An improved approximation algorithm for requirement cut

Anupam Gupta\textsuperscript{a}, Viswanath Nagarajan\textsuperscript{b,∗}, R. Ravi\textsuperscript{c}

\textsuperscript{a} Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA
\textsuperscript{b} IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA
\textsuperscript{c} Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA 15213, USA

\textbf{A B S T R A C T}

This note presents improved approximation guarantees for the requirement cut problem: given an \(n\)-vertex edge-weighted graph \(G = (V, E)\) and \(g\) groups of vertices \(X_1, \ldots, X_g \subseteq V\) with each group \(X_i\) having a requirement \(r_i\), between 0 and \(|X_i|\), the goal is to find a minimum cost set of edges whose removal separates each group \(X_i\) into at least \(r_i\) disconnected components. We give a tight \(\Theta((\log g) \cdot \max k, n)\) approximation ratio for this problem when the underlying graph is a tree, and show how this implies an \(O((\log k \cdot \log g) \cdot \max k, n)\) approximation ratio for general graphs, where \(k = |\cup_{i=1}^g X_i| \leq n\).

© 2010 Elsevier B.V. All rights reserved.

\section{Introduction}

We study the requirement cut problem \cite{steinemulticut} which is a generalization of several known graph partitioning problems. The input to requirement cut is an \(n\)-vertex undirected graph \(G = (V, E)\) with edge costs \(c : E \to \mathbb{R}_+\), and \(g\) groups of vertices \(X_1, \ldots, X_g \subseteq V\) with each group \(X_i\) having an integer valued requirement \(r_i\) between 0 and \(|X_i|\). The objective is to find a set \(S \subseteq E\) of edges minimizing the cost \(\sum_{e \in S} c_e\), such that each group \(X_i\) (for \(i \in [g]\)) lies in at least \(r_i\) disconnected components of \(G \setminus S\). We denote as \(k := |\cup_{i=1}^g X_i| \leq n\) the number of vertices that participate in at least one group. Requirement cut generalizes several previously studied cut problems such as multicut \cite{multicut}, multitype cut \cite{multitype}, multi-way cut \cite{multicut}, Steiner multicut \cite{steinermulticut}, and \(k\)-cut \cite{kcut}.

In this note, we obtain better approximation ratios for the requirement cut problem. For when the graph \(G\) is a tree, we obtain a tight \(\Theta((\log g) \cdot \max k, n)\) approximation guarantee (Section 2). This improves on the \(O((\log g) \cdot n)\) bound from \cite{steinemulticut}, where \(R = \max_{1 \leq i \leq g} r_i\) is the maximum requirement. For instances of requirement cut on general graphs, we give an \(O((\log k \cdot \log g) \cdot \max n, g)\) approximation algorithm (Section 3) improving on \(O((\log n \cdot \log g) \cdot \max n, g)\) from \cite{steinemulticut}. The Steiner multicut problem \cite{steinermulticut} is the special case of requirement cut when all requirements are 2. Our \(O((\log k \cdot \log g) \cdot \max n, g)\) approximation ratio represents a logarithmic improvement (in some cases) over the previously best-known bounds of \(O((\log^3 g) \cdot \max n, g)\) \cite{steinermulticut} and \(O((\log n \cdot \log g) \cdot \max n, g)\) \cite{steinemulticut} (here \(t := \max_{1 \leq i \leq g} |X_i|\) is the maximum group size; note that \(k \leq \min(n, gt)\)).

All of our algorithms are based on rounding a natural LP relaxation \cite{steinemulticut} for requirement cut. Our improvement in the tree case comes from a better rounding algorithm that first randomly partitions the tree (in a dependent manner) into parts of small diameter, and then applies randomized rounding as in \cite{steinemulticut}. The improvement for general graphs relies on a slightly stronger statement of the FRT \cite{frt} tree embedding (Theorem 7).

We note that in recent work, \cite{exponential} also obtained an approximation guarantee of \(O((\log k \cdot \log g) \cdot \max n, g)\) for requirement cut, that is independent of the graph size \(n\). However the focus in that paper was more general, namely constructing vertex-sparsifiers that approximate all terminal min-cuts, and their requirement cut result follows as an application. On the other hand, in this paper, we deal directly with the requirement cut problem and obtain improved guarantees.

