# *Minimum $T$-cuts and optimal $T$-pairings*

Romeo Rizzi*

March 9, 1998

*Dipartimento di Matematica Pura ed Applicata, Università di Padova
Via Belzoni 7, 35131 Padova, Italy
romeo@euler.math.unipd.it

**Abstract**

We give a new, simple algorithm for finding a minimum $T$-cut. Our algorithm is more convenient than the ones proposed and employed until now in respect of both difficulty of implementation and time complexity. We introduce the notion of $T$-pairing and obtain a purely combinatorial min-max characterization for the size of a minimum $T$-cut.

**Key words**: $T$-cut, minimum $T$-cut, $T$-pairing, Gomory-Hu tree.

## 1 Introduction

Given a graph $G$ and a node set $S \subseteq V(G)$, the *cut* $\delta_G(S)$ (or more simply $\delta(S)$, if no confusion can arise) is the set of those edges in $E(G)$ with precisely one endnode in $S$.

A graft $(G, T)$ is a graph $G(V, E)$ in which an even number of nodes $T \subseteq V$ have been distinguished as *odd*. The *$T$-parity* of a set of nodes $S \subseteq V$ is the parity of $|S \cap T|$. When $S \subseteq V$ is $T$-odd then $\delta(S)$ is a *$T$-odd cut* or *$T$-cut*. Let $(G, T)$ be a graft and $c : E \mapsto \mathbb{R}_+$ be a *capacity* function. A minimum $T$-cut for $(G, T, c)$ is a $T$-cut $\delta(X)$ of $(G, T)$ for which:

$$c(\delta(X)) = \lambda_{G,T} = \min\{c(\delta(S)) : \delta(S) \text{ is a } T\text{-cut of } (G, T)\}$$

where the capacity $c(F)$ of a set $F$ of edges is defined as $\sum_{e \in F} c(e)$.

The minimum $T$-cut problem is of both theoretical and practical significance. Efficient algorithms for finding minimum $T$-cuts are the means by which we can actually separate blossom inequalities for the matching polytope. Moreover, minimum $T$-cuts procedures are at present employed in several "state of the art" branch and cut algorithms for different and relevant problems, like TSP. To say more, such procedures are often among the most expensive components of these branch and cut projects both in terms of programmer time and resource usage.

Given two nodes $u, v$ of $G$, denote by $\lambda_G(u, v) = \lambda_{G,\{u,v\}}$ the capacity of a minimum $\{u, v\}$-cut. In this paper we will consider two combinatorial problems:

- Finding a minimum $T$-cut for a given $(G, T, c)$.
- Pairing up the nodes in $T$ as $T = \{s_1, t_1; s_2, t_2; \ldots\}$ in such a way that $min\{\lambda_G(s_i, t_i) : i = 1, \ldots, \frac{|T|}{2}\}$ is as big as possible.

In Section 3, we give a new, simple algorithm for finding a minimum $T$-cut. In Section 4, the two combinatorial problems introduced above are linked by a min-max result. We emphasize that this min-max result is in fact the only purely combinatorial "good characterization" of minimum $T$-cuts noticed until now. In Section 5, we introduce Gomory-Hu trees (see [3]) and show how Gomory-Hu trees are commonly used for finding minimum $T$-cuts (see [7] and [5]). Gomory-Hu trees are then employed to give a shorter proof of our min-max characterization for the size of a minimum $T$-cut.

## 2 Switching Cuts

The complement in $V$ of $S \subseteq V$ is denoted by $\overline{S} = V \setminus S$. *Switching* $S$ means replacing $S$ by $\overline{S}$. For example, if $S = X$, then after switching $S$ we obtain that $S = \overline{X}$ and $\overline{S} = X$.

Let $(G, T)$ be a graft and $S \subseteq V$ a set of nodes.

