An Approximation Algorithm for the Generalized $k$-Multicut Problem

Peng Zhang †  Daming Zhu ‡  Junfeng Luan §

May 6, 2012

Abstract

Given a graph $G = (V, E)$ with nonnegative costs defined on edges, a positive integer $k$, and a collection of $q$ terminal sets $D = \{S_1, S_2, \ldots, S_q\}$, where each $S_i$ is a subset of $V(G)$, the Generalized $k$-Multicut problem asks to find a set of edges $C \subseteq E(G)$ at the minimum cost such that its removal from $G$ cuts at least $k$ terminal sets in $D$. A terminal subset $S_i$ is cut by $C$ if all terminals in $S_i$ are disconnected from one another by removing $C$ from $G$. This problem is a generalization of the $k$-Multicut problem and the Multiway Cut problem. The famous Densest $k$-Subgraph problem can be reduced to the Generalized $k$-Multicut problem in trees via an approximation preserving reduction.

In this paper, we first give an $O(\sqrt{q})$-approximation algorithm for the Generalized $k$-Multicut problem when the input graph is a tree. The algorithm is based on a mixed strategy of LP-rounding and greedy approach. Moreover, the algorithm is applicable to deal with a class of NP-hard partial optimization problems. As its extensions, we then show that the algorithm can be used to give $O(\sqrt{q \log n})$-approximation for the Generalized $k$-Multicut problem in undirected graphs and $O(\sqrt{q})$-approximation for the $k$-Forest problem.

Keywords. $k$-Multicut, $k$-Forest, LP-rounding, Approximation Algorithm, Combinatorial Optimization.

1 Introduction

The cut problems have been a classical and active topic in combinatorial optimization and approximation algorithms for a long time. The problems find lots of applications in many areas such as telecommunication, routing, transportation and VLSI design. The Multicut problem [10, 11] is one of the most well-known cut problems. Roughly speaking, given an edge-costed graph and a set of terminal pairs, the Multicut problem asks for an edge set with minimized total cost such that its removal from the input graph disconnects all the terminal pairs. Its partial version, i.e., the $k$-Multicut problem [17, 14, 18, 20], which

*A preliminary version of this paper appears in proceedings of the 2nd IEEE International Joint Conference on Computational Sciences and Optimization (CSO 2009) [22].

†School of Computer Science and Technology, Shandong University, Jinan 250101, China. Supported by NSFC 60970003 and China Postdoctoral Science Foundation 20080441144, 200902562. Email: algzhang@sdu.edu.cn.

‡School of Computer Science and Technology, Shandong University, Jinan 250101, China. Email: dmzhu@sdu.edu.cn.

§School of Computer Science and Technology, Shandong University, Jinan 250101, China. Supported by NSFC 60970003. Email: jfluan@sdu.edu.cn.
focuses on disconnecting at least \( k \) terminal pairs at the minimum cost, has also been well studied. In this paper we consider the Generalized \( k \)-Multicut problem. First we give the formal definitions for the \( k \)-Multicut problem and the Generalized \( k \)-Multicut problem.

**Definition 1.1 (The \( k \)-Multicut Problem)** (Instance) In the \( k \)-Multicut problem, we are given a graph \( G = (V, E) \) with nonnegative \( \{c_e\} \) defined on edges, a terminal pair set \( D = \{(s_1, t_1), (s_2, t_2), \ldots, (s_q, t_q)\} \), where all terminals are vertices in \( V(G) \), and an integer \( k \geq 0 \).

(\text{Query}) We are asked to find a minimum cost edge set \( C \subseteq E(G) \) such that its removal from \( G \) cuts at least \( k \) terminal pairs in \( D \). We say that a terminal pair \( (s, t) \in D \) is cut by edge set \( C \) if \( s \) and \( t \) are disconnected by removing \( C \) from \( G \).

