The Steiner k-Cut Problem*

Chandra Chekuri† Sudipto Guha‡ Joseph (Seffi) Naor§

September 23, 2005

Abstract

We consider the Steiner k-cut problem which generalizes both the k-cut problem and the multiway cut problem. The Steiner k-cut problem is defined as follows. Given an edge-weighted undirected graph $G = (V, E)$, a subset of vertices $X \subseteq V$ called *terminals*, and an integer $k \leq |X|$, the objective is to find a minimum weight set of edges whose removal results in k disconnected components, each of which contains at least one terminal. We give two approximation algorithms for the problem: a greedy $(2 - \frac{2}{k})$-approximation based on Gomory-Hu trees, and a $(2 - \frac{2}{|X|})$-approximation based on LP rounding. We use the insight from the rounding to develop an exact bi-directed formulation for the global minimum cut problem (the k-cut problem with $k = 2$).

Keywords: Multiway cut, k-cut, Steiner tree, minimum cut, approximation algorithm.

1 Introduction

The k-cut problem and the multiway cut problem are fundamental graph partitioning problems. In both problems we are given an undirected edge-weighted graph $G = (V, E)$ with $w(e)$ denoting the weight of edge $e \in E$. In the k-cut problem the goal is to find a minimum weight set of edges whose removal separates the graph into k disconnected components. In the multiway cut problem we are given a set of k terminals, $X \subseteq V$, and the goal is to find a minimum weight set of edges whose removal separates the graph into components such that each terminal is in a different connected component. In this paper we consider a generalization of the two problems, namely, the Steiner k-cut problem. In this problem, we are given an undirected weighted graph G, a set of *terminals* $X \subseteq V$, and an integer $k \leq |X|$. The goal is to find a minimum weight set of edges whose removal separates the graph into k components with vertex sets $V_1, V_2, \ldots, V_k$, such that $V_i \cap X \neq \emptyset$ for $1 \leq i \leq k$. If $X = V$, we obtain the k-cut problem. If $|X| = k$ we obtain the multiway cut problem.

The k-cut problem can be solved in polynomial time for fixed k [5, 6], but is NP-complete when k is part of the input [5]. In contrast, the multiway cut problem is NP-complete for all $k \geq 3$ and is also APX-hard for all $k \geq 3$ [2]. It follows that the Steiner k-cut problem is NP-complete and APX-hard for all $k \geq 3$. For the multiway cut problem Calinescu, Karloff and Rabani [1] gave a $1.5 - 1/k$ approximation using an interesting geometric relaxation. Karger et al. [7] improved the

*A preliminary version of this paper appeared in *Proc. of ICALP*, 2003.

†Bell Labs, 600 Mountain Ave, Murray Hill, NJ 07974. chekuri@research.bell-labs.com

‡Dept. of Computer Information Sciences, Univ. of Pennsylvania, Philadelphia, PA 19104. sudipto@cis.upenn.edu. This research was supported in part by an Alfred P. Sloan Research Fellowship and by an NSF Award CCF-0430376.

§Computer Science Dept., Technion, Haifa 32000, Israel. naor@cs.technion.ac.il

analysis of the integrality gap of this relaxation and obtained an approximation ratio of $1.3438 - \epsilon_k$, where $\epsilon_k \to 0$ as $k \to \infty$. For the k-cut problem Saran and Vazirani [11] gave a $2 - \frac{2}{k}$ approximation algorithm using a greedy algorithm. This result was improved by [13] to $2 - \frac{3}{k}$ for odd k and to $2 - \frac{3k-4}{k^2-k}$ for even k. Recently, two different 2-approximations for the k-cut problem were obtained. The algorithm of Naor and Rabani [9] is based on rounding a linear programming formulation of the problem and the algorithm of Ravi and Sinha [10] is based on the notion of network strength and Lagrangean relaxation.

The authors have learnt of related independent work of Maeda, Nagamochi and Ibaraki [8] (in Japanese) and Zhao, Nagamochi, and Ibaraki [14]. The Steiner k-cut was considered in [8] where it is shown that a greedy algorithm similar to the one we describe in this paper has an approximation ratio of $2 - 2/k$. In [14], the authors define a generalization of the Steiner k-cut problem which they refer to as the multiway partition problem (MPP). MPP is defined as follows. We are given a finite set V, a set of terminals $X \subseteq V$ and an integer k such that $|X| \geq k$. We are also given a submodular function f on V that assigns a real value $f(S)$ to each subset $S \subseteq V$. The function f is provided as an oracle. The goal is to partition V into k sets $V_1, V_2, \ldots, V_k$ such that $V_i \cap X \neq \emptyset$ for $1 \leq i \leq k$ and minimize $f(V_1) + f(V_2) + \ldots + f(V_k)$. It is shown in [14] that the greedy algorithm that iteratively increases the size of the partition yields a $(2 - \frac{2}{k})$-approximation for MPP. The Steiner k-cut problem can be seen to be a special case of MPP: given an edge weighted graph $G = (V, E)$, we can define a submodular function f where $f(S) = \frac{1}{2} \sum_{e \in \delta_G(S)} w_e$.

