, the following command (line 2) is correct.

The command **for** (line 3) updates the values of the cuts for the new generated cut-tree, according to the decreasing of capacity of the edges $e_x$. Observe that in line 4, we find the application of the results of Lemma 3, in lines 8–11, we see the results of Lemma 6, and in lines 15–18 and 25–31, we verify the results of Lemma 5.

When the **if** condition in line 15 is true, the algorithm can reuse the cut $[u, v]$. Therefore, commands 15–18, related to Lemma 5, can be understood as follows.

Suppose that the GHW algorithm just finished the command in line 4, and let $[u, v]$ be any non-valid edge in which $v$ is closer to a valid edge than $u$. Let $U$ be the subtree rooted at $u$ with $v \notin U$. Once the cuts in $\bigcap_{x=1}^{k} P_{i,j}^{x}$ are not in $U$, all paths $P_{i,j}^{x}$ that contain at least one edge in $U$ must contain the edge $[u, v]$. This situation remains during the evolution of the GHW algorithm, because, although there are reconnections in line 23, non-valid cuts never change when its extremities alter. In Theorem 2, which we introduce below, this condition is expressed by $\forall x, \{i_x, j_x\} \not\subset X$. The reconnections in line 23 occur when the cut $[u, v]$ cannot be reused in the new generated cut-tree. Theorem 2 allows reconnections in the case of non-valid edges by showing that some new minimum cuts do not cross old minimum cuts. Observe that, as the new computed cut has capacity lower than the previous $[u, v]$ cut, it must contain at least one edge $e_x = [i_x, j_x]$ that does not belong to the previous $[u, v]$ cut. Theorem 2 includes this information when defining the cut $\left(U,\ \bar{U}\right)$.

**Theorem 2** *Let $G$ be a network with $k$ edges $e_x = [i_x, j_x]$, where $x$=1, 2,..., $k$, of capacities $c(e_x) = \alpha_x$. Let $\left(X,\ \bar{X}\right)$ be a $(g - h)$ minimum cut in $G$ with $g \in X$,$h \in \bar{X}$ and $\forall x, \{i_x, j_x\} \not\subset X$. Let further $\left(U,\ \bar{U}\right)$ be a cut that separates at least one pair $i_x$ and $j_x$ such that $\{i_x, j_x\} \subseteq \bar{X}$. For $c(e_x) = \lambda_x\ <\ \alpha_x$:*

(i) *if $\left(U,\ \bar{U}\right)$ separates $g$ and $h$ with $g \in U$, then $c\left(U \cup X,\ \overline{U \cup X}\right) \leq c\left(U,\ \bar{U}\right)$;*
(ii) *if $\left(U,\ \bar{U}\right)$ does not separate $g$ and $h$ with $g \in \bar{U}$, then $c\left(U \backslash X,\ \overline{U \backslash X}\right) \leq c\left(U,\ \bar{U}\right)$.*

***Proof*** For this proof, we denote by $c^{\Theta}$ the capacity of the cut in the new network when $c(e_x) = \lambda_x\ <\ \alpha_x$.To prove item (i), consider the capacity of the following cuts:

- $$c\left(X,\ \bar{X}\right) = c\left(U \cap X,\ \bar{X}\right) + c\left(\bar{U} \cap X,\ \bar{X}\right)$$
- $$c\left(U \cap X,\ \overline{U \cap X}\right) = c\left(U \cap X,\ \bar{X}\right) + c\left(U \cap X,\ \bar{U} \cap X\right)$$

Observe that both cuts, $\left(X,\ \bar{X}\right)$ and $\left(U \cap X,\ \overline{U \cap X}\right)$, separate $g$ and $h$. As $\left(X,\ \bar{X}\right)$ is a $(g - h)$ minimum cut, then $c\left(\bar{U} \cap X,\ \bar{X}\right) \leq c\left(U \cap X, \bar{U} \cap X\right)$. According to the condition of the theorem $\forall x, \{i_x, j_x\} \not\subset X$, $c^{\theta}\left(U \cap X, \bar{U} \cap X\right) = c\left(U \cap X, \bar{U} \cap X\right)$; therefore:

$$c^{\theta}\left(\bar{U} \cap X,\ \bar{X}\right) \leq c^{\theta}\left(U \cap X, \bar{U} \cap X\right). \tag{1}$$