**Definition 1.2 (The Generalized \( k \)-Multicut Problem)** (Instance) The input of the Generalized \( k \)-Multicut problem consists of a graph \( G = (V, E) \) with nonnegative edge costs \( \{c_e\} \), a collection of terminal sets \( D = \{S_1, S_2, \ldots, S_q\} \), where each \( S_i \) is a subset of \( V(G) \), and an integer \( k \geq 0 \).

(\text{Query}) The goal of the problem is to find a minimum cost edge set \( C \subseteq E(G) \) such that its removal from \( G \) cuts at least \( k \) terminal sets in \( D \). We say that a terminal set \( S \in D \) is cut by edge set \( C \) if all terminals in \( S \) are disconnected from one another by removing \( C \) from \( G \).

As defined above, the generalization of a multicut problem means that every terminal pair in the problem is extended to a terminal set. In this sense we have the Generalized Multicut problem (its definition can be easily deduced and is omitted here for simplicity) and the Generalized \( k \)-Multicut problem. Given a graph and a terminal set, the Multiway Cut problem [8, 5, 16] finds an edge set that cuts the terminal set at the minimum cost. It is easy to see that the Generalized \( k \)-Multicut problem contains both the \( k \)-Multicut problem and the Multiway Cut problem as its special cases.

We note here that all graphs we consider in this paper are undirected. As usual, given an input graph, we use \( n \) to denote the number of its vertices, and \( m \) to denote the number of its edges.

**Motivation.** We consider the Generalized \( k \)-Multicut problem in trees (\( k \)-GMC(T)) and in undirected graphs (\( k \)-GMC(G)), respectively. We study the Generalized \( k \)-Multicut problem since on the one hand, the problem is a natural generalization of the \( k \)-Multicut problem, and on the other hand, we found (by the first author in [21]) an interesting relationship between the \( k \)-GMC(T) problem and the famous Densest \( k \)-Subgraph (D\&S) problem.

In [21, Subsection 3.1], it was proved by an approximation preserved reduction that if \( k \)-GMC(T) can be approximated within a factor of \( f(n) \), then the Minimum \( k \)-Edge Coverage (M\&EC) problem [15] can be approximated within a factor of \( f(n) \). Note that each terminal set is of size only 3 in the instance of \( k \)-GMC(T) used in the reduction. Combined with the result [15] that if M\&EC can be approximated within \( f(n) \), then D\&S can be approximated within \( 2f^2(n) \), this implies that an \( f(n) \)-approximation algorithm for \( k \)-GMC(T) would lead to a \( 2f^2(n) \)-approximation algorithm for D\&S.

So the \( k \)-GMC(T) problem is almost as hard as the D\&S problem, although \( k \)-GMC(T) is defined in trees. By contrast, the \( k \)-Multicut in trees problem admits constant approximation (see Subsection 1.1). This means that for the \( k \)-GMC(T) problem, simply augmenting the size of each terminal set from 2 to 3 significantly increase the difficulty of approximating it.
The approximation ratio $O(n^{1/3-\epsilon'})$ for some specific constant $\epsilon' > 0$ for DkS [9] has been known for a long time. Very recently, an improved ratio $O(n^{1/4+\epsilon})$ for any constant $\epsilon > 0$ for DkS just appeared [3].

1.1 Related Works

The $k$-GMC(T) problem (as well as the $k$-GMC(G) problem) is clearly both NP-hard and MAX SNP-hard since so is the Multicut in trees problem [11]. $k$-GMC(T) was first proposed in [21] and a preliminary approximation ratio $\frac{2}{3}(q + 1)$ for the problem was given therein. To the best of our knowledge, we do not know any approximation result for the $k$-GMC(G) problem.

For the Multicut in trees problem, Garg, Vazirani and Yannakakis [11] gave the currently best known approximation ratio 2 via a primal-dual approximation algorithm, which is referred to as the GVY algorithm in this paper. The Generalized Multicut in trees (GMC(T) for short) problem can be easily reduced to the Multicut in trees problem by considering for each terminal set $S \in D$ all possible pairs of terminals in $S$, and thus can be approximated within a factor of 2 by the GVY algorithm. Bentz [2] studied the complexity of the Multicut problem and its many variants in bounded tree-width graphs and digraphs.