1.1 Results

We provide two approximation algorithms for the Steiner k-cut problem. The first algorithm we present is combinatorial and has an approximation ratio of $(2 - \frac{2}{k})$. This algorithm is based on choosing cuts from the Gomory-Hu tree of the given graph and is similar to approximation algorithms developed for the k-cut problem and the multiway cut problem [12]. Maeda et al. [8] obtained the same result earlier but our proof is considerably simpler. Also, as we mentioned earlier, Zhao, Nagamochi, and Ibaraki [14] show that the greedy algorithm yields a $(2 - \frac{2}{k})$ approximation for MPP. Our main result is a 2-approximation algorithm for the Steiner k-cut problem which is based on rounding a linear programming formulation. Although our formulation is a straight forward generalization of the formulation in [9] (for the k-cut problem), our rounding scheme differs substantially. The rounding in [9] exploits the properties of optimal solutions to the LP relaxation. These properties do not hold for the relaxation of the Steiner k-cut problem. Instead we rely on the primal dual algorithm and analysis of Goemans and Williamson [4] for the Steiner tree problem. As a consequence, our rounding algorithm extends to any feasible solution of the linear programming formulation. This interesting new connection might have future applications.

We conclude with a bi-directed formulation for the global minimum cut problem and prove that the linear relaxation of this formulation is exact. The formulation and analysis are inspired by our analysis for the Steiner k-cut problem. This formulation and its integrality gap may have been known previously, however, we could not find a published reference and hence include it here.

2 Combinatorial $(2 - \frac{2}{k})$-approximation algorithm

We assume without loss of generality that the given graph G is connected. A natural greedy algorithm for the Steiner k-cut problem is the following iterative algorithm. In each iteration, find a minimum weight cut that increases the number of distinct components that contain a terminal. This algorithm has been shown to achieve a $(2 - \frac{2}{k})$ approximation algorithm for both the k-cut

problem and the multiway cut problem (see for e.g., [12]) and for MPP [14]. However, the analysis of this algorithm is non-trivial. As in [11, 12], we consider an alternative algorithm that is based on the Gomory-Hu tree representation of the minimum cuts in a graph. Recall that a Gomory-Hu tree for an edge-weighted undirected graph $G = (V, E)$ is an edge-weighted tree $T = (V, E_T)$ with weight function c that has the following property: for all $u, v \in V$, the weight of a minimum cut separating u and v in G is equal to the smallest edge weight on the unique path between u and v in T. In particular, for $(u, v) \in E_T$, $c(u, v)$ is the weight of the minimum cut separating u and v in G and the partition of V induced by the removal of (u, v) from T induces such a minimum cut. We run the natural greedy algorithm mentioned above on the tree T:

> Iteratively, pick the smallest weight edge in T separating a pair of terminals that are not already separated, until k components, each of which contains a terminal, are generated.

It is easy to see that we pick $k - 1$ edges in T. We take the union of the cuts associated with these edges and this is our solution for the Steiner k-cut problem in G.

Proposition 2.1 *The algorithm produces a feasible solution to the Steiner k-cut problem.*

We need a simple proposition about Gomory-Hu trees.

Proposition 2.2 *Let $T = (V, E_T)$ be a Gomory-Hu tree for a connected graph $G = (V, E)$. For any pair of vertices (s, t) in G and an $s - t$ cut $(S, V - S)$ in G, there is an edge $(u, v) \in E_T$ such that $u \in S$, $v \in V - S$, and (u, v) lies on the path between s and t in T.*

Now we argue about the cost of the solution. Our analysis is similar to that of the analysis for the Gomory-Hu tree based algorithm for the k-cut problem (see Theorem 4.8 in [12, Page 42]). However, the analysis is not a straight forward extension - the terminals in the Steiner k-cut problem constrain the choice of cuts and we need to do identify a mapping to the optimal set of cuts in a careful manner.

Lemma 2.3 *The cost of the $(k - 1)$ edges picked by the algorithm is at most $(2 - 2/k)$ times the cost of the optimal solution.*

Proof: Fix an optimal solution A to the Steiner k-cut problem and let $V_1, V_2, \ldots, V_k$ be the partitioning of V defined by A. Clearly, each set V_i $(i = 1, \ldots, k)$ contains at least one terminal from X. From each set V_i we arbitrarily choose a terminal t_i contained in V_i. Define cuts $A_i = (V_i, V \setminus V_i)$ for $i = 1, \ldots, k$, and let $w(A_i)$ denote the weight of cut A_i. Assume without loss of generality that $w(A_1) \leq w(A_2) \leq \cdots \leq w(A_k)$. Each edge that is cut in the optimum solution A participates in exactly two of the cuts $A_1, \ldots, A_k$, hence the weight of the optimal solution A is $w(A) = \sum_{i=1}^{k} w(A_i)/2$. Let $B_1, B_2, \ldots, B_{k-1}$ denote the $k-1$ cuts chosen by the above Gomory-Hu tree based algorithm. We claim that

$$w(B_i) \leq w(A_i), \qquad 1 \leq i \leq k - 1. \tag{1}$$

Assuming the claim, we have that

$$\sum_{i=1}^{k-1} w(B_i) \leq \left(1 - \frac{1}{k}\right) \sum_{i=1}^{k} w(A_i) \leq 2\left(1 - \frac{1}{k}\right) w(A),$$

which proves the desired bound on the performance of the algorithm.