Consider now the capacity of the following cuts in the new generated network:

- $$c^{\theta}\left(U,\ \bar{U}\right) = c^{\theta}\left(\bar{U} \cap \bar{X},\ U\right) + c^{\theta}\left(U \cap X,\ \bar{U} \cap X\right) + c^{\theta}\left(U \cap \bar{X},\ \bar{U} \cap X\right)$$
- $$c^{\theta}\left(U \cup X,\ \overline{U \cup X}\right) = c^{\theta}\left(\bar{U} \cap \bar{X},\ U\right) + c^{\theta}\left(\bar{U} \cap \bar{X}, \bar{U} \cap X\right)$$

From Eq. (1) and $\left(\bar{U} \cap \bar{X}, \bar{U} \cap X\right) \subseteq \left(\bar{U} \cap X,\ \bar{X}\right)$, we conclude that:

$$c^{\theta}\left(U \cup X,\ \overline{U \cup X}\right) \leq c^{\theta}\left(U,\ \bar{U}\right).$$

To prove item (ii) consider the capacity of the following cuts:

- $$c\left(X,\ \bar{X}\right) = c\left(\bar{U} \cap X,\ \bar{X}\right) + c\left(U \cap X,\ \bar{X}\right)$$
- $$c\left(\bar{U} \cap X,\ \overline{\bar{U} \cap X}\right) = c\left(\bar{U} \cap X,\ \bar{X}\right) + c\left(\bar{U} \cap X, U \cap X\right)$$

Observe that both cuts, $\left(X,\ \bar{X}\right)$ and $\left(\bar{U} \cap X,\ \overline{\bar{U} \cap X}\right)$, separate $g$ and $h$. As $\left(X,\ \bar{X}\right)$ is a $(g-h)$ minimum cut, then $c\left(U \cap X,\ \bar{X}\right) \leq c\left(\bar{U} \cap X, U \cap X\right)$. According to the condition of the theorem $\forall x, \{i_x, j_x\} \not\subset X, c^{\theta}\left(\bar{U} \cap X, U \cap X\right) = c\left(\bar{U} \cap X, U \cap X\right)$; therefore:

$$c^{\theta}\left(U \cap X,\ \bar{X}\right) \leq c^{\theta}\left(\bar{U} \cap X, U \cap X\right). \tag{2}$$

Consider now the capacity of the following cuts in the new generated network:

- $$c^{\theta}\left(U,\ \bar{U}\right) = c^{\theta}\left(U \cap \bar{X},\ \bar{U}\right) + c^{\theta}\left(\bar{U} \cap X,\ U \cap X\right) + c^{\theta}\left(U \cap X,\ \bar{U} \cap \bar{X}\right)$$
- $$c^{\theta}\left(U \backslash X,\ \overline{U \backslash X}\right) = c^{\theta}\left(U \cap \bar{X},\ \bar{U}\right) + c^{\theta}\left(U \cap X, U \cap \bar{X}\right)$$

From Eq. (2) and $\left(U \cap X, U \cap \bar{X}\right) \subseteq \left(U \cap X,\ \bar{X}\right)$, we conclude that:

$$c^{\theta}\left(U \backslash X,\ \overline{U \backslash X}\right) \leq c^{\theta}\left(U,\ \bar{U}\right).$$

□

From this point, the correctness of GHW algorithm follows the correctness of the HW algorithm.

## 6 Identifying protein complexes

Here, we address the problem of identifying protein complexes in yeast two-hybrid protein–protein interaction (Y2H PPI) networks solved by the algorithm proposed in Mitrofanova et al. (2009) that uses cut-trees. We show that it is possible to save computational effort by incorporating the proposed GHW algorithm with a modification, along with a new theoretical result about reusable cuts, into the Mitrofanova et al. algorithm.

Y2H PPI networks comprise proteins (nodes) and interactions of the type Y2H (edges). In the view of graph theory, a PPI network is an undirected and unweighted network. When there is an identification of two adjacent proteins that do not belong to the same protein complex, we say that a False Positive (FP) occurs, while when two proteins from the same complex do not share an edge, we say that a False Negative (FN) occurs.