\subsection{Linear programming relaxation}

The following is a natural LP relaxation for requirement cut with a variable \(z_{[u,v]}\) for each unordered pair \(u, v \in V\) of vertices. This is also the linear program that was studied in \cite{steinemulticut}. A similar LP for the Steiner \(k\)-cut problem was studied in \cite{steinerkcut}.

\begin{align*}
\min & \sum_{e \in E} c_e z_e \\
\text{s.t.} & \sum_{e \in E_i} z_e \geq r_i - 1 & \forall T_i : \text{spanning tree in complete graph on } X_i, \forall i = 1, \ldots, g \\
& z_{[u,v]} \leq z_{[u,w]} + z_{[v,w]} & \forall u, v, w \in V \\
& 0 \leq z_{[u,v]} \leq 1 & \forall u, v \in V.
\end{align*}

Note that the second constraint (triangle inequality) ensures that \(z\) defines a metric on vertices \(V\). Also in the first constraint, \(T_i\) ranges
over all spanning trees in the complete graph \((X, \frac{\sum_{i \in X} g_i}{2})\); although some edges of \(T_1\) may be absent in \(E\), their lengths under \(z\) are well-defined. The linear program \(\mathcal{L}\mathcal{P}\) can be solved in polynomial time using the ellipsoid algorithm. Let \(d^*\) denote an optimal solution to \(\mathcal{L}\mathcal{P}\). Our rounding algorithms work with a slightly different length function \(d\), defined as \(d(u, v) := \min\{2, d^*(u, v)\}\) for all \(u, v \in V\). Since \(d^*\) is a metric, so is \(d\). It is also clear that all lengths in both \(d^*\) and \(d\) are in \([0, 1]\). We state a useful claim about metric \(d\), which is Claim 1 from [8].

**Claim 1 (8).** For any group \(X_i (i = 1, \ldots, g)\), the minimum Steiner tree on \(X_i\) under metric \(d\) has length at least \(\tau_i - 1\).

### 2. Requirement cut on trees

In this section, we consider instances of requirement cut when the underlying graph is a tree \(T = (V, E)\) with edge costs \(c : E \to \mathbb{R}_+\). The algorithm is based on rounding the fractional solution \(d\) to \(\mathcal{L}\mathcal{P}\) (see Section 1.1). Applying rounded randomizing on this fractional solution, [8] obtained an \(O(\log gk)\) approximation guarantee. We give a different rounding scheme that achieves a better \(O(\log g)\) bound. Our rounding scheme consists of two stages: first we randomly partition the tree \(T\) into several subtrees of small diameter, and then apply randomized rounding on each subtree.

The first stage of rounding pays an initial logarithmic factor in the cost but ensures that the diameters of the resulting subtrees are at most inverse logarithmic. The second stage now proceeds differently from [8] in that rather than rounding edges according to their LP value in a logarithmic number of rounds to ensure feasibility, we instead boost the rounding probability by a logarithmic factor and perform only one stage of rounding. The small diameter is crucial in allowing the probability boosting in this single-stage rounding step and this is where the first-stage preprocessing is useful. The algorithm is formally described in Fig. 1, and its analysis is given below.

**First-stage rounding.** Note that by triangle inequality, \(d(u, v) \leq \min\{d(u, w), d(w, v)\}\) for all \(u, v \in V\). The first stage (Step 4) cuts edges \(E_1 \subseteq E\); let \(C_1 = \sum_{e \in E_1} c_e\) be its cost. Observe that the probability that an edge \(e \in E\) is cut in Step 4 \(P(e \in E_1) = \min\{\frac{\alpha}{2}, 1\}\). Thus the expected cost of edges cut in this stage \(E[C_1] < \frac{\alpha}{2} \sum_{e \in E} c_e \cdot d_e\).

Note that the above cutting procedure ensures that the diameter (under metric \(d\)) of each subtree is at most \(2\alpha\). For each group \(X_i\) (where \(i \in [g]\)), let \(S_i\) denote the number of distinct subtrees among \(T\) that contain terminals from \(X_i\), and let \(r_i := \max\{r_i - S_i, 0\}\) be the residual requirement of group \(X_i\). By renumbering groups, let \(X_1, \ldots, X_{g'}\) (for \(g' \leq g\)) denote the groups with positive residual requirement. In the second stage of rounding, it suffices to restrict attention to the groups \(X_i\) for \(i = 1, \ldots, g'\).