To prove (1), we identify a set of edges $e_1, e_2, \ldots, e_{k-1}$ of the Gomory-Hu tree T with the following properties.

1. $w(A_i) \geq c(e_i)$, for $1 \leq i \leq k-1$. Since $w(A_1) \leq w(A_2) \leq \ldots \leq w(A_k)$, it follows that $w(A_i) \geq \max_{1 \leq j \leq i} c(e_i)$.

2. The removal of $e_1, e_2, \ldots, e_i$ creates $i+1$ components in T, each containing a terminal.

Assuming the existence of $e_1, e_2, \ldots, e_{k-1}$ as above, let $f_1, f_2, \ldots, f_{k-1}$ be the edges of T picked by the algorithm. We claim that $c(f_i) \leq \max_{1 \leq j \leq i} c(e_i)$; this follows by observing that there is some edge in $\{e_1, e_2, \ldots, e_i\}$ that when added to $\{f_1, \ldots, f_{i-1}\}$ would yield a new component containing a terminal. If not, removing the edges in $\{f_1, f_2, \ldots, f_{i-1}\} \cup \{e_1, \ldots, e_i\}$ would result in at most i components each containing a terminal which contradicts the definition of the e_i. Therefore,

$$w(B_i) = c(f_i) \leq \max_{1 \leq j \leq i} c(e_i) \leq w(A_i).$$

We obtain $e_1, \ldots, e_{k-1}$ as follows. Let $E' \subseteq E_T$ be the set of edges of T that cross the partition of V induced by the optimum solution $V_1, V_2, \ldots, V_k$. In other words, $(u,v) \in E'$ if and only if $(u,v) \in E_T$, $u \in V_i$, $v \in V_j$, and $i \neq j$. Root the tree at t_k. For each t_i, $1 \leq i \leq k-1$, we let e_i to be first edge in the directed path from t_i to the root t_k that is in E'. By Proposition 2.2, e_i exists. Also, for $i \neq j$, e_i and e_j are distinct; otherwise, the path between t_i and t_j in T would not have any edges in E' and this contradicts Proposition 2.2. Since e_i crosses the partition V_i, from the Gomory-Hu tree property, $w(A_i) \geq c(e_i)$. We claim that removing $e_1, e_2, \ldots, e_i$ from T will disconnect the set $\{t_1, t_2, \ldots, t_i, t_k\}$ in T. Suppose that this is not the case. Clearly, t_k is separated from $t_1, \ldots, t_i$, therefore for some $h, \ell \leq i$, t_h and t_ℓ are connected by a path P after removing $e_1, \ldots, e_i$. Let v be the least common ancestor of t_h and t_ℓ in T rooted at t_k. From our assumption e_h and e_ℓ are both above v. This implies that no edge in P is in E', and therefore P connects t_h and t_ℓ even after $e_1, \ldots, e_{k-1}$ are removed, contradicting Proposition 2.2. ■

Given a Gomory-Hu tree for the input graph, the iterative greedy algorithm that we described can be easily implemented in $O(n^2)$ time. This could potentially be improved but we do not attempt it since the running time to build a Gomory-Hu tree is currently $\Omega(n^2)$ even for sparse graphs. We conclude with the following theorem.

Theorem 2.4 *There is a $(2 - \frac{2}{k})$-approximation algorithm for the Steiner k-cut problem that runs in $O(n^2 + T)$ time where T is the time required to build a Gomory-Hu tree for the input graph.*

3 Linear Programming Formulation and a 2-approximation

We consider the following integer programming formulation for the Steiner k-cut problem. For each edge e we have a binary variable $d(e)$ which is 1 if the edge e belongs to the cut and 0 otherwise. Let T be a Steiner tree on the terminal set X in G. In any feasible Steiner k-cut, at least $k-1$ edges of T have to be cut. Based on this we obtain the following integer program for the Steiner k-cut problem.

$$\begin{aligned}
(K) \qquad \min \sum_{e \in E} w(e) \cdot d(e) \quad & \textit{subject to}: \\
\sum_{e \in T} d(e) \;&\geq\; k-1 \qquad\qquad \forall\, T : T \text{ Steiner tree on } X \\
d(e) \;&\in\; \{0,1\} \qquad\qquad \forall\, e \in E
\end{aligned}$$

A relaxation of this integer program is obtained by allowing the variables $d(e)$ to assume values in $[0,1]$. The variables $d(e)$ are to be interpreted as inducing a semi-metric[1] on V. Our formulation above is a straight forward extension of the formulation of Naor and Rabani [9] for the k-cut problem. In the k-cut problem $X = V$, and hence [9] considers only spanning trees of G.