For the $k$-Multicut in trees problem, Levin and Segev [17] gave a $\frac{3}{4} + \epsilon$-approximation algorithm for arbitrarily small constant $\epsilon > 0$. Their approach is based on the Lagrangian relaxation technique, which reduces the partial version of the Multicut in trees problem (i.e., the $k$-Multicut in trees problem) to its prize-collecting version (i.e., the Prize-collecting Multicut in trees problem) [17]. The $\frac{3}{4} + \epsilon$-approximation for $k$-Multicut in trees was also independently obtained by Golovin, Nagarajan and Singh [14]. By more careful analysis to the Lagrangian relaxation technique, Mestre [18] was able to give an improved $2 + \epsilon$-approximation algorithm for $k$-Multicut in trees, for any small constant $\epsilon > 0$.

For the Multicut in undirected graphs problem, Garg, Vazirani and Yannakakis [10] gave an $O(\log q)$-approximation algorithm via a region growth based LP-rounding approach. Later, Avidor and Langberg [1] extended the region growth technique to deal with the Multi-Multiway Cut in undirected graphs problem (known as the Generalized Multicut in undirected graphs (GMC(G) for short) problem in this paper) and obtained approximation ratio $O(\log q)$ for the problem. (Note that if we apply to GMC(G) the reduction mentioned above for GMC(T), we can only obtain $O(\log n)$-approximation for GMC(G).) Recently, Räcke [20] gave a randomized $O(\log n)$-approximation algorithm for the $k$-Multicut in undirected graphs problem using his famous tree decomposition technique.

Nagarajan and Ravi [19] considered a more generalized cut problem, called the Requirement Cut problem, which asks to find a minimum cost set of edges such that its removal separates each terminal set $S_i$ into at least $r_i$ disconnected components, where $r_i$’s are parameters given in the input. They gave an $O(\log n \log(q \max_i \{r_i\}))$-approximation algorithm for the Requirement Cut problem. The Generalized $k$-Multicut problem studied in this paper can be viewed as the partial version of the Requirement Cut problem when $r_i = |S_i|$ for all $i$.

1.2 Our Results

In this paper, we give an $O(\sqrt{q})$-approximation algorithm for the $k$-GMC(T) problem, improving the previously best known approximation ratio $\frac{2}{3}(q + 1)$ for this problem [21]. Our algorithm is based on a simple greedy approach (see algorithm $A$ in Subsection 2.1)
and an LP-rounding with scaling approach (see the bicriteria approximation algorithm $B$ in Subsection 2.2).

For any small $\epsilon > 0$, algorithm $B$ finds an edge set $C$ in polynomial time such that $C$ cuts at least $(1 - \epsilon)k$ terminal sets in $D$ and its cost is at most $\frac{2}{\epsilon} \cdot \frac{q - k + \epsilon k}{k}$ times the optimum. The analysis of algorithm $B$ exploits an observation that by appropriate scaling, part of any feasible solution to the LP-relaxation of $k$-GMC($T$) can be converted to a feasible solution to the LP-relaxation of the Multicut in trees problem. Inspired by the work for the Prize-collecting Traveling Salesman problem due to Bienstock, Goemans et al. [4] and the work for the Prize-collecting Generalized Steiner Tree problem due to Hajiaghayi and Jain [15], we found a relation between the feasible solutions to the LP-relaxation of the $k$-GMC($T$) problem and to the LP-relaxation of the Multicut in trees problem.

Given that we have already a partial solution which cuts at least $(1 - \epsilon)k$ terminal sets, we need only to cut at most $\epsilon k$ terminal sets to obtain a feasible solution to $k$-GMC($T$). This task can be performed by the greedy algorithm and the approximation ratio of our final algorithm (see algorithm $C$ in Subsection 2.3) is proved to be $O(\sqrt{q})$ by carefully choosing an $\epsilon$.