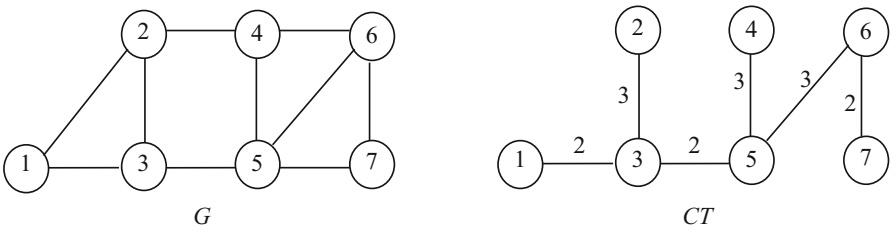


**Fig. 10** Network $G$ (left) and its cut-tree $CT$ (right)

The correctly identification of protein complexes in Y2H PPI networks is still a challenge, as experimental results suffer from a high rate of errors, i.e., high amount of FP and FN occurrences.

Identification approaches based on graph theory and networks analysis usually look for densely connected or clique-like protein complexes. Therefore, high FP and FN rates may interfere or even preclude the search for such complexes, either by creating regions of "false" density, through FP, or by hiding clique-like complexes, through FN.

To overcome the difficulties that come with FPs and FNs, Mitrofanova et al. (2009) exploited the Gomory Hu cut-tree structure to identify the protein complexes in Y2H PPI networks. Those authors observed that, even when a complex loses some edges, the proteins of the complex remain connected by enough paths in the network. For that reason, they used the number of edge-disjoint paths as a connectivity measure between proteins. In a Y2H PPI network, the number of edge-disjoint paths between a pair of proteins corresponds to the maximum flow value between that pair, and between all pairs of $n$ proteins, it corresponds to a cut-tree over the $n$ proteins.

The algorithm developed by Mitrofanova et al. (2009) begins by calculating a cut-tree for a given Y2H PPI network. Then, it removes the edges with least capacity from the cut-tree, i.e., it removes the cuts, represented by these edges, from the network. These two steps are applied, recursively, in each connected component induced by the removal of the cuts until there are no more edges in the connected components.

By way of example, we apply the algorithm of Mitrofanova et al. on the network $G$ illustrated in Fig. 10, where we can also find the result of the first step, the generated cut-tree $CT$.

In the second step, the algorithm identifies the edges [1, 3], [3, 5] and [6, 7] as the minimum cut edges in $CT$. Therefore, the edges [1, 2], [1, 3], [2, 4], [3, 5], [5, 7] and [6, 7] are removed from $G$, generating the connected components $G_1$ and $G_2$, illustrated in Fig. 11, along with theirs cut-trees $CT_1$ and $CT_2$. At this point, the algorithm continues the execution on $G_1$ and $G_2$, until there is no more connected component with edges.

At each phase, the algorithm of Mitrofanova et al. computes the cut-tree of each connected component without reusing $(s-t)$ minimum cuts from previously computed cut-trees. We already observed in Sect. 4 that some statements were made about reusable cuts when the capacity of some edges decreases or when some edges are deleted from the network. Next, we show a new theoretical result about reusable cuts

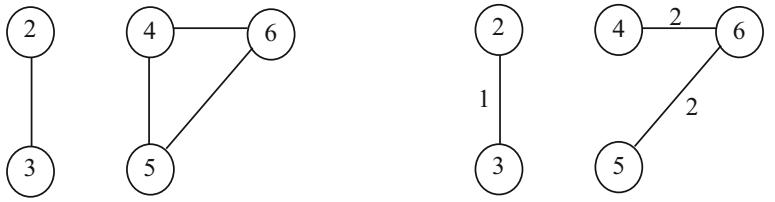


**Fig. 11** Connected components $G_1$ and $G_2$ induced from $G$ (left) and theirs cut-trees $CT_1$ and $CT_2$ (right)

when we consider specifically the deletion of all the edges of a $(s-t)$ minimum cut in an undirected network, as it occurs in the algorithm of Mitrofanova et al. To this end, the introduction of a previous lemma is required.