Unfortunately, we do not know how to solve the above linear program in polynomial time. Consider, for example, the separation oracle required for running the Ellipsoid algorithm. Given edge weights $d(e)$, the separation oracle has to check that the minimum weight Steiner tree on X in G is of weight at least $k-1$. However, this problem is NP-hard. Note that for the k-cut problem, a polynomial time separation oracle is available because the minimum spanning tree (MST) of a graph can be computed in polynomial time.

We can use an approximate separation oracle based on the MST heuristic for the Steiner tree problem. Given edge weights $d(e)$, $e \in E$, we define the metric completion. For an unordered pair or vertices uv we let $d(uv)$ denote the shortest path distance from u to v in G with edge weights defined by d. Let G_X be the complete graph on the terminal set X. The oracle computes the MST on G_X where for each pair uv in G_X the weight of the edge uv is $d(uv)$. If the MST is of weight at least $k-1$, the oracle concludes that d is feasible. If the weight of the MST is less than $k-1$, it is easy to find a corresponding Steiner tree on X in G whose weight is less than $k-1$. In other words, we are solving the following relaxation:

$$(K') \qquad \min \sum_{uv \in E(G)} w(uv) \cdot d(uv) \quad \textit{subject to}:$$

$$\sum_{uv \in E(T)} d(uv) \geq k-1 \qquad T \text{ spanning tree in } G_X \tag{2}$$

$$d(uv) + d(vw) \geq d(uw) \qquad u,v,w \in V \tag{3}$$

$$d(uv) \in [0,1] \qquad u,v \in V \tag{4}$$

For an edge $e \in E(G)$ with $e = uv$, we use $d(e)$ and $d(uv)$ interchangeably. The next lemma follows from the discussion.

Lemma 3.1 *The linear program K' is a valid relaxation for the Steiner k-cut problem and it can be solved optimally in polynomial time.*

For the multiway cut problem we note that the linear program K' is equivalent to a linear program that constrains the terminals to be at distance at least 1 from each other. This latter linear program has been shown to have an integrality gap of $2(1-1/k)$ [2]. We will obtain the same result as well for the Steiner k-cut problem. We now prove a property of feasible solutions to K' that will be useful later.

Lemma 3.2 *In any feasible solution to K' there is $X' \subseteq X$ such that $|X'| \geq k$, and for any two distinct vertices u and v in X', $d(uv) > 0$.*

Proof: For any two, not necessarily distinct, vertices u and v in X, define a relation R as follows: uRv if and only if $d(uv) = 0$. Since d is symmetric and satisfies triangle inequality (hence the relation is transitive), R defines an equivalence relation on X. We need to prove that the number of equivalence classes in R is at least k. Suppose this is not the case. For any two vertices a and b

[1] A semi-metric is a distance function that is symmetric and satisfies the triangle inequality. It differs from a metric in that it need not satisfy reflexivity, that is, non-identical objects can be at distance 0 from each other.

in V, $d_{ab} \leq 1$. Hence, there is a spanning tree on X of cost at most $\ell - 1$, where ℓ is the number of distinct equivalence classes. If $\ell < k$, we get a contradiction to the the feasibility of the solution to K'. ■

Note that the above proof is constructive and a set X' satisfying the required properties can be easily computed. In the rest of the paper it is convenient to assume that $X' = X$ and that for each $u, v \in X$, $d(uv) > 0$.

3.1 A strategy to round the Linear Program

We show how to round a solution to (K') to yield a 2-approximation to the Steiner k-cut problem. To this end, we use the Goemans and Williamson primal-dual approximation algorithm for the Steiner tree problem [4] (henceforth referred to as the GW algorithm) to find a family of cuts.

Let $\bar{d}$ be any feasible solution to linear program (K'). Then, $\bar{d}$ defines a weight function on the edges of G. Let $G_{\bar{d}}$ denote the resulting edge weighted graph. We run the GW primal-dual algorithm on the graph $G_{\bar{d}}$ to create a Steiner tree on X. To find a minimum Steiner tree on X in $G_{\bar{d}}$, the GW algorithm uses the following cut based LP relaxation of the Steiner tree problem. Let $x(e)$ be 1 if e is in the Steiner tree and 0 otherwise: every cut that separates the terminal set has to be covered by at least one edge. This yields the following linear program where the variables are relaxed to be in $[0, 1]$. Note that the variables $\bar{d}(e)$ in the formulations below are treated as constants obtained from a solution to (K').

Every subset of vertices S naturally defines a cut which we denote by $\delta(S)$.

$$(STP) \qquad \min \sum_e \bar{d}(e) \cdot x(e) \quad \textit{subject to}:$$

$$\sum_{e \in \delta(S)} x(e) \geq 1 \qquad \forall\, S : S \text{ separates } X \tag{5}$$

$$x(e) \in [0, 1] \qquad \forall\, e \tag{6}$$

The dual of this linear program is the following.

$$(STD) \qquad \max \sum_S y(S) \quad \textit{subject to}:$$

$$\sum_{S : e \in \delta(S)} y(S) \leq \bar{d}(e) \qquad \forall \text{ e} \tag{7}$$

$$y(S) \geq 0 \qquad \forall\, S : S \text{ separates } X \tag{8}$$

The GW algorithm is a primal-dual algorithm that incrementally grows a dual solution while maintaining feasibility and computes a corresponding feasible primal Steiner tree such that the cost of the Steiner tree computed is at most twice the value of the dual solution found. Let y' be the dual solution produced by the GW algorithm upon termination and let T be the tree returned by the algorithm. Then the following properties hold for y' and T [4].