In fact, we can show the integrality gap of the LP relaxation for $k$-GMC($T$) we use is asymptotically at least $q$. Obviously, algorithm $C$ gets around this gap by a mixed strategy of LP-rounding and greedy approach.

Moreover, algorithm $C$ implies a general approach to deal with a class of NP-hard partial optimization problems. Let us call it the LP-rounding plus greedy approach. As its applications, we show that the LP-rounding plus greedy approach can be used to give $O(\sqrt{q})$-approximation for the $k$-Forest problem and $O(\sqrt{q \log n})$-approximation for the $k$-GMC($G$) problem. The current best approximation ratio for the $k$-Forest problem is $O(\min\{\sqrt{n}, \sqrt{k}\})$ [13]. Although our approximation ratio $O(\sqrt{q})$ does not improve the current best ratio for $k$-Forest, we give another sub-linear approximation ratio for the problem.

The conference version of algorithm $B$ for $k$-GMC($T$) appeared in [22]. In this paper we further improve the approximation ratio for $k$-GMC($T$) to $O(\sqrt{q})$ and give several extensions of our improved algorithm. The known approximation results for the multicut problems in trees and in undirected graphs are given in Table 1.

<table>
<thead>
<tr>
<th></th>
<th>Trees</th>
<th>Undirected Graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$-Multicut</td>
<td>$2 + \epsilon$ [18]</td>
<td>$O(\sqrt{n})$ [20]</td>
</tr>
<tr>
<td>Generalized $k$-Multicut</td>
<td>$O(\sqrt{q})$ (this paper)</td>
<td>$O(\sqrt{q \log n})$ (this paper)</td>
</tr>
</tbody>
</table>

Table 1: Approximation results for the multicut problems

2 Approximating $k$-GMC($T$)

2.1 A Greedy Approximation Algorithm

We restate the greedy algorithm $A$ for $k$-GMC($T$) in [21] and give a tight example to it.

**Algorithm $A$ for $k$-GMC($T$)**

Input: an instance $(T, D, k)$ of $k$-GMC($T$).
$D = \{ \{v_n, v_{n-1}, v_1\}, \{v_n, v_{n-1}, v_2\}, \ldots, \{v_n, v_{n-1}, v_{n-3}\}\}$

\[
\begin{align*}
\gamma & \quad 1 + \gamma \\
v_n & \quad v_{n-1} \quad v_{n-2} \\
& \quad 1 \\
& \quad v_3 \\
& \quad \vdots \\
& \quad v_{n-3}
\end{align*}
\]

$T$

Output: a cut $C$ whose removal cuts at least $k$ terminal sets in $D$.

1. Optimally solve each instance $(T, S_i)$ of the Multiway Cut in trees problem for $i = 1, \ldots, q$. Let edge set $C_i^*$ be the obtained optimal solution to instance $(T, S_i)$.

2. return the union of the first $k$ cheapest edge sets as the final solution $C$.

For algorithm $A$ we have Theorem 2.1.

**Theorem 2.1** ([21]) Algorithm $A$ is a $k$-approximation algorithm for the $k$-GMC($T$) problem.

**Proof:** Algorithm $A$ runs in polynomial time since the Multiway Cut in trees problem can be optimally solved in polynomial time [6] and even in linear time [7].

Denote by $\text{OPT}_{MWC}(T, S_i)$ the optimum of instance $(T, S_i)$ (i.e., the cost of $C_i^*$). Without loss of generality, assume that the terminal sets $S_i$’s are just in the order such that $\text{OPT}_{MWC}(T, S_1) \leq \text{OPT}_{MWC}(T, S_2) \leq \cdots \leq \text{OPT}_{MWC}(T, S_q)$. Consider an optimal solution $C^* \subseteq E(T)$ to the instance of $k$-GMC($T$) with cost $\text{OPT}$. Let $D^* = \{S'_1, S'_2, \ldots\}$ be the collection of terminal sets cut by $C^*$ (notice that $|D^*| \geq k$). Similarly, without loss of generality we assume that $\text{OPT}_{MWC}(T, S'_1) \leq \text{OPT}_{MWC}(T, S'_2) \leq \cdots \leq \text{OPT}_{MWC}(T, S'_{|D^*|})$. Since $D^*$ is a subset of $D$, we have $\text{OPT}_{MWC}(T, S_i) \leq \text{OPT}_{MWC}(T, S'_i)$ for every $i = 1, \ldots, k$. So at last we have