**Lemma 7** *Let $CT$ be a cut-tree of a connected network $G$ and $C$ and $C_1$ two distinct minimum cuts of $CT$. Then, $c(C \cap C_1)$ is at most* $\min\{\lfloor c(C)/2 \rfloor, \lfloor c(C_1)/2 \rfloor\}$.

***Proof*** Let nodes $s$ and $t$ be the extremities of $C$ and nodes $u$ and $v$ the extremities of $C_1$ in $CT$. Consider the complementary subsets $(H, I, J)$ of the node set of $G$ resulting from the deletion of edges $[s, t]$ and $[u, v]$ from the $CT$. Without loss of generality, let $s \in H, t, u \in I$ and $v \in J$. Then, the capacities of the cuts are:

- $$c(C) = c(H,\ I \cup J) = c(H,\ I) + c(H,\ J)$$
- $$c(C_1) = c(J,\ H \cup I) = c(J,\ I) + c(H,\ J).$$

Assume $c(C \cap C_1)$ is higher than $c(C)/2$. Therefore, $c(H,\ J) > c(H,\ I)$, but then the cut $(I,\ H \cup J)$, which also separates $u$ and $v$, has lower capacity than $C_1$, since:

- $$c(I,\ H \cup J) = c(J,\ I) + c(H,\ I).$$

In this way, $C_1$ is not a $(u-v)$ minimum cut, contradicting the assumption. □

In the case of removing all the edges of a $(s-t)$ minimum cut, Lemma 3 does not guarantee any reusable minimum cut. The reader can understand this through Lemma 7.

In the following, we show a theoretical result on reusable cuts that we will incorporate in the algorithm proposed by Mitrofanova et al.

**Theorem 3** *Let $CT$ be a cut-tree of a connected network $G$ and $C$ and $C_1$ two distinct minimum cuts of $CT$. Let nodes $u$ and $v$ be the extremities of $C_1$ in $CT$. Let $G^0$ be the network obtained when all edges of $C$ have null capacity. If $c(C \cap C_1)$ is equal to* $\min\{\lfloor c(C)/2 \rfloor, \lfloor c(C_1)/2 \rfloor\}$, *then $C_1$ is a $(u-v)$ minimum cut in $G^0$ with capacity decreased by $c(C \cap C_1)$.*

***Proof*** In this proof, we do not consider cuts that cross $C$, once they cannot be $(u-v)$ minimum cuts in $G^0$. We divide this proof into two parts. First suppose $\lfloor c(C)/2 \rfloor \leq \lfloor c(C_1)/2 \rfloor$. Let nodes $s$ and $t$ be the extremities of $C$ in $CT$. Consider the complementary subsets $(H, I, J)$ of the node set of $G$ resulted from the deletion of edges $[s, t]$ and $[u, v]$ from the $CT$. Without loss of generality, let $s \in H$, $t, u \in I$ and $v \in J$. Consider also the complementary subsets $(H, I_2, J_2)$ of the node set of $G$ where $u \in I_2$ and $v \in J_2$. Let $C_2$ be a cut in $G$ that separates $u$ and $v$ and does not intersect $C$. Then, the capacity of $C_1$ and $C_2$ can be written as:

- $$c(C_1) = c(J,\ H \cup I) = c(J,\ I) + c(H,\ J)$$

- $$c(C_2) = c(J_2,\ H \cup I_2) = c(J_2,\ I_2) + c(H,\ J_2).$$

Since in $G^0$ $c(H,\ J) = c(H,\ J_2) = 0$, assume $c(J_2,\ I_2) < c(J,\ I)$ so that $C_1$ would not be a $(u-v)$ minimum cut in $G^0$. As $C_1$ is a $(u-v)$ minimum cut in $G$, then $c(C_1) \leq c(C_2)$; therefore, $c(H,\ J_2) > c(H,\ J)$. By the assumption $\lfloor c(C)/2 \rfloor \leq \lfloor c(C_1)/2 \rfloor$, $c(H,\ J) = \lfloor c(C)/2 \rfloor$, so $c(H,\ J_2) > c(C)/2$. Hence, $c(H,\ I_2) < c(C)/2$, since the capacity of $C$ can be written as:

- $$c(C) = c(H,\ I_2) + c(H,\ J_2).$$

Thus, $c(H,\ I_2) \leq c(H,\ J)$, but the cut $(I_2,\ H \cup J_2)$, which also separates $u$ and $v$, has lower capacity than $C_1$, since:

- $$c(I_2,\ H \cup J_2) = c(J_2,\ I_2) + c(H,\ I_2).$$

In this way, $C_1$ is not a $(u-v)$ minimum cut, contradicting the assumption.