**Observation 2.1** *Switching $S$ does not change the $T$-parity of $S$ (nor of $\delta(S)$).*

*explanation:* $|S \cap T|$ and $|\overline{S} \cap T|$ have the same parity since $|T|$ is even. □

**Proposition 2.2** *Let $(G, T)$ be a graft and $S, X \subseteq V$. Then we have:*

*i) Switching $S$ changes the $T$-parity of $S \cap X$ if and only if $X$ is $T$-odd.*

*ii) $S \cap X$ and $S \cup X$ have the same $T$-parity if and only if $S$ and $X$ have the same $T$-parity.*

*iii) Switching $S$ changes the $T$-parity of $S \cup X$ if and only if $X$ is $T$-odd.*

*proof:* Note that $|(S \cap X) \cap T| = |S \cap (X \cap T)|$ whose parity is affected by switching $S$ if and only if $|X \cap T|$ is odd, that is $X$ is $T$-odd. This implies $i)$.
To obtain $ii)$ note that $|(S \cap X) \cap T| + |(S \cup X) \cap T| = |S \cap T| + |X \cap T|$.
To conclude note that $iii)$ is a consequence of $i)$ and $ii)$. □

It is well known that the cuts of a graph enjoy a property called *submodularity*. Submodularity is at the heart of the Gomory-Hu tree algorithm but is also crucial for our new minimum $T$-cut algorithm to be described in the next section.

**Lemma 2.3** *Let $G$ be a graph with capacity function $c : E \mapsto \mathbb{R}_+$. Let $S_1, S_2 \subseteq V$.*

$$c(\delta(S_1 \cap S_2)) + c(\delta(S_1 \cup S_2)) \leq c(\delta(S_1)) + c(\delta(S_2)) \tag{1}$$

*Proof:* We claim that every edge $uv$ of $G$ contributes to the right side of (1) not less than to the left side.
By Proposition 2.2 if $S_1 \cap S_2$ and $S_1 \cup S_2$ have different $\{u,v\}$-parities so do $S_1$ and $S_2$. Hence, if the claim above were false, then both $S_1 \cap S_2$ and $S_1 \cup S_2$ would be $\{u,v\}$-odd. Assume therefore w.l.o.g. that $u \in S_1 \cap S_2$ and $v \notin S_1 \cup S_2$. This implies $u \in S_1, S_2$ and $v \notin S_1, S_2$. □

Inequality 1 is called *the submodular inequality.* A second fundamental property of cuts is the following:

**Lemma 2.4** *For every three nodes $u, v, z \in V$, $\lambda_G(u,v) \geq \min\{\lambda_G(u,z), \lambda_G(z,v)\}$.*

*Proof:* Every $\{u,v\}$-cut is either $\{u,z\}$-odd or $\{z,v\}$-odd. □

Lemma 2.4 has the following generalization:

**Lemma 2.5** *Let $v_1, \ldots, v_n$ be nodes in $V$. Then $\lambda_G(v_1, v_n) \geq \min_{1 \leq i \leq n-1} \lambda_G(v_i, v_{i+1})$.*

# 3 Minimum $T$-cuts: a new algorithm

Given a graft $(G,T)$ and a $T$-even set $S \subseteq V(G)$, denote by $G_S$ the graph obtained from $G$ by identifying all nodes in $S$ into a single node and let $T_S = T \setminus S$. Note that $(G_S, T_S)$ is a graft. When $S = \{s,t\} \subseteq T$ then we rely on a shorter notation $G_{s,t} = G_{\{s,t\}}$ and $T_{s,t} = T_{\{s,t\}}$.

Since node identification does not affect the edge set of a graph, a capacity function $c$ for $G$ is also a capacity function for $G_S$ and $G_{s,t}$. Two sets $S, X \subseteq V$ are said *crossing* if $S \cap X \neq \emptyset$ for every possible switching of $S$ and $X$.