1. y' is a feasible solution to (STD).
2. T is a tree that spans the terminal set X.

3. Sets S (representing cuts) with $y'(S) > 0$ form a laminar family. Let $\mathcal{S}$ denote this family of sets.

4. $\sum_{e \in T} \bar{d}(e) \leq 2(1 - 1/|X|) \sum_{S \in \mathcal{S}} y'(S)$.

5. For any $u \in X$, $\sum_{S:u \in S} y'(S) \leq \frac{1}{2} \cdot \max_{v \in X, v \neq u} \bar{d}(uv)$. In particular, $\sum_{S:u \in S} y'(S) \leq \frac{1}{2}$.

6. For any $u, v \in X$ such that $\bar{d}(uv) > 0$, there exists a cut S such that $y'(S) > 0$ and $|S \cap \{u, v\}| = 1$.

With the above discussion in place, we are ready to describe our rounding procedure. For a cut S, let $w(S) = \sum_{e \in \delta(S)} w(e)$ denote the weight of S in G. We observe the following:

Claim 3.3 $\sum_{S \in \mathcal{S}} y'(S) w(S) \leq \sum_e w(e) \bar{d}(e)$.

Proof: We have the following:

$$\begin{aligned} \sum_{S \in \mathcal{S}} y'(S) w(S) &= \sum_{S \in \mathcal{S}} y'(S) \sum_{e \in \delta(S)} w(e) \\ &= \sum_e w(e) \sum_{S: e \in \delta(S)} y'(S) \\ &\leq \sum_e w(e) \bar{d}(e). \end{aligned}$$

The final inequality follows from Constraint (7) since y' is a feasible solution to (STD). ■

Claim 3.4 $2(1 - 1/|X|) \sum_{S \in \mathcal{S}} y'(S) \geq (k - 1)$.

Proof: The GW algorithm guarantees that $2(1 - 1/|X|) \sum_S y'(S) \geq \sum_{e \in T} \bar{d}(e)$. Since T is a spanning tree on X, from the feasibility of $\bar{d}$ for (K'), $\sum_{e \in T} \bar{d}(e) \geq k - 1$ by Equation (2). The claim now follows. ■

3.2 Choosing the Cuts

We describe how we choose the cuts from $\mathcal{S}$. We partition $\mathcal{S}$ into classes $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_\ell$ such that two cuts S and S' are in the same class $\mathcal{S}_i$ if and only if $S \cap X = S' \cap X$. Clearly, the number of classes is at least $|X| \geq k$. For a class $\mathcal{S}_i$, let C_i be a least weight cut in $\mathcal{S}_i$. Let $\mathcal{C}$ be the collection of C_i, $1 \leq i \leq \ell$. Without loss of generality assume that the classes are ordered such that $w(C_1) \leq w(C_2) \leq \ldots \leq w(C_\ell)$.

The algorithm considers classes in increasing order of their index and while considering class $\mathcal{S}_i$, adds C_i to the solution if adding the cut produces a new component containing a terminal from X. The process stops when $k - 1$ cuts are chosen. This procedure is well-defined and yields a feasible for the following reason. From Lemma 3.2 and Property 6 of the GW algorithm, if all the cuts $C_1, C_2, \ldots, C_\ell$ are chosen, we obtain k (or more) components each containing a terminal from X. We argue about the quality of the solution. Let $1 = i_1 < i_2 < \ldots < i_{k-1} < \ell$ denote the indices of the $k - 1$ classes chosen by the algorithm. We let $y'(\mathcal{S}_i)$ denote $\sum_{S \in \mathcal{S}_i} y'(S)$).

Definition 3.5 *Given a collection of distinct cuts $\mathcal{B}$, we say that a cut $C \in \mathcal{B}$ is* basic *with respect to $\mathcal{B}$ if there is no cut $C' \in \mathcal{B}$ such that $C' \subsetneq C$.*

From laminarity of $\mathcal{S}$ and hence of $\mathcal{B}$, the set of basic cuts in $\mathcal{B}$ is well defined and they are disjoint. Let $\mathcal{A}_j$ denote the set of cuts $C_1, C_2, \ldots, C_j$.