$$cost(C) \leq \sum_{i=1}^{k} \text{OPT}_{MWC}(T, S_i) \leq \sum_{i=1}^{k} \text{OPT}_{MWC}(T, S'_i) \leq k \cdot \text{OPT},$$

where the last inequality holds since $C^*$ is clearly a feasible solution to instance $(T, S'_i)$ of the Multiway Cut in trees problem for every $i = 1, \ldots, k$.

A tight example for algorithm $A$. Consider the instance of $k$-GMC($T$) given in Figure 1. The cost of the solution found by algorithm $A$ on this instance is $k + \gamma$ while the optimum of the instance is $1 + 2\gamma$ (when $k \geq 2$). As $\gamma$ approaches 0, the approximation ratio of algorithm $A$ on this instance approaches $k$. 

Figure 1: A tight example for algorithm $A$
2.2 A Bicriteria Approximation Algorithm

In this section we give an approximation algorithm, called algorithm $B$, for $k$-GMC(T) with bicriteria performance ratio $(\frac{2(q-k+k+\epsilon)}{ek}, 1 - \epsilon)$, where $\epsilon > 0$ is any small number (not necessarily fixed). For any small number $\epsilon > 0$, the algorithm finds a solution in polynomial time such that it cuts at least $(1 - \epsilon)k$ terminal sets in $D$ and its cost is at most $\frac{2(q-k+k+\epsilon)}{ek}$ times the optimum. The algorithm is based on the LP-rounding approach. The following (LP$_k$-T) is the linear program relaxation for $k$-GMC(T), where the subscript $k$ stands for the $k$-version of the GMC(T) problem.

\[
\min \sum_{e \in E(T)} c_e x_e \quad \text{(LP}_k\text{-T)}
\]

subject to

\[
\sum_{e \in P} x_e + z_i \geq 1, \quad \forall 1 \leq i \leq q, \forall P \in P_i \tag{1}
\]

\[
\sum_{1 \leq i \leq q} z_i \leq q - k \tag{2}
\]

\[
x_e \geq 0, \quad \forall e \in E(T)
\]

\[
z_i \geq 0, \quad \forall 1 \leq i \leq q
\]

Consider the integral version of (LP$_k$-T). The variable $x_e \in \{0, 1\}$ indicates whether edge $e$ is in the cut. Let $C = \{e \in E : x_e = 1\}$. The variable $z_i \in \{0, 1\}$ indicates whether terminal set $S_i$ is cut, that is, $z_i = 0$ means that we should cut $S_i$. Since the input graph is a tree, there is a unique path for every terminal pair $(s, s')$ coming from terminal set $S_i$. The notation $P_i$ in constraint (1) denotes the set of such unique paths for all possible terminal pairs coming from $S_i$. Then constraint (1) states that if terminal set $S_i$ is cut by $C$ (when $z_i = 0$) then every path in $P_i$ should be disconnected. Constraint (2) states that the number of terminal sets that are cut must be at least $k$. The objective function of (LP$_k$-T) is to minimize the total cost of the picked edges. So we know that the integral version of (LP$_k$-T) is really a linear program formulation for the $k$-GMC(T) problem.