The following algorithm computes $\lambda_{G,T}$:

**Algorithm 1** MIN_T-CUT $(G,T,c)$

1. if $T = \emptyset$ then return $\infty$; *comment:* $(G,T)$ contains no $T$-cut
2. let $s$ and $t$ be any two different nodes in $T$;
3. let $\delta(S)$ be a minimum $\{s,t\}$-cut;
4. if $S$ is $T$-odd then return $\min\{c(\delta(S)), \text{MIN\_T-CUT } (G_{s,t}, T_{s,t}, c)\}$;
   else return $\min\{\text{MIN\_T-CUT } (G_S, T_S, c), \text{MIN\_T-CUT } (G_{\overline{S}}, T_{\overline{S}}, c)\}$;

### Correctness

For a given $(G,T,c)$, let $s$ and $t$ be any two different nodes in $T$ and let $\delta(S)$ be a minimum $\{s,t\}$-cut. The correctness of the above procedure relies on the following two lemmas:

**Lemma 3.1** *If $\delta(S)$ is $T$-odd then $\lambda_{G,T} = \min\{c(\delta(S)), \lambda_{G_{s,t},T_{s,t}}\}$.*

*Proof:* This follows from the observation that the $T_{s,t}$-cuts of $G_{s,t}$ are precisely the $T$-cuts of $G$ that are not $\{s,t\}$-odd. □

**Lemma 3.2** *If $\delta(S)$ is $T$-even then $\lambda_{G,T} = \min\{\lambda_{G_S,T_S}, \lambda_{G_{\overline{S}},T_{\overline{S}}}\}$.*

*Proof:* First note that to every $T_S$-cut in $(G_S, T_S)$ and to every $T_{\overline{S}}$-cut in $(G_{\overline{S}}, T_{\overline{S}})$ corresponds a $T$-cut in $(G,T)$ of the same capacity. This implies $\lambda_{G,T} \leq \min\{\lambda_{G_S,T_S}, \lambda_{G_{\overline{S}},T_{\overline{S}}}\}$.

For the converse, let $\delta(X)$ be any minimum $T$-cut for $(G,T,c)$. If $S, X$ are not crossing then switch $S$ and $X$ so that $S \cap X = \emptyset$. So $\delta_G(X)$ is a $T$-cut in $(G_S, T_S)$ and $\lambda_{G_S,T_S} \leq \lambda_{G,T}$.

Assume therefore that $S$ and $X$ are crossing. Remember that $S$ is $\{s,t\}$-odd and $T$-even whereas $X$ is $T$-odd by assumption. By Proposition 2.2, switching $S$ changes the $T$-parity of $S \cap X$ whereas switching $X$ changes the $\{s,t\}$-parity of $S \cup X$ leaving the $T$-parity of $S \cap X$ unaffected. By possibly switching $S$ we can assume that $S \cap X$ is $T$-odd. By possibly switching $X$ we can assume that $S \cup X$ is $\{s,t\}$-odd. (without changing the $T$-parity of $S \cap X$).

At this point $c(\delta(S \cap X)) \geq c(\delta(X))$ for $\delta(X)$ is a minimum $T$-cut and $c(\delta(S \cup X)) \geq c(\delta(S))$ for $\delta(S)$ is a minimum $\{s,t\}$-cut. By submodularity (see Lemma 2.3) $c(\delta(S \cap X)) = c(\delta(X))$. Thus $\delta(S \cap X)$ is a minimum $T$-cut. Moreover $S$ and $S \cap X$ do not cross. □

### Time Complexity

Each time $S$ is $T$-odd the cardinality of $T$ is reduced by 2. Each time $S$ is $T$-even the set $T$ is partitioned into two non-empty subsets $T \cap S$ and $T \cap \overline{S}$. Thus the number of recursions is at least $|T|/2$ and at most $|T| - 1$. To conclude the algorithm performs at least $|T|/2$ and at most $|T| - 1$ max-flow min-cut computations. The algorithm based on the Gomory-Hu tree, which is the method employed until now to find minimum $T$-cuts, performs always $|T| - 1$ max-flow min-cut computations.