Lemma 3.6 *Let q_j be the number of basic cuts in $\mathcal{A}_j$ and let p_j be the number of components created by the algorithm after the first j cuts have been considered. Then*

- $\sum_{1 \leq h \leq j} y'(\mathcal{S}_h) \leq q_j/2$.
- *$p_j \geq q_j$ and if $p_j = q_j$ then the components are induced by the basic cuts in $\mathcal{A}_j$ and $X \subset \cup_{h=1}^{j} C_j$.*

Proof: From the analysis of the GW algorithm we have that for any cut S, $\sum_{S' \supseteq S} y'(S') \leq \Delta/2$ where Δ is the diameter of G. In our case $\Delta = 1$. Since every cut in $\mathcal{A}_j$ is a superset of some basic cut in $\mathcal{A}_j$, we have that $\sum_{1 \leq h \leq j} y'(\mathcal{S}_h) \leq q_j/2$.

Let $r_1 < r_2 < \ldots < r_{q_j}$ be the indices of the basic cuts in $\mathcal{A}_j$. Note that the cuts in $\mathcal{S}$ are laminar and hence these basic cuts are disjoint. We now argue that $p_j \geq q_j$. Let $X_h = X \cap C_{r_h}$, $1 \leq h \leq q_j$, and let $X' = X - \uplus_{h=1}^{q_j} X_h$. We claim that for $h < h'$, X_h and $X_{h'}$ are in separate components; otherwise, the algorithm when processing C_{r_h} would add it to the solution and separate X_h and $X_{h'}$. Therefore $p_j \geq q_j$. By the same argument, it follows that if X' is not empty, X_h and X' are in separate components as well and in this case $p_j \geq q_j + 1$. Thus, if $p_j = q_j$, $X' = \emptyset$ and each X_h is in a separate component. ■

Let $\alpha = 1/(1 - 1/|X|)$. From the analysis of the GW algorithm we have that $\sum_{h=1}^{\ell} y'(\mathcal{S}_h) = \sum_S y'(S) \geq \alpha(k-1)/2$. The main tool in our analysis is the following lemma.

Lemma 3.7 *For $1 \leq r \leq k-1$, $\sum_{j \geq i_r} y'(\mathcal{S}_j) \geq \alpha(k-r)/2$.*

Proof: Let $f = i_r - 1$, then $p_f = r$. We consider two cases based on q_f.

If $p_f > q_f$, we have that $q_f \leq r - 1$ and by Lemma 3.6, $\sum_{1 \leq h \leq f} y'(\mathcal{S}_h) \leq (r-1)/2$. Since $\sum_{1 \leq h \leq \ell} y'(\mathcal{S}_h) \geq \alpha(k-1)/2$ it follows that $\sum_{i_r \leq j \leq \ell} y'(\mathcal{S}_j) \geq \alpha(k-r)/2$.

Now we consider the case $p_f = q_f$. From Lemma 3.6, the components at this stage are induced by the basic cuts in $\mathcal{A}_f$. Let the basic cuts be $C_{j_1}, C_{j_2}, \ldots, C_{j_r}$. Let X_h denote the terminals in C_{j_h}. Recall that $X = \uplus_h X_h$ and hence $\sum_{1 \leq h \leq r} |X_h| = |X|$. The tree T created by the GW algorithm is of cost $k-1$. We note that the part of the tree that connects the components $C_{j_1}, C_{j_2}, \ldots, C_{j_r}$ costs at most $r-1$ since the diameter of the graph is at most 1. For $1 \leq h \leq r$, let T_h be the minimal subtree of T that connects X_h. It follows that $\sum_{1 \leq h \leq r} \sum_{e \in T_h} \bar{d}_e \geq k - 1 - (r-1) \geq k - r$. Let $L_h = \{i \mid (C_i \cap X) \subsetneq X_h\}$ be the indices of classes that contain a proper subset of terminals from X_h. From the analysis of the GW algorithm applied to tree T_h and terminals set X_h, we obtain that

$$\sum_{i \in L_h} y'(\mathcal{S}_i) \geq \frac{1}{2(1 - 1/|X_h|)} \sum_{e \in T_h} \bar{d}_e.$$

Therefore

$$\sum_{1 \leq h \leq r} \sum_{i \in L_h} y'(\mathcal{S}_i) \;\geq\; \sum_{1 \leq h \leq r} \frac{1}{2(1 - 1/|X_h|)} \sum_{e \in T_h} \bar{d}_e \;\geq\; \frac{1}{2(1 - 1/|X|)}(k - r).$$

We now claim that if $i \in \uplus_h L_h$ then $i > f = i_r - 1$. For if $i \in L_h$ then C_{j_h} would not be basic in $C_1, C_2, \ldots, C_f$. Therefore

$$\sum_{j \geq i_r} y'(\mathcal{S}_h) \;\geq\; \sum_{1 \leq h \leq r} \sum_{i \in L_h} y'(\mathcal{S}_i).$$

The lemma follows. ■

Corollary 3.8 $\sum_{r=1}^{k-1} w(C_{i_r}) \leq 2(1 - 1/|X|) \leq \sum_S y'(S)w(S)$.