**Second-stage rounding.** The analysis of this part is conditioned on any instantiation of forest \(T\) that results after the first stage. It is clear that each edge \(e \in E\) has \(d_e \leq \alpha\). The edges picked in the second stage (Step 6) are \(E_2 \subseteq E\); let \(C_2 := \sum_{e \in E_2} c_e\) be its cost. Since each edge \(e \in E\) is included in \(E_2\) with probability at most \(\frac{\alpha}{2}\), the expected cost \(E[C_2] \leq \frac{\alpha}{2} \sum_{e \in E} c_e \cdot d_e\). We will show that \(E_2\) satisfies all residual requirements with high probability. This would imply that \(E_1 \cup E_2\) is a feasible integral solution to the requirement cut instance w.h.p.

Before we prove the main lemma about feasibility of the solution obtained, we introduce a definition. Let \(d' : \left[\frac{1}{2}\right] \to \mathbb{R}_+\) be a new distance function on \(V\) defined as follows:

\[
d'_{(u, v)} := \begin{cases} 1 & \text{if } u \text{ and } v \text{ lie in different subtrees of } T \text{ and } u \neq v \\ \frac{d_{(u, v)}}{2\alpha} & \text{if } u \text{ and } v \text{ lie in the same subtree of } T \\
\end{cases}
\]

Observe that for any vertices \(u, v\) in the same subtree of \(T\), we have \(d'_{(u, v)} \leq 2\alpha\) (this follows from the first-stage rounding). So \(d'\) takes values in \([0, 1]\). It also follows that \(d'\) satisfies the triangle inequality, and hence it is a metric. Note that Step 6 picks each edge \(e \in T\) independently with probability \(d'_{e}\). We now prove the following key property of the second-stage rounding.

**Lemma 2.** For any group \(i \in [g']\), the probability that \(X_i\) lies in fewer than \(r_i\) distinct components of \(T\) \(\cup E_2\) is at most \(\frac{1}{2^{r_i}}\).

**Proof.** We first state some definitions and claims proved in [8] that are useful in this proof. Vertices in \(V \setminus X_i\) are called Steiner, and \(X_i\)-vertices are called terminal. Define \(H_i\) as the forest obtained from \(T\) by repeatedly removing all degree 1 Steiner vertices and short-cutting over all degree 2 Steiner vertices. This ensures that all Steiner vertices in \(H_i\) have degree at least 3. Note that \(H_i\) is not necessarily a subgraph of \(T\); however each edge of \(H_i\) corresponds to a path in \(T\), and the paths for different edges of \(H_i\) are disjoint. We say that an edge \(e \in H_i\) is removed/disconnected iff some edge in the path corresponding to \(e\) is removed in \(T\) (in the second stage). The following claim is Claim 2 proved in [8].

**Claim 3 (8).** The removal of any \(m \geq 1\) edges of forest \(H_i\) results in at least \(\lceil \frac{m+1}{2} \rceil\) more components containing terminals.

In the following, for any subset \(A \subseteq \left[\frac{v}{2}\right]\) of edges and metric \(z\) on \(V, z(A) := \sum_{e \in A} z_e\) is the length of \(A\) under \(z\). For any pair of vertices \(u, v \in V\) that lie in the same subtree of \(T\), let \(p_{u, v}\) denote the probability that vertices \(u\) and \(v\) are disconnected in the second-stage rounding; i.e., \(p_{u, v}\) is the probability that some edge on the \(u-v\) path in forest \(T\) is picked into \(E_2\). Consider a 0–1 random variable \(Z_{e}^{i}\) for each edge \(e = (u, v) \in H_i\) which is 1 if vertices \(u\) and
Claim 4 [8]. \( E(Y_1) = \sum_{e \in H} p_e \geq \left(1 - \frac{1}{2}\right) \cdot d'(H_i) \).

Since \( Y_i \) is a sum of independent 0–1 random variables, a Chernoff bound as in [8] gives

\[
\Pr \left[ \text{increase in } X_i \text{-components less than } \frac{1}{4} E(Y_i) \right] 
\leq \Pr \left[ Y_i < \frac{1}{2} E(Y_i) \right] 
\leq e^{-E(Y_i)/8} = e^{-d'(H_i)/16}
\]

where the first inequality is Claim 3 and the last inequality follows from Claim 4.

Claim 5. \( d'(H_i) \geq \frac{1}{2g} r'_i \).

Proof. Note that forest \( H_i \) consists of exactly \( s_i \) components. Consider a tree \( T_i \) obtained by adding \( s_i - 1 \) edges connecting the \( s_i \) subtrees of \( H_i \). Clearly \( T_i \) is a Steiner tree on \( X_i \), so Claim 1 implies \( d(T_i) \geq r_i - 1 \). Since each edge has \( d \)-length at most 1, we obtain \( d(T_i) \geq d(H_i) - (s_i - 1) \geq \frac{1}{2} r'_i \). By the definition of metric \( d' \), we have \( d'(T_i) = \frac{1}{2} d'(H_i) \geq \frac{1}{2} r'_i \).