In this second part, suppose $\lfloor c(C)/2 \rfloor > \lfloor c(C_1)/2 \rfloor$; hence, $c(C) > c(C_1)$. Let $s$ and $t$, $(H, I, J)$, $(H, I_2, J_2)$ and $C_2$ be the nodes, the complementary subsets and the cut defined in the beginning of the proof. Again, the capacity of $C_1$ and $C_2$ can be written as:

- $$c(C_1) = c(J,\ H \cup I) = c(J,\ I) + c(H,\ J)$$

- $$c(C_2) = c(J_2,\ H \cup I_2) = c(J_2,\ I_2) + c(H,\ J_2).$$

Since in $G^0$ $c(H,\ J) = c(H,\ J_2) = 0$, assume $c(J_2,\ I_2) < c(J,\ I)$ so that $C_1$ would not be a $(u-v)$ minimum cut in $G^0$. As $c(C_1) \leq c(C_2)$, then $c(H,\ J_2) > c(H,\ J)$. By the assumption $\lfloor c(C)/2 \rfloor > \lfloor c(C_1)/2 \rfloor$, $c(H,\ J) = \lfloor c(C_1)/2 \rfloor$, so $c(J,\ I) = \lceil c(C_1)/2 \rceil$; then:

$$c(J_2,\ I_2) \leq c(H,\ J). \tag{3}$$

Therefore:

$$c(J_2,\ I_2) < c(H,\ J_2). \tag{4}$$

Now there are two possibilities regarding node $t$. With $t \in I_2$, consider the capacities of the cuts $(I_2,\ H \cup J_2)$ and $C$:

- $$c(I_2,\ H \cup J_2) = c(J_2,\ I_2) + c(H,\ I_2)$$
- $$c(C) = c(H,\ I_2) + c(H,\ J_2).$$

From (4), $(I_2,\ H \cup J_2)$ is a cut that has less capacity than $C$ and separates nodes $s$ and $t$. In this way $C$ is not a $(s-t)$ minimum cut, contradicting the assumption.

With $t \in J_2$, $C_2 \geq C$, then:

$$c(J_2,\ I_2) + c(H,\ J_2) \geq c(H,\ I_2) + c(H,\ J_2)$$

Therefore:

$$c(J_2,\ I_2) \geq c(H,\ I_2) \tag{5}$$

From (3) and (5), we have $c(H,\ I_2) \leq c(H,\ J)$; thus, the cut $(I_2,\ H \cup J_2)$, which separates $u$ and $v$, has less capacity than $C_1$, since we assumed $c(J_2,\ I_2) < c(J,\ I)$. In this way, $C_1$ is not a $(u-v)$ minimum cut, contradicting the assumption. □

Analyzing Theorem 3 in the context of Mitrofanova et al.'s algorithm, we are only concerned about the capacity of cut $C$ because $c(C) < c(C_1)$ always. If $c(C) = 1$, then $\lfloor c(C)/2 \rfloor = 0$, which means that every $C_1$ is reusable on $G^0$. This case was already mentioned in Mitrofanova et al. (2009). If $c(C) = 2$ or $c(C) = 3$, then $\lfloor c(C)/2 \rfloor = 1$; therefore, if $c(C \cap C_1) = 1$, $C_1$ is a reusable cut on $G^0$ with capacity decreased by one. For $c(C) = 4$ or $c(C) = 5$, if $c(C \cap C_1) = 2$, $C_1$ is reusable with capacity decreased by two, and so forth.

Therefore, based on Theorem 3, and on Lemmas 5 and 6, we can save minimum cut computations in the algorithm of Mitrofanova et al. In the light of this fact, we propose the following procedure:

1. Adapt the GHW algorithm to include the reusable cuts addressed in Theorem 3;
2. Modify the algorithm of Mitrofanova et al. to remove, one at a time, all the least minimum cuts of the cut-tree, instead of removing them all at once;
3. After the removal of each $(s-t)$ minimum cut from the cut-tree, apply the GHW algorithm on both resulting cut-trees, and compute the $(s-t)$ minimum cuts (line 13 of GHW) on the corresponding connected components.