Proof: For $1 \leq h \leq \ell$ let $z_h = \sum_{j \geq h} y'(\mathcal{S}_j)$. Recall that $1 = i_1 < i_2 < \ldots < i_{k-1} < \ell$ are the indices of the cuts chosen by the algorithm and that $w(C_1) \leq w(C_2) \leq \ldots \leq w(C_\ell)$. Hence,

$$\begin{aligned}
\sum_S y'(S)w(S) &= \sum_{h=1}^{\ell} \sum_{S \in \mathcal{S}_h} y'(S)w(S) \\
&\geq \sum_{h=1}^{\ell} y'(\mathcal{S}_h)w(C_h) \\
&\geq w(C_{i_{k-1}})z_{i_{k-1}} + \sum_{r=1}^{k-2} w(C_{i_r})(z_{i_r} - z_{i_{r+1}}).
\end{aligned}$$

From Lemma 3.7 we have that $z_{i_r} \geq \alpha(k-r)/2$. The right hand side of the last inequality above is minimized when $z_{i_r} = \alpha(k-r)/2$ for $1 \leq r \leq k-1$. Therefore,

$$\sum_S y'(S)w(S) \;\geq\; \frac{1}{2}\alpha \sum_{r=1}^{k-1} w(C_{i_r}).$$

This yields the desired inequality. ■

From Corollary 3.8 and Claim 3.3 we obtain that

$$\sum_{r=1}^{k-1} w(C_{i_r}) \;\leq\; 2(1 - 1/|X|) \sum_S y'(S)w(S) \;\leq\; 2(1 - 1/|X|) \sum_e w_e \bar{d}_e.$$

Thus the integrality gap of (K') is upper bounded by $2(1 - 1/|X|)$.

Lower bound on the integrality gap: The integrality gap of (K') (and (K)) is no better than $2(1-1/|X|)$ even when $k = 2$ and $X = V$, i.e., the global minimum cut problem. Consider the unit weight cycle on n vertices. Clearly, an integral solution has to cut at least two edges to separate the cycle into two components. Consider the following feasible solution to the relaxation. We set $d(e) = 1/(n-1)$ on each edge of the cycle; for all other edges, $d(e)$ is the shortest path distance induced by the distances on the cycle edges. The value of this solution is $n/(n-1)$. Hence, the integrality gap is $2(1 - 1/n)$.

Theorem 3.9 *The integrality gap of the LP (K') is $2(1 - 1/|X|)$.*

4 An exact formulation for the global minimum cut problem

In the previous section we saw that linear program (K') has an integrality gap of $2(1-1/n)$ for the 2-cut problem, i.e., for the global minimum cut problem. Here we give a bi-directed formulation of the global minimum cut problem. Given an undirected weighted graph $G = (V, E)$ let $G^b = (V, A)$ be the directed graph obtained by replacing each edge $e \in E$ between u and v by two directed arcs (u, v) and (v, u). The weights of both (u, v) and (v, u) in G^b are set to $w(e)$. Let r be any vertex in $V(G)$. An *arborescence* in a directed graph rooted at a vertex r is a spanning out-tree from r (also known as a *branching*). Our formulation is based on G^b. For an arc $a \in A$, let $d(a) = 1$ if a is chosen to the cut, and let $d(a) = 0$ otherwise. The following is a valid integer program for the global minimum cut problem. The root r is chosen arbitrarily.

$$
\begin{aligned}
(B) \qquad \min \sum_{a \in A} w(a) \cdot d(a) \quad & \textit{subject to}: \\
\sum_{a \in T} d(a) \;&\geq\; 1 && \forall\, T : T \text{ arborescence rooted at } r \text{ in } G^b \\
d(a) \;&\in\; \{0,1\} && \forall\, a \in A
\end{aligned}
$$

Although the above integer program is similar to integer program (K), we remark that, for $k > 2$, we do not obtain a valid formulation for the k-cut problem if we replace the right hand side of the constraint above by $k - 1$.

We obtain a linear program by relaxing each variable $d(a)$ to be in $[0, 1]$. We show that the value of the linear program is exactly equal to the global minimum cut of the graph G. The separation oracle needed to solve (B) in polynomial time by the Ellipsoid algorithm is the minimum cost arborescence problem in directed graphs. We can use the algorithm of Edmonds [3] for this purpose. In fact, Edmonds [3] showed that the arborescence polytope is integral and we use this to show that (B) is exact for the minimum cut problem. The proof is similar in outline to the one in Section 3, however, we use arborescences in place of spanning trees, and the result of Edmonds [3] on the integrality of the arborescence polytope in place of the GW algorithm. Let $\bar{d}$ be an optimal solution to (B). Let $G^b_{\bar{d}}$ be the graph G^b equipped with $\bar{d}$ as costs on the edges of G^b. We find a minimum cost arborescence in $G^b_{\bar{d}}$ using the following formulation. For each arc a, variable $x(a) = 1$ if a belongs to the arborescence and 0 otherwise.

$$
\begin{aligned}
(AP) \qquad \min \sum_{a \in A} d(a) \cdot x(a) \quad & \textit{subject to}: \\
\sum_{a \in \delta(S)} x(a) \;&\geq\; 1 && \forall\, S : S \neq V \text{ and } r \in S \\
x(a) \;&\in\; [0,1] && \forall\, a
\end{aligned}
$$

The dual of the above linear program is the following.