In the sense of minimizing the objective function value, the constraints $x_e \leq 1, \forall e$ and $z_i \leq 1, \forall i$ can be omitted in the LP relaxation (LP$_k$-T) for $k$-GMC(T). If there is an edge $e$ such that $x_e > 1$, then we can decrease $x_e$ to 1 while all constraints in (LP$_k$-T) are not violated and the objective function value is strictly decreased. If there is an $i$ such that $z_i > 1$, then there must be a $j \neq i$ such that $z_j < 1$ by constraint (2). So we can decrease $z_i$ by $\min\{z_i - 1, 1 - z_j\}$ and increase $z_j$ by the same amount, without violating constraint (2). Since $z_j$ is strictly increased, all $x_e$’s related to $z_j$ in constraint (1) can be decreased by a certain amount, leaving constraint (1) not violated. This implies the objective function value is strictly decreased.

**Integral gap of (LP$_k$-T).** The $k$-GMC(T) instance in Figure 1 is also an example to show the integrality gap of (LP$_k$-T) is asymptotically at least $q$. Fix $k$ to be 1 in the instance. Denote by $e_j$ the edge to the left of vertex $v_i$, for $1 \leq j \leq n - 1$. Then $\forall 1 \leq i \leq q, z_i = 1 - \frac{1}{q}, x_{e_{q-1}} = x_{e_{q-2}} = \frac{1}{q}$ and $\forall 1 \leq j \leq n - 3, x_{e_j} = 0$ form a fractional optimal solution to (LP$_k$-T) whose value $\text{OPT}_f$ is $\frac{1+2\gamma}{q}$. Since the optimum $\text{OPT}$ of this instance is $1 + \gamma$, the integrality gap of (LP$_k$-T) is at least $\frac{\text{OPT}}{\text{OPT}_f} = \frac{1+\gamma}{1+2\gamma} - q$.

Before giving algorithm $B$, we first give the linear program relaxation (LP$_\gamma$-T) for the
GMC(T) problem, which will be used in algorithm $B$. (The subscript $s$ means the standard version of GMC(T).)

$$
\min \left\{ \sum_{e \in E(T)} c_e x_e \mid \sum_{e \in P} x_e \geq 1, \forall i, \forall P \in \mathcal{P}_i; x_e \geq 0, \forall e \right\} \quad (LP_s-T)
$$

(LP_s-T) is similar to (LP_k-T) and thus we omit its explanation. As mentioned in Subsection 1.1, an integral feasible solution to (LP_s-T) can be found in polynomial time by the GVY algorithm.

**Algorithm $B$ for $k$-GMC(T)**

Input: an instance $(T, D, k)$ of $k$-GMC(T).

Output: a cut $C$ whose removal cuts at least $(1-\epsilon)k$ terminal sets in $D$.

1. Solve (LP_k-T) to obtain an optimal fractional solution $(x^*, z^*)$.
2. $\alpha \leftarrow \frac{q-k}{q-k+\epsilon k}.$
3. **for each** $1 \leq i \leq q$ **do if** $z^*_i > \alpha$ **then** $\hat{z}_i \leftarrow 1$ **else** $\hat{z}_i \leftarrow 0.$
4. $Q \leftarrow \{ i : \hat{z}_i = 0 \}.$
5. Obtain a feasible solution $\hat{x}$ to (LP_s-T) on $Q$ by the GVY algorithm.
6. **return** $C \leftarrow \{ e : \hat{x}_e = 1 \}$.

In step 5, (LP_s-T) on $Q$ is the linear program (LP_s-T) with condition $\forall i$ (that is, $\forall 1 \leq i \leq q$) replaced by $\forall i \in Q$.

**Theorem 2.2** Given any small $\epsilon \in (0, 1)$ (not necessarily fixed) such that the rounding threshold $\alpha$ can be computed in polynomial time, algorithm $B$ outputs in polynomial time a solution to the instance $(T, D, k)$ of $k$-GMC(T) which cuts at least $(1-\epsilon)k$ terminal sets in $D$ at the cost of at most $2 \cdot \frac{2-q-k+\epsilon k}{k} \cdot OPT$, where $OPT$ denotes the optimum of the instance.

