A Simple Minimum T-Cut Algorithm

Romeo Rizzi*

October 23, 2002

Dipartimento di Informatica e Telecomunicazioni, Università di Trento
via Sommarive 14, 38050 Povo, Italy
romeo@science.unitn.it

Abstract

We give a simple algorithm for finding a minimum T-cut. At present, all known efficient algorithms for this problem go through the computation of a Gomory-Hu tree. While our algorithm bases on the same fundamental properties of uncrossing as the previous methods, still it provides an ad-hoc solution. This solution is easier to implement and faster to run. Our results extend to the whole of symmetric submodular functions.

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

1 Introduction

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

A *graft* (G, T) is a connected graph G 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 $\mathbb{R}_+$ and $\mathbb{N}$ denote the set of non-negative reals and the set of non-negative integers (i.e. naturals), respectively. Let (G, T) be a graft and $c : E \mapsto \mathbb{R}_+$ be a *cost* 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 cost $c(F)$ 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.

In Section 3, we give a simple algorithm to find a minimum T-cut. At present, all known efficient algorithms for this problem go through the computation of a Gomory-Hu tree. While

*Research carried out with financial support of the project TMR-DONET nr. ERB FMRX-CT98-0202 of the European Community.

our algorithm bases on the same fundamental properties and techniques of submodularity and uncrossing as the previous methods, still it provides an ad-hoc solution. This solution is easier to implement and faster to run. In [4], we gave a min-max formula which links the minimum T-cut problem to the problem of pairing up the nodes in T as $T = \{s_1, t_1; s_2, t_2; \ldots\}$ so that $min\{\lambda_{G,\{s_i,t_i\}} : i = 1, \ldots, \frac{|T|}{2}\}$ is greatest possible. In Section 4, we show that our algorithm also computes such an optimal pairing, hence providing an algorithmic proof of the min-max formula. In Section 2, we point out the crucial properties of cuts in graphs which are needed in the following. These properties state the cut function to be symmetric and submodular. Therefore, even if we have decided to confine our exposition to cuts in graphs, our results and arguments apply, without any modification, to the whole of symmetric submodular functions.

2 Basic facts: submodularity and uncrossing

The background and also the main ingredients and techniques of our simple solutions can be found in [1, 3, 2]. In this section, we recall the only basic facts which are needed. The reader willing to accept Lemma 2.4 here below can skip this section altogether, and avoid getting involved into our somewhat unconventional (but we believe convenient) view on submodularity.

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, if and only if X is T-odd. This gives i).
To obtain ii) note that $|(S \cap X) \cap T| + |(S \cup X) \cap T| = |S \cap T| + |X \cap T|$.
Finally, iii) is a consequence of i) and ii). □

The following lemma expresses a property of cuts known as *submodularity.*

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

$$c(\delta(S_1 \cap S_2)) + c(\delta(S_1 \cup S_2)) \leq c(\delta(S_1)) + c(\delta(S_2)) \quad (1)$$

Proof: We claim that each edge uv contributes to the right at least as to the left side of (1).

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, had our claim to be false, then both $S_1 \cap S_2$ and $S_1 \cup S_2$ would be $\{u,v\}$-odd. Assume w.l.o.g. that $u \in S_1 \cap S_2$ and $v \notin S_1 \cup S_2$. But then, $u \in S_1, S_2$ and $v \notin S_1, S_2$. $\square$

If $S \cap X \neq \emptyset$ for every possible switching of S and X, then S and X are said to *cross*. All what we will need about cut functions is that they obey to the following lemma.

Lemma 2.4. *Let T_1, T_2 be even cardinality subsets of V. Let $\delta(S_1)$ be a minimum T_1-cut and assume that S_1 is T_2-even. Then there exists a minimum T_2-cut $\delta(S_2)$ such that S_1 and S_2 do not cross.*

Proof: Let $\delta(X)$ be a minimum T_2-cut. We remark that, by Proposition 2.2, switching S_1 changes the T_2-parity of $S_1 \cup X$ whereas switching X changes the T_1-parity of $S_1 \cap X$ leaving the T_2-parity of $S_1 \cup X$ unaffected. Therefore, by possibly switching S_1, we can assume that $S_1 \cup X$ is T_2-odd. Afterwards, by possibly switching X, we can assume that $S_1 \cap X$ is T_1-odd without affecting the T_2-parity of $S_1 \cup X$.