Using Claim 4 with the above, \( E(Y_i) \geq \frac{1}{4} r'_i \geq 4 \cdot r'_i \). Using inequality (1).

It is now easy to prove the performance guarantee of the algorithm. The main ingredient of this result is the following (slight) generalization of the FRT tree embedding [4]. A tree metric on vertices \( V \) is given by a tuple \((T, \kappa)\) where \( T \) is a tree on vertex-set \( V \geq V \) and \( \kappa : \left(\frac{1}{2}\right) \rightarrow \mathbb{R}_+ \) is a metric on \( T \) satisfying for all \( x, y \in V \), \( \kappa(x, y) = \sum_{e \in E_T} \kappa_e \) where \( \kappa_e \) denotes edges on the unique \( x \)-\( y \)-path in \( T \). Note that although we are only interested in distances between vertices of \( V \), we allow the tree to be defined on a vertex-set larger than \( V \).

Theorem 7. Suppose \( d \) is a metric on a finite vertex-set \( V \) with a designated subset \( W \subseteq V \). Then there is a distribution \( T \) of tree metrics on \( V \) satisfying:

1. Non-contracting. For all \((T, \kappa) \in \mathcal{T} \), \( \kappa(x, y) \geq d(x, y) \) for all \( x, y \in W \).
2. Low stretch. \( E_T [\kappa(x, y)] \leq \rho \cdot d(x, y) \), for all \( u, v \in V \), where \( \rho = O(\log |W|) \).

Furthermore, trees from this distribution can be sampled in polynomial time.

Proof. The proof of this theorem is essentially identical to that in [4] (where \( W = V \)). We only state the modification required in the algorithm, and how the analysis follows from [4]. We will describe an algorithm that generates a random tree metric from the desired distribution \( T \).

We assume without loss of generality that the smallest distance in \( d \) is 1. Choose \( \delta \in N \) such that \( 2^{\delta - 1} \leq \max_{x, y \in V} d(x, y) < 2^{\delta} \). A cluster refers to any subset of vertices \( V \), and a partial partition is a collection of disjoint clusters. Extending a definition from [4], we define a hierarchical cut decomposition of \((V, d)\) as a sequence \( D_0, \ldots, D_t \) of partial partitions such that \( D_t = \{V\} \) and:

- For each \( 0 \leq i < \delta \), \( D_i \) is a partition of some subset of clusters from \( D_{i+1} \).
- Each cluster in \( D_i \) has radius at most \( 2^\delta \), for all \( 0 \leq i \leq \delta \).

Hierarchical cut decomposition to a tree metric. Any hierarchical cut decomposition of \((V, d)\) naturally corresponds to a rooted tree \( T \) as follows. Each cluster in \( \cup_{i=0}^t D_i \) corresponds to a vertex in \( T \), and the vertex corresponding to cluster \( V \) (in \( D_0 \)) is called the root. The children in \( T \) (if any) of cluster \( S \in D_{i+1} \) are those clusters of \( D_i \) that are subsets of \( S \). We also define a length function \( l \) on edges of \( T \), where any edge between a \( D_{i+1} \)-cluster and a \( D_i \)-cluster has length \( 2^\delta \). We now extend tree \( T \) to obtain another tree \( T' \) that has vertices \( V \) as its leaves: each leaf vertex of \( T' \) (corresponding to say cluster \( S \)) has \( |S| \) new children in \( T \) that are labeled \([v] \in S \). The new edges that are introduced in \( T' \) all have zero length (under function \( l \)). Let \( U \) denote the vertices of the resulting tree \( T' \); note that the internal vertices of \( T' \) correspond to clusters in \( \cup_{i=0}^t D_i \) and the leaf vertices of \( T' \) are precisely \( V \). Finally we define metric \( \kappa : \left(\frac{1}{2}\right) \rightarrow \mathbb{R}_+ \) such that for any \( x, y \in U \), \( \kappa(x, y) := \sum_{e \in E_T} l(e) \) (here \( \kappa_e \) denotes edges on the unique \( x \)-\( y \)-path in \( T' \)). Thus any hierarchical cut decomposition of \((V, d)\) corresponds to a tree metric \((T, \kappa)\) on \( V \).