Call a pair $\{s,t\}$ *useful* if there exists a minimum $\{s,t\}$-cut which is $T$-odd. At present, the most convenient methods to find minimum $\{s,t\}$-cuts actually compute a maximum flow from $s$ to $t$. Thus all such methods require only a small amount of extra work to return a minimum $\{s,t\}$-cut which is $T$-odd whenever $\{s,t\}$ is useful. Assume nodes $s$ and $t$ in step 2. of algorithm *Min_T-cut* are choosen at random and with uniform probability. By Lemma 5.2, $\{s,t\}$ is useful with probability at least $\frac{1}{|T|}$. Exploiting this fact and considering the less favourable case (whenever $|T| > 2$, either $|T \cap S| = 2$ or $|T \cap \overline{S}| = 2$) we can easily see that the probability that algorithm *Min_T-cut* will perform as many as $|T| - 1$ max-flow min-cut computations is at most $\left(\frac{|T|-1}{|T|}\right)\left(\frac{|T|-3}{|T|-2}\right)\cdots\left(\frac{3}{4}\right)\left(\frac{1}{2}\right) \leq \frac{1}{\sqrt{|T|}}$ and tends therefore to 0 when $|T|$ grows. Even more, for any constant $K$, the probability that algorithm *Min_T-cut* will perform more than $|T| - K$ max-flow min-cut computations tends to 0 when $|T|$ grows.

## 4 $T$-pairings: a good characterization for the size of a minimum $T$-cut

Let $(G, T)$ be a graft with capacity function $c : E \mapsto \mathbb{R}_+$. A *$T$-pairing* is a partition $\mathcal{P}$ of $T$ such that $|P| = 2$ for every class $P$ of $\mathcal{P}$. The classes of a $T$-pairing are called *pairs.*

If $\lambda_G(u, v)$ is the capacity of a minimum $\{u, v\}$-cut then the value of a $T$-pairing $\mathcal{P}$ is defined as:

$$val_G(\mathcal{P}) = \min_{\{u,v\}\in\mathcal{P}} \lambda_G(u, v)$$

**Theorem 4.1** *For every $(G, T, c)$ the maximum value of a $T$-pairing equals the minimum capacity of a $T$-cut.*

*Proof:* Let $\mathcal{P}$ be any $T$-pairing and $\delta(S)$ be any $T$-cut. Since $\delta(S)$ is $T$-odd, $\mathcal{P}$ contains a pair $\{u, v\}$ such that $\delta(S)$ is $\{u, v\}$-odd. Therefore $c(\delta(S)) \geq \lambda_G(u, v) \geq val_G(\mathcal{P})$. We conclude that $\lambda_{G,T}$ is always at least as big as the maximum value of a $T$-pairing.

For the converse, consider a single iteration of algorithm *Min_T-cut*, described in Section 3.

Let $s$ and $t$ be two odd nodes. Let $\delta(S)$ be a minimum $\{s, t\}$-cut. In each iteration the algorithm contemplates two possibilities:

*Case 1:* $\delta(S)$ is $T$-odd. In Section 3 we showed that $\lambda_{G,T} = \min\{c(\delta(S)), \lambda_{G_{s,t},T_{s,t}}\}$. Lemma 4.2 below shows that if $\mathcal{P}'$ is a $T_{s,t}$-pairing with $val_{G_{s,t}}(\mathcal{P}') = \lambda_{G_{s,t},T_{s,t}}$ then $\mathcal{P} = \mathcal{P}' \cup \{\{s, t\}\}$ is a $T$-pairing with $val_G(\mathcal{P}) \geq \min\{c(\delta(S)), val(\mathcal{P}')\} = \min\{c(\delta(S)), \lambda_{G_{s,t},T_{s,t}}\} = \lambda_{G,T}$.

*Case 2:* $\delta(S)$ is $T$-even. In Section 3 we showed that $\lambda_{G,T} = \min\{\lambda_{G_S,T_S}, \lambda_{G_{\overline{S}},T_{\overline{S}}}\}$. Lemma 4.3 below shows that if $\mathcal{P}_S$ is a $T_S$-pairing with $val_{G_S}(\mathcal{P}_S) = \lambda_{G_S,T_S}$ and $\mathcal{P}_{\overline{S}}$ is a $T_{\overline{S}}$-pairing with $val_{G_{\overline{S}}}(\mathcal{P}_{\overline{S}}) = \lambda_{G_{\overline{S}},T_{\overline{S}}}$ then $\mathcal{P} = \mathcal{P}_S \cup \mathcal{P}_{\overline{S}}$ is a $T$-pairing with $val_G(\mathcal{P}) \geq \min\{val_{G_S}(\mathcal{P}_S), val_{G_{\overline{S}}}(\mathcal{P}_{\overline{S}})\} = \min\{\lambda_{G_S,T_S}, \lambda_{G_{\overline{S}},T_{\overline{S}}}\} = \lambda_{G,T}$. □