At this point, $c(\delta(S_1 \cap X)) \geq c(\delta(S_1))$ since $\delta(S_1)$ is a minimum T_1-cut. By submodularity, $c(\delta(S_1 \cup X)) = c(\delta(X))$. Thus $\delta(S_1 \cup X)$ is a minimum T_2-cut. And clearly, S_1 and $S_1 \cup X$ do not cross. $\square$

3 Minimum T-cuts: a simple algorithm

Given a graft (G,T) and a T-even set $S \subseteq V$, denote by G_S the graph obtained from G by identifying all nodes in S into a single node and letting $T_S := T \setminus S$. 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 cost function c for G is also a cost function for G_S and $G_{s,t}$.

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

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

1. if $T = \emptyset$ then return ∞; *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: Indeed, 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 every T_S-cut in (G_S, T_S) and every $T_{\overline{S}}$-cut in $(G_{\overline{S}}, T_{\overline{S}})$ is also a T-cut in (G, T). 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). By Lemma 2.4, we can assume that S and X do not cross. This means that the edge set $\delta_G(X)$ is either a T_S-cut in G_S or a $T_{\overline{S}}$-cut in $G_{\overline{S}}$. □

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 pretty involved 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 chosen at random and with uniform probability. By Lemma 4.4, $\{s, t\}$ is useful with probability at least $\frac{1}{|T|-1}$. Exploiting this fact, 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

$$\left(\frac{|T|-2}{|T|-1}\right)\left(\frac{|T|-4}{|T|-3}\right)\cdots\left(\frac{4}{5}\right)\left(\frac{2}{3}\right) \leq \sqrt{\left(\frac{|T|-1}{|T|}\right)\left(\frac{|T|-2}{|T|-1}\right)\cdots\left(\frac{3}{4}\right)\left(\frac{2}{3}\right)} = \sqrt{\left(\frac{2}{|T|}\right)}$$

and tends therefore to 0 when $|T|$ grows. (Just observe that, where $\delta(S)$ is the cut considered in step 3., then the less favorable case in bounding this probability occurs when $|T \cap S| = 2$ or $|T \cap \overline{S}| = 2$ whenever $|T| > 2$).

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 Computing optimal T-pairings

Let (G,T) be a graft with cost function $c : E \mapsto \mathbb{R}_+$. A *$T$-pairing* is a partition of T into pairs. The value of a T-pairing $\mathcal{P}$ is defined as:

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

where $\lambda_G(u,v)$ denotes the cost of a minimum $\{u,v\}$-cut. 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, $c(\delta(S)) \geq \lambda_G(u,v) \geq val_G(\mathcal{P})$ and the value of $\mathcal{P}$ is a lower bound on $\lambda_{G,T}$.

In this section, we show that algorithm *Min_T-cut* actually finds a T-pairing of value $\lambda_{G,T}$. Indeed, consider a single iteration of the algorithm. 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. By Lemma 3.1, $\lambda_{G,T} = \min\{c(\delta(S)), \lambda_{G_{s,t},T_{s,t}}\}$. By Lemma 4.1 here below, 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. By Lemma 3.2, $\lambda_{G,T} = \min\{\lambda_{G_S,T_S}, \lambda_{G_{\overline{S}},T_{\overline{S}}}\}$. By Lemma 4.2 here below, 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 4.1. *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: Indeed, the $\{u,v\}$-cuts of $G_{s,t}$ are exactly the $\{u,v\}$-cuts of G that are not $\{s,t\}$-odd. □

Lemma 4.2. *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 $\lambda_G(u,v) \leq \lambda_{G_S}(u,v)$. For the converse, let $\delta(X)$ be any minimum $\{u,v\}$-cut in G. By Lemma 2.4, we can assume that S and X do not cross. But then the edge set $\delta_G(X)$ is a $\{u,v\}$-cut in G_S and $\lambda_{G_S}(u,v) = \lambda_G(u,v)$. □

To summarize this section, the above arguments lead to the following results:

Theorem 4.3. *For every (G,T,c) the maximum value of a T-pairing equals the minimum cost of a T-cut.*

Lemma 4.4. *For every node u in T there exists a node $v \in T\setminus\{u\}$ such that $\{u,v\}$ is useful.*

Proof: Apply algorithm *Min_T-cut* to (G,T,c). At each recursion, keep choosing $s = u$ until the minimum $\{s,t\}$-cut $\delta(S)$ is T-odd. By Lemma 4.2, $\{u,t\}$ is useful w.r.t. (G,T,c). □

Acknowledgments

Thanks are due to the anonymous referees of a previous version of this work. Their suggestions and advice significantly helped improving the presentation.

References

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

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

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

[4] R. Rizzi, Minimum T-cuts and optimal T-pairings. Discrete Mathematics. 257 (1) (2002) 177–181.