**Proof:** Since $0 < \epsilon < 1$, the rounding threshold $\alpha$ is in $(0, 1)$. In algorithm $B$, we devote to cutting all terminal sets whose index is in $Q$. Since $\forall i \in Q$, $z^*_i \leq \alpha$, we know that $\sum_{e \in P} x^*_e \geq 1 - z^*_i \geq 1 - \alpha$ for all $i \in Q$ and $P \in \mathcal{P}_i$ by constraint (1).

Define $x'_e = \min\{\frac{1}{1-\alpha} x^*_e, 1\}$ for every edge $e$. Then, $x'$ is a feasible solution to the linear program (LP_s-T) on $Q$. By the GVY algorithm, the cost of $\hat{x}$ in step 5 is at most $2 \times OPT_f(LP_s-T(Q))$, where $OPT_f(LP_s-T(Q))$ is the (fractional) optimum of (LP_s-T) on $Q$. So we get an integral solution $(\hat{x}, \hat{z})$ to (LP_k-T) satisfying

$$
\sum_{e \in E(T)} c_e \hat{x}_e \leq 2 \times OPT_f(LP_s-T(Q)) \leq 2 \sum_{e \in E(T)} x'_e
$$

$$
\leq 2 \times \frac{2}{1-\alpha} \sum_{e \in E(T)} x^*_e \leq 2 \times \frac{2}{1-\alpha} OPT.
$$

Denote by $Z$ the number of terminal sets which are not cut by the solution $(\hat{x}, \hat{z})$, i.e., $Z = \sum_i \hat{z}_i$. By step 3, $\hat{z}_i = 1$ if and only if $z^*_i > \alpha$. Then we know that $Z < q - (1-\epsilon)k$. 

7
Otherwise the number of $i$’s such that $z_i^* > \alpha$ would be at least $q - (1 - \epsilon)k$ and hence 
\[ \sum_i z_i^* > (q - (1 - \epsilon)k) \cdot \alpha = q - k, \] 
violating constraint (2) of (LP$_k$-T). So the solution $(\hat{x}, \hat{z})$ cuts at least 
$q - Z > (1 - \epsilon)k$ terminal sets in $D$.

Given that the rounding threshold $\alpha$ can be computed in polynomial time, we know that 
algorithm $B$ runs in polynomial time. This concludes the theorem.

### 2.3 The Final Approximation Algorithm

We have already known that for any $0 < \epsilon < 1$, algorithm $B$ will give a bicriteria solution which cuts at least $(1 - \epsilon)k$ terminal sets. If it actually cuts at least $k$ terminal sets, then we are done. Otherwise we have to cut another $\epsilon k$ terminal sets to get a feasible solution to the $k$-GMC(T) problem. But we just have the greedy algorithm $A$ to complete this task, since the task is in fact a new $k$-GMC(T) instance. This idea gives us the final algorithm $C$.

**Algorithm $C$ for $k$-GMC(T)**

Input: an instance $I = (T, D, k)$ of $k$-GMC(T).

Output: a cut whose removal cuts at least $k$ terminal sets in $D$.

1. if $k \leq \sqrt{q}$ then

2. call algorithm $A$ to obtain a solution $C$.

3. return $C$ as the final solution to $I$ and stop.

4. endif

5. call algorithm $B$ with $\epsilon \leftarrow \frac{\sqrt{q}}{k}$ to obtain a solution $C'$.

6. $k' \leftarrow |D'|$, where $D'$ is the collection of terminal sets cut by $C'$.

7. if $k' \geq k$ then return $C'$ as the final solution to $I$ and stop.

8. $D_0 \leftarrow D - D'$, $k_0 \leftarrow k - k'$. This defines a new instance $J = (T, D_0, k_0)$ of $k$-GMC(T).

9. call algorithm $A$ on instance $J$ to obtain a solution $C_0$.

10. return $C'' \leftarrow C' \cup C_0$ as the final solution to $I$.