**Lemma 4.2** *Let $\delta(S)$ be a minimum $\{s, t\}$-cut in $(G, c)$. Then we have:*

$$\lambda_G(u, v) \geq \min\{c(\delta(S), \lambda_{G_{s,t}}(u, v)\} \qquad \forall\, u, v \in V(G) \setminus \{s, t\}$$

*Proof:* This follows from the observation that the $\{u, v\}$-cuts of $G_{s,t}$ are exactly the $\{u, v\}$-cuts of $G$ that are not $\{s, t\}$-odd. □

**Lemma 4.3** *Let $\delta(S)$ be a minimum $\{s, t\}$-cut in $(G, c)$. Then we have:*

$$\lambda_G(u, v) = \lambda_{G_S}(u, v) \qquad \forall\, u, v \in V(G) \setminus S$$

*Proof:* Let $u$ and $v$ be any two nodes in $V(G) \setminus S$. Obviously $\lambda_G(u, v) \leq \lambda_{G_S}(u, v)$. To show the converse let $\delta(X)$ be any minimum $\{u, v\}$-cut in $G$. If $S, X$ are not crossing then switch $X$ so that $S \cap X = \emptyset$. (If $S$ gets switched then $S \cap X \neq \emptyset$ since $X$ is $\{u, v\}$-odd). But then $\delta_G(X)$ is a $\{u, v\}$-cut in $G_S$ and $\lambda_{G_S}(u, v) = \lambda_G(u, v)$.

Since $S$ is $\{s, t\}$-odd we can assume by possibly switching $X$ that $S \cap X$ is $\{s, t\}$-odd. This implies $c(\delta(S\cap X))\geq c(\delta(S))$. By submodularity (see Lemma 2.3) $c(\delta(S\cup X))\leq c(\delta(X))$. Note that $S\cup X$ is $\{u,v\}$-odd since $S\cap\{u,v\}=\emptyset$ and $X$ is $\{u,v\}$-odd. Thus $\delta(S\cup X)$ is a minimum $\{u,v\}$-cut. Moreover $X$ and $S\cup X$ do not cross. $\square$

Now we give an interesting consequence of Theorem 4.1. Given a graft $(G,T)$ let $\mathcal{P}$ be a $T$-pairing of value $\lambda_{G,T}$. Consider the following result of Nash-Williams.

**Theorem 4.4 (Nash-Williams [6])** *An undirected graph has an orientation such that for every ordered pair $(x,y)$ of nodes there are $\lfloor\lambda(x,y)/2\rfloor$ arc-disjoint paths from $x$ to $y$.*

By Theorem 4.4 there exists an orientation $D(V,A)$ of $G(V,E)$ with the property that for every pair $P\in\mathcal{P}$ there are $\lfloor\lambda_{G,T}/2\rfloor$ arc-disjoint paths from whichever node in $P$ to the other. Let $S\subseteq V$ be any $T$-odd set of nodes. Since $S$ is $\{s,t\}$-odd for some pair $\{s,t\}\in\mathcal{P}$ then $d_D^-(S),d_D^+(S)\geq\lfloor\lambda_{G,T}/2\rfloor$.

## 5 Gomory-Hu trees and minimum $T$-cuts

Given a graph $G$ and $c:E\mapsto\mathbb{R}_+$ let $H$ be any tree with $V(H)=V$. To any edge $e$ of $H$ we can associate a set of nodes $S_e\subseteq V$ such that $\{e\}=\delta_H(S_e)$. Tree $H$ is called a *Gomory-Hu tree* if for every edge $e=uv$ of $H$ the cut $\delta_G(S_e)$ is a minimum $\{u,v\}$-cut of $G$. We emphasize that a Gomory-Hu tree $H$ is not required to be a subgraph of $G$, i.e. the fact that two nodes are adjacent in $H$ does not imply that they are adjacent in $G$.
In [3], Gomory and Hu proved that a Gomory-Hu tree always exists by giving an algorithm which constructs such a tree for any pair $(G,c)$. Their algorithm is also polynomial and essentially requires the execution of $n-1$ max-flow min-cut computations in order to obtain a Gomory-Hu tree. In the next section we describe a generalization of the algorithm of Gomory and Hu.