Note that step 2 is necessary to meet the requirement $\bigcap_{x=1}^{k} P_{i,j}^{x} \neq \emptyset$ of the GHW algorithm. In addition, within the algorithm of Mitrofanova et al., one can apply the algorithms of Gomory and Hu or Gusfield to compute the first cut-tree.

Next, we show the potential of the above procedure through an example. Let the network $G$ and its cut-tree $CT$ exist as in Fig. 12. By the Mitrofanova et al. algorithm, we should remove the edges in $G$ that compose the cut [8, 9] of $CT$. Then, the edges [5, 9] and [8, 9] of $G$ are deleted, generating the network $G'$. At this time, the procedure applies the GHW algorithm to $G$, $G'$ and $CT$ to compute dynamically the new cut-tree $CT'$.

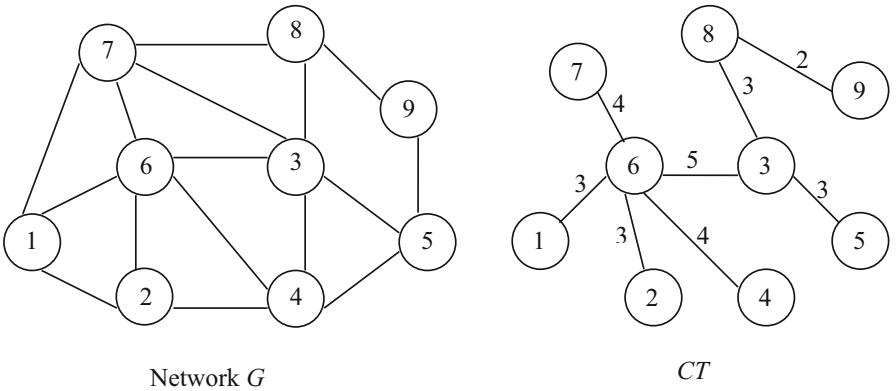


**Fig. 12** Network $G$ and its cut-tree $CT$

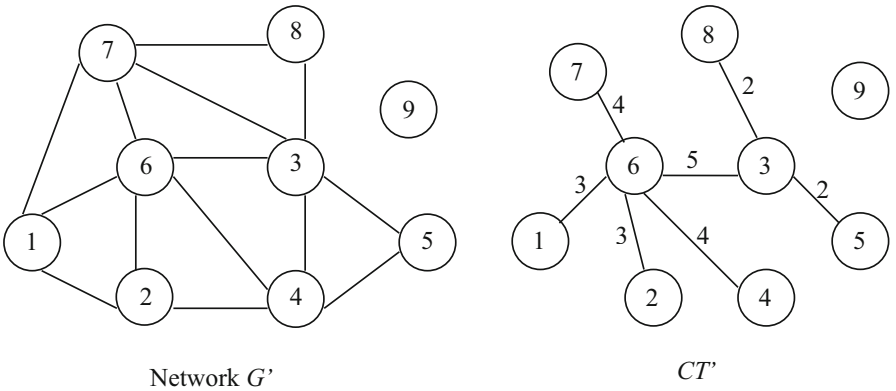


**Fig. 13** Network $G'$ and its cut-tree $CT'$

With the inclusion of Theorem 3 in the routine, the GHW algorithm marks as valid the cuts [3, 5] and [3, 8] and decreases their capacities by one. This is possible because edge [5, 9] belongs to the cuts [3, 5] and [3, 8] in $CT$.

GHW continues computing a minimum cut $(U, \bar{U})$ between nodes 3 and 6 in $G'$. Since $c[6, 3] = c(U, \bar{U})$, GHW marks the cut [3, 6] as valid and executes the command lines 16, 17 and 18. In line 18, the algorithm marks as valid the cuts [1, 6], [2, 6], [4, 6] and [6, 7] and finishes its execution. The output, $G'$ and $CT'$, is illustrated in Fig. 13.