$$
\begin{aligned}
(AD) \qquad \max \sum_{S} y(S) \quad & \textit{subject to}: \\
\sum_{S : a \in \delta(S)} y(S) \;&\leq\; d(a) && \forall \text{ a} \\
y(S) \;&\geq\; 0 && \forall\, S : S \neq V \text{ and } r \in S
\end{aligned}
$$

Let $\bar{x}^*$ and $\bar{y}^*$ be optimal primal and dual solutions to (AP) and (AD) on the graph G_d^b. From the feasibility of $\bar{d}$, it follows that $\sum_a d(a)x^*(a) \geq 1$. From weak duality we therefore also obtain that $\sum_S y^*(S) \geq 1$. Let $\mathcal{S} = \{S \mid y^*(S) > 0\}$ be the set of all cuts with strictly positive dual values. Let $C \in \mathcal{S}$ be a cut such that $w(S)$ is the cheapest cut. We pick C as our solution. We now show that $w(C) \leq \sum_a w(a)d(a)$ which shows that the weight of the cut is at most the value of the optimal solution to (B). We see that

$$\begin{aligned}
\sum_S y^*(S)w(S) &= \sum_S y^*(S) \sum_{a \in \delta(S)} w(a) \\
&= \sum_a w(a) \sum_{S: a \in \delta(S)} y^*(S) \\
&\leq \sum_a w(a)d(a).
\end{aligned}$$

The last inequality follows from the feasibility of y^*. We have that $\sum_S y^*(S)w(S) \leq \sum_a w(a)d(a)$ and $\sum_S y^*(S) \geq 1$. Therefore, the weight of the cheapest cut is no more than $\sum_a w(a)d(a)$.

Theorem 4.1 *The LP relaxation of (B) can be solved in polynomial time and is an exact formulation for the global minimum cut problem.*

5 Conclusions

Our study of linear programming relaxations for the Steiner k-cut problem was partly motivated by the goal of obtaining an approximation algorithm for the k-cut problem with a ratio better than 2. This has been accomplished for the multiway cut problem by a strengthened LP relaxation [1]. Our results show that the available approximation techniques for the k-cut problem extend to the Steiner k-cut problem. In the process we have shown an interesting connection between laminar cut families obtained from the primal-dual algorithm of Goemans and Williamson [4] and their use in analyzing the LP relaxation for the Steiner k-cut problem. We hope that our ideas will be useful in developing and analyzing stronger LP relaxations that have integrality gap strictly smaller than 2 for the k-cut problem and the Steiner k-cut problem. Several important questions are open.

- Is there an approximation algorithm for the k-cut problem with a ratio better than 2?
- Is the k-cut problem APX-hard?
- What is the integrality gap of the geometric relaxation in [1] for the multiway cut problem?

Acknowledgments: We thank David Shmoys and Zoya Svitkina for pointing out an erroneous proof in a previous version of the paper. We thank two anonymous referees for their comments which helped improve the clarity of the proofs in Section 3.2.

References

[1] G. Călinescu, H. Karloff, and Y. Rabani. An improved approximation algorithm for multiway cut. *Journal of Computer and System Sciences*, 60:564–574, 2000.

[2] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM J. on Computing*, 23:864–894, 1994.

[3] J. Edmonds. Optimum branchings. *J. Res. Nat. Bur. Standards*, B71:233-240, 1967.

[4] M. Goemans and D. Williamson. A general approximation technique for constrained forest problems. *SIAM J. on Computing*, 24:296–317, 1995.

[5] O. Goldschmidt and D. Hochbaum. Polynomial algorithm for the k-cut problem. *Math. of OR*, 19:24–37, 1994.

[6] D. Karger and C. Stein. A new approach to the minimum cut problem. *JACM*, 43:601–640, 1996.

[7] D. Karger, P. Klein, C. Stein, M. Thorup, and N. Young. Rounding algorithms for a geometric embedding of minimum multiway cut. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, 668–678, 1999.

[8] N. Maeda, H. Nagamochi, T. Ibaraki. Approximate algorithms for multiway objective point split problems of graphs (in Japanese). Computing devices and algorithms (in Japanese) (Kyoto, 1993). Surikaisekikenkyusho Kokyuroku, 833:98–109, 1993.

[9] J. Naor and Y. Rabani. Approximating k-cuts. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, 26–27, 2001.

[10] R. Ravi and A. Sinha. Approximating k-cuts via Network Strength. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, 621–622, 2002.

[11] H. Saran and V. Vazirani. Finding k-cuts within twice the optimal. *SIAM J. on Computing*, 24:101–108, 1995.

[12] V. Vazirani. **Approximation Algorithms**. Springer, 2001.

[13] L. Zhao, H. Nagamochi, T. Ibaraki. Approximating the minimum k-way cut in a graph via minimum 3-way cuts. *J. Combinatorial Optimization*, 5:397–410, 2001.

[14] L. Zhao, H. Nagamochi, and T. Ibaraki. Greedy splitting algorithms for approximating multiway partition problems. *Math. Programming*, 102:167-183, 2005.