Let $H$ be a tree and $T\subseteq V(H)$ with $|T|$ even. Let $F$ be the set of edges $h$ of $H$ such that $S_h$ is $T$-odd. We call $F$ the *$T$-join of $H$* and observe that since $H$ is connected we have the following:

(i) $|\delta_F(S)|>0$ for every $T$-odd node set $S\subseteq V$.

(ii) the connected components of $(V,F)$ are all $T$-even.

(iii) every node in $T$ is connected to some other node in $T$ by some path in $(V,F)$.

In [7], Padberg and Rao gave a first polynomial time algorithm to compute minimum $T$-cuts. Their algorithm was based on the following observation:

**Lemma 5.1** *Let $H$ be a Gomory-Hu tree for $(G,c)$. Take any $T\subseteq V(G)$ with $|T|$ even. Then there exists an edge $\bar{f}$ of $H$ such that $\delta_G(S_{\bar{f}})$ is a minimum $T$-cut in $(G,T,c)$.*

*Proof:* Let $\delta_G(S)$ be a minimum $T$-cut in $(G,T,c)$. Let $F$ be the $T$-join of $H$. By (i) above, let $\bar{f}=uv$ be any edge in $\delta_F(S)$. We claim that $\delta(S_{\bar{f}})$ is a minimum $T$-cut. In fact $S_{\bar{f}}$ is

$T$-odd since $\bar{f} \in F$. Moreover $\delta_G(S_{\bar{f}})$ is a minimum $\{u,v\}$-cut since $\bar{f}$ is an edge of $H$. So $c(\delta_G(S_{\bar{f}})) \leq c(\delta_G(S))$ since $S$ is $\{u,v\}$-odd. □

By the same arguments, but exploiting (iii) above, one obtains the following:

**Lemma 5.2** *In a graft $(G,T)$ let $s$ be any node in $T$. Then there exists a node $t \in T \setminus \{s\}$ such that $\{s,t\}$ is useful.*

Let $H$ be a Gomory-Hu tree of $G$. To every edge $h$ of $H$ associate a capacity $\bar{c}(h) = c(\delta_G(S_h))$. Lemma 2.5 implies the following relevant property of Gomory-Hu trees:

**Property 5.3** *For every two nodes $u,v \in V$ the minimum capacity of a $\{u,v\}$-cut in $(H,\bar{c})$ equals the minimum capacity of a $\{u,v\}$-cut in $(G,c)$.*

*Proof:* Let $e_1,\ldots,e_k$ be the edges of the path in $H$ from $u$ to $v$. Then $\lambda_{H,\{u,v\}} = \min_{1\leq i\leq k} \lambda_{H,e_i} = \min_{1\leq i\leq k} \lambda_{G,e_i} = \lambda_{G,\{u,v\}}$. □

A shorter proof of Theorem 4.1 in terms of Gomory-Hu trees follows.

*A shorter proof of Theorem 4.1:* Let $F$ be the $T$-join of $H$. By Lemma 5.1 and Property 5.3, $\lambda_{G,T} = \min_{h\in F} \bar{c}(h)$. Obviously every component of $(V,F)$ has an even number of nodes in $T$. Construct a $T$-pairing $\mathcal{P}$ by pairing up odd nodes into a same component of $(V,F)$. By Property 5.3 $val_G(\mathcal{P}) = \lambda_{G,T}$ and Theorem 4.1 follows. □

## 6 X-trees as a generalization of Gomory-Hu trees

Finding a Gomory-Hu tree requires $|V|-1$ max-flow min-cut computations. Thus it is not convenient to construct a whole Gomory-Hu tree when the ultimate goal is to compute a minimum $T$-cut. However, as shown by Lemma 5.1, once a Gomory-Hu tree has been obtained we can quickly locate a minimum $T$-cut for any possible even set $T \subseteq V$. In other words the full computation of the Gomory-Hu tree gives us a way to solve in a single run instances for different sets $T$.