For this case, while the original Mitrofanova et al. algorithm would perform 7 minimum cut computations, the proposed procedure executed only one minimum cut computation.

## 7 Conclusion

In this work, we constructed an overview of the theory of sensitivity analysis on multi-terminal maximum flows, considering the capacity variation occurring in a single edge

of a given network. Subsequently, we extended this theory to cover capacity variations in more than one edge in the network.

In view of the obtained theoretical results, we presented two approaches to compute cut-trees dynamically, considering increasing and decreasing variations on edge capacities. The first one is a straight modification introduced into the Gomory and Hu algorithm, and the second one results in a generalization of the Hartmann and Wagner algorithm.

Additionally, we introduced a new theoretical result on reusable cuts and the GHW algorithm with a modification on the command lines of the algorithm of Mitrofanova et al. to improve the identification of protein complexes in Y2H PPI networks. The proposed changes in the existent algorithm were able to reduce the minimum cut computations required in the identification of these complexes.

For future research, the following questions remain open: What is the maximum number of parametric edges for which the dynamic computation of cut-trees remains worthwhile? Under what circumstances can additional reusable cuts be found? For decreasing variations, can we compute a cut-tree dynamically when $\bigcap_{x=1}^{k} P_{i,j}^{x} = \emptyset$?

## References

Agarwal K, Arora SR (1976) Synthesis of multi-terminal communication nets: finding one or all solutions. IEEE Trans Circuits Syst 23(3):141–146

Barth D, Berthomé P, Diallo M, Ferreira A (2006) Revisiting parametric multi-terminal problems: maximum flows, minimum cuts and cut-tree computations. Discret Optim 3:195–205

Bhalgat A, Hariharan R, Kavitha T, Panigrahi D (2007) An $\tilde{O}(mn)$ Gomory–Hu tree construction algorithm for unweighted graphs. In: STOC'07, pp 605–614

Diallo M (2011) Méthodes d'Optimisation Appliquées aux Réseaux de Flots et Télécoms. Editions universitaires europeennes

Elmaghraby SE (1964) Sensitivity analysis of multi-terminal flow networks. Oper Res 12(5):680–688

Ford LR, Fulkerson DR (1973) Flows in networks. Princeton University Press, Princeton

Goldberg AV, Tsioutsiouliklis K (2001) Cut-tree algorithms: an experimental study. J Algorithms 38:51–83

Gomory RE, Hu TC (1961) Multi-terminal network flows. SIAM J Comput 9(4):551–570

Gusfield D (1990) Very simple methods for all pairs network flow analysis. SIAM J Comput 19:143–155

Hartmann T, Wagner D (2013) Dynamic Gomory–Hu tree construction—fast and simple. CoRR volume arXiv:1310.0178

Hu TC, Shing MT (2002) Combinatorial algorithms, enlarged, 2nd edn. Dover Publications, Inc., Mineola

Liu X, Lin H, Tian Y (2011) Segmenting webpage with Gomory–Hu tree based clustering. J Softw 6(12):2421–2425

Mitrofanova A, Farach-Colton M, Mishra B (2009) Efficient and robust prediction algorithms for protein complexes using Gomory–Hu trees. Pac Symp Biocomput 14:215–226

Scutellà M (2007) A note on the parametric maximum flow problem and some related reoptimization issues. Ann Oper Res 150(1):231–244

## Affiliations

**João Paulo de Freitas Araujo[1] · Madiagne Diallo[1] · Fernanda Maria Pereira Raupp[2] · Pascal Berthomé[3] · José Eugenio Leal[1]**

Madiagne Diallo
madiagne.diallo@gmail.com

Fernanda Maria Pereira Raupp
fernanda@lncc.br

Pascal Berthomé
pascal.berthome@insa-cvl.fr

José Eugenio Leal
jel@puc-rio.br

1 Departamento de Engenharia Industrial, PUC-Rio, Rua Marquês de São Vicente, 225, Gávea, Rio de Janeiro, RJ 22451-900, Brazil

2 Laboratório Nacional de Computação Científica, Av. Getúlio Vargas, 333, Quitandinha, Petrópolis, RJ CEP 25651-075, Brazil

3 INSA Centre Val de Loire, 88 Boulevard Lahitolle, 60013 – 18022 Bourges Cedex, France