The algorithm. We now present a modified version of the algorithm from [4] that creates the desired hierarchical cut decomposition (cf. algorithm Partition \((V, d)\) [4]). There are two differences from [4]: we take a permutation on \( W \) (instead of a permutation on the entire vertex-set \( V \)), and a cluster’s decomposition stops when it has at most one \( W \)-vertex (as opposed to stopping when it has at most one \( V \)-vertex).

1. Choose u.a.r. permutation \( \pi \) of \( W \).
2. Choose \( \beta \) u.a.r. from \([1, 2]\).
3. Set \( \delta := \lceil \log_2 (\max_{u,v \in W} d(\pi(u), \pi(v))) \rceil \), \( D_0 := |V|, i := \delta - 1 \).
4. While \( D_{i+1} \) has any cluster containing more than one \( W \)-vertex:
   a. For each cluster \( S \) in \( D_{i+1} \) containing more than one \( W \)-vertex, do:
Theorem 7. There is an LP rounding algorithm for requirement cut.

1. Solve the LP relaxation $LP(I)$ to obtain metric $(V, d^*)$.
2. Apply Theorem 7 for $(V, d^*)$ with $W = \bigcup_{i=1}^{k} X_i$ to obtain a random tree metric $(T, \kappa)$ on $V$. Let $V'$ and $E'$ denote the vertex-set and edge-set of $T$.
3. Define the cost of each edge $e \in E'$ as $c'_e := \sum_{j \in \text{Sep}_e} c_j$, where $\text{Sep}_e = \{(i, j) \in E' | e \text{ lies on the } i - j \text{ path in } T\}$. Define $\kappa'_{(u,v)} := \min\{\kappa_{(u,v)}, 1\}$ for all $u, v \in V'$.
4. Consider an instance $J$ of requirement cut on tree $T$ with costs $c'$ and requirements $\{X_i, r_i\}_{i=1}^{k}$.
5. Treating $\kappa'$ as a fractional solution to linear program $LP(J)$, round it using Theorem 6 to an integral solution $S \subseteq E'$.

Output: the edges $\bigcup_{e \in S} \text{Sep}_e$ as the solution to $J$.

Theorem 8. There is an LP rounding algorithm for requirement cut that returns a solution of cost $O((\log k \cdot \log g) \cdot LP_{opt}$ with probability at least $\frac{1}{2}$, where $LP_{opt}$ is the optimal LP value. Here $g$ is the number of groups, and $k = |\bigcup_{i=1}^{k} X_i|$ is the number of vertices participating in the groups.

Fig. 2. Algorithm for requirement cut on general graphs.

The algorithm for requirement cut is identical to that of [8]: the only difference is the use of this stronger tree embedding in Step 2 below. Given an instance $I = \langle V, E, c, \{X_i, r_i\}_{i=1}^{k}\rangle$ of requirement cut, this proceeds as in Fig 2.

The non-contracting condition in Theorem 7 with $W = \bigcup_{i=1}^{k} X_i$ suffices to show (as in [8]) that $\kappa'$ is a feasible fractional solution to $LP(f)$. Namely, for any $x, y \in W$, we have $d'_{(x,y)} \leq \kappa_{(x,y)}$ and $d_{(x,y)} \leq 1$; so $\kappa'_{(x,y)} = \min\{\kappa_{(x,y)}, 1\} \geq d_{(x,y)}$. Hence for any spanning tree $T'$ on graph $X_i$, $T'$'s length under $\kappa'$ is at least as much as its length under $d^*$ (since $X_i \subseteq W$), which is at least $r_i - 1$ by the feasibility of $d^*$ in $LP(f)$. So $\kappa'$ is a feasible fractional solution to $LP(f)$.

Exactly as in [8], the low stretch condition in Theorem 7 now implies that the expected cost (under $c'$) of metric $\kappa'$ is at most $\sum_{e \in E'} c'_e \cdot d'_e$. Here $\rho = O(\log k)$, for $k = |W| = \bigcup_{i=1}^{k} X_i$. Using the $O(\log g)$ LP rounding algorithm of Theorem 6 in Step (5), it follows that the cost of solution $S$ is $O(\rho \cdot \log g) \sum_{e \in E'} c'_e \cdot d'_e$. Finally it is easy to see that if $S$ is a feasible solution to $f$, then $\sum_{e \in S} \text{Sep}_e$ is a feasible solution to $J$, and $\sum_{e \in S} c'_e = \sum_{f \in \bigcup_{e \in S} \text{Sep}_e} c_j$. This proves the following.

Fig. 2. Algorithm for requirement cut on general graphs.

References