Given a pair $(G,c)$ and some even sets of nodes $T_1,\ldots,T_k \subseteq V$, assume you want to compute a minimum $T_i$-cut in $(G,c)$ for any possible $i \in \{1,\ldots,k\}$. Let $X = \cup_{i=1}^k T_i$. In this section we show how only $|X|-1$ max-flow min-cut computations can suffice.

Given a graph $G$, a capacity function $c : E \mapsto \mathbb{R}_+$ and a set $X \subseteq V$ with $X \neq \emptyset$, let $H$ be any tree whose node set $V(H)$ is a partition of $V(G)$ having precisely one node of $X$ in each class. The class of $V(H)$ containing $x \in X$ is denoted by $[x]$. To any edge $e$ of $H$ we can associate a set $S_e \subseteq V(H)$ such that $\{e\} = \delta_H(S_e)$. Denote by $\langle S_e \rangle$ the node set $\cup_{[x]\in S_e}[x]$. Tree $H$ is called an *X-tree* if for every edge $e = [x][y]$ of $H$ the cut $\delta_G(\langle S_e \rangle)$ is a minimum $\{x,y\}$-cut of $G$.

By obvious modifications, the proofs of Lemma 5.1 and Property 5.3 can be adapted to obtain the following generalizations:

**Lemma 6.1** *Let $H$ be an X-tree for $(G,c)$. Then there exists an edge $\bar{f}$ of $H$ such that $\delta_G(\langle S_{\bar{f}} \rangle)$ is a minimum $T$-cut in $(G,T,c)$ whenever $T \subseteq X$ with $|T|$ even.*

**Property 6.2** *Let $H$ be an X-tree for $(G, X, c)$. To every edge $h$ of $H$ associate a capacity $\bar{c}(h) = c(\delta_G(\langle S_h \rangle))$. Then, for every two nodes $u, v \in X$, the minimum capacity of a $\{[u], [v]\}$-cut in $(H, \bar{c})$ equals the minimum capacity of a $\{u, v\}$-cut in $(G, c)$.*

Let $x$ be any node in $X$. The following algorithm performs $|X| - 1$ max-flow min-cut computations in order to obtain an X-tree.

---

**Algorithm 2** X-TREE $(G, X, c, x)$

---

> if $X = \{x\}$ then return $(\{V(G)\}, \emptyset)$; *comment: a tree of a single node $V(G)$*
> take any $y \in X \setminus \{x\}$;
> let $\delta(S)$ be a minimum $\{x, y\}$-cut in $(G, c)$ (and assume $x \in S$);
> $H_S \leftarrow$ X-TREE $(G_S, X \setminus S, c, y)$;
> $H_{\overline{S}} \leftarrow$ X-TREE $(G_{\overline{S}}, X \setminus \overline{S}, c, x)$;
> let $V_s$ be the class of $V(H_S)$ containing the node $s$ of $G_S$ obtained by contraction of $S$;
> let $V_{\overline{s}}$ be the class of $V(H_{\overline{S}})$ containing the node $\overline{s}$ of $G_{\overline{S}}$ obtained by contraction of $\overline{S}$;
> remove node $s$ from $V_s$ and node $\overline{s}$ from $V_{\overline{s}}$;
> define $\overline{c}(V_s V_{\overline{s}}) := c(\delta(S))$;
> return a tree $H$ with $V(H) = V(H_S) \cup V(H_{\overline{S}})$ and $E(H) = E(H_S) \cup E(H_{\overline{S}}) \cup \{V_s V_{\overline{s}}\}$.

---

**Correctness**

For any edge $h = [x_1][x_2]$ of $H$ we must show that $\delta_G(\langle S_h \rangle)$ is a minimum $\{x_1, x_2\}$-cut of $G$.

Assume first $h = V_s V_{\overline{s}}$. Let $x_1$ be the only node in $V_s \cap X$ and $x_2$ be the only node in $V_{\overline{s}} \cap X$. Since $\delta(S)$ is a minimum $\{x, y\}$-cut in $(G, c)$ then $\delta(s)$ is a minimum $\{s, y\}$-cut in $(G_S, c)$ and $\delta(\overline{s})$ is a minimum $\{\overline{s}, x\}$-cut in $(G_{\overline{S}}, c)$. By repeated applications of Lemma 6.3, $\delta(s)$ is a minimum $\{s, x_1\}$-cut in $G_S$ and $\delta(\overline{s})$ is a minimum $\{\overline{s}, x_2\}$-cut in $G_{\overline{S}}$. Hence $\delta(S)$ is a minimum $\{x_1, x_2\}$-cut in $G$.

Otherwise assume w.l.o.g. that $h = [x_1][x_2]$ is an edge of $H_S$. Let $\langle S_h \rangle_{H_S}$ be the union of the classes of $V(H_S)$ in a same connected component of $H_S \setminus h$. By induction $\delta_{G_S}(\langle S_h \rangle_{H_S})$ is a minimum $\{x_1, x_2\}$-cut in $(G_S, c)$. By Lemma 4.3, $\delta_G(\langle S_h \rangle_H) = \delta_{G_S}(\langle S_h \rangle_{H_S})$ is a minimum $\{x_1, x_2\}$-cut in $(G, c)$.

**Lemma 6.3** *Given a graph $G$, a set $X \subseteq V$, a node $s \in V \setminus X$, and a node $x \in X$ such that $\delta(s)$ is a minimum $\{s, x\}$-cut, let $y$ be a node in $X \setminus \{x\}$. If $\delta(U)$ is a minimum $\{x, y\}$-cut (and assume $s \in U$) then there exists an $x' \in X \cap U$ such that $\delta_{G_{\overline{U}}}(s)$ is a minimum $\{s, x'\}$-cut in $G_{\overline{U}}$.*

*Proof:* If $x \in U$ then $x' = x$ will work. If $x \notin U$ then $d(U) \geq d(s)$ and $y \in U$. By Lemma 2.4, $\lambda_{G_{\overline{U}}}(s, y) \geq \lambda_G(s, y) \geq \min\{\lambda_G(s, x), \lambda_G(x, y)\} \geq d(s)$. Hence $x' = y$ will work. □

**Remark 6.4** *For every $v \in V$,* X-TREE $(G, V, c, v)$ *returns a Gomory-Hu tree for $(G, c)$.*

## References


[1] J. Edmonds and R. Giles, A min-max relation for submodular functions on graphs. Studies in Integer Programming, *Annals of Discrete Mathematics* (P.L.Hammer, et al., eds.) 1 (1977) 185–204.

[2] J. Edmonds and E. Johnson, Matching, Euler tours and the Chinese postman problem. Mathematical Programming 5 (1973), 88–124.

[3] R. Gomory and T.C. Hu, Multi-terminal network flows. SIAM 9 (1961) 551–570.

[4] Mei Gu Guan, Graphic programming using odd or even points. Chinese Mathematics 1 (1962) 273–277.

[5] L. Lovász and M.D. Plummer, *Matching Theory*, Akadémiai Kiadó (1986)

[6] Nash-Williams, On orientation, connectivity and odd vertex pairings in finite graphs. Canad. J. Math. 12 (1960) 555–567.

[7] M.W. Padberg and M.R. Rao, Odd minimum cut-sets and $b$-matchings. Mathematics of Operation Research 7 (1982) 67–80.

[8] A. Sebő, The Schrijver system of odd join polyhedra. Combinatorica 8 (1988) 103–116.

[9] A. Sebő, Undirected distances and the postman-structure of graphs. Journal of Combinatorial Theory (B) 49 (1990) 10–39.

[10] P.D. Seymour, Odd cuts and plane multicommodity flows. Proceedings of the London Mathematical Society 42 (1981) 178–192.