**Theorem 2.3** Algorithm $C$ is an $O(\sqrt{q})$-approximation algorithm for the $k$-GMC(T) problem.

**Proof:** If $k \leq \sqrt{q}$, algorithm $C$ will return the edge subset $C$ as the final solution, which is found by the greedy algorithm $A$. By Theorem 2.1, $C$ is a $k$-approximate solution to instance $I$. The theorem is proved in this case. So in the following we assume that $k > \sqrt{q}$.

Since $k > \sqrt{q}$ and $\epsilon = \frac{\sqrt{q}}{k}$, we know that $0 < \epsilon < 1$. Recall that $k'$ is the number of terminal sets cut by $C'$. Denote by $OPT(I)$ the optimum of instance $I$. By Theorem 2.2, for the edge subset $C''$ and the number $k'$ we have

\[
\text{cost}(C') \leq \frac{2(q - (1 - \epsilon)k)}{\epsilon k} \cdot OPT(I) \leq 2\sqrt{q} \cdot OPT(I)
\]
and $k' \geq (1-\epsilon)k$. If it happens that $k' \geq k$, algorithm $C$ will return $C'$ and the theorem follows.

Then let us consider the case $k' < k$. In this case, algorithm $C$ calls the greedy approximation algorithm $A$ to obtain a solution $C_0$ to the new instance $J = (T, D_0, k_0)$. Since $C_0$ cuts at least $k_0 = k - k'$ terminal sets in $D_0 = D - D'$, it is obvious that $C'' = C' \cup C_0$ is a feasible solution to instance $I$.

Suppose that $D^* \subseteq D$ is the collection of terminal sets cut by an optimal solution $C^*$ to instance $I$. Since $|D'| = k' < k$ and $|D^*| \geq k$, there are at least $k_0 = k - k'$ terminal sets in $D^* - D'$. Since $C^*$ obviously cuts all terminal sets in $D^* - D'$, $C^*$ is a feasible solution to instance $J$. It turns out that $\text{OPT}(J) \leq \text{cost}(C^*)$. By Theorem 2.1, $C_0$ is a $k_0$-approximate solution to instance $J$. So we get that $\text{cost}(C_0) \leq k_0 \cdot \text{OPT}(I) \leq k_0 \cdot \text{OPT}(I)$. Thus for the edge subset $C''$ we have

$$\text{cost}(C'') \leq \text{cost}(C') + \text{cost}(C_0) \leq (2\sqrt{q} + k_0) \cdot \text{OPT}(I) \leq (2\sqrt{q} + \epsilon k) \cdot \text{OPT}(I) = 3\sqrt{q} \cdot \text{OPT}(I).$$

Algorithm $A$ runs in polynomial time. It is obvious that the rounding threshold $\alpha$ used in algorithm $B$ can be computed in polynomial time when $\epsilon$ is set to be $\frac{\sqrt{q}}{k}$. So algorithm $B$ runs in polynomial time. It follows that algorithm $C$ is in polynomial time. This concludes the theorem.

### 3 Extensions

Algorithm $C$ for the $k$-GMC(T) problem implies a general approach to deal with the $k$-version problems of a class of NP-hard optimization problems; first use a bicriteria approximation algorithm similar to algorithm $B$ to produce a partially feasible solution to the problem, where the unfinished task forms a new instance of the problem; and then use an (greedy) algorithm similar to algorithm $A$ to produce a feasible solution to the new instance which makes the partial solution a completely feasible solution. This approach relies on three ingredients: 1) the problem admits a $k$-approximation algorithm; 2) any feasible solution to the LP relaxation of the $k$-version problem can be converted to a feasible solution to the LP relaxation of the standard version of the problem; 3) there is an approximation algorithm based on the LP relaxation of the standard version problem with good approximation ratio.

Let us call this approach the LP-rounding plus greedy approach. We give two examples for this approach.

#### 3.1 $k$-GMC(G)

Recall that $k$-GMC(G) denotes the Generalized $k$-Multicut in undirected graphs problem. To apply the LP-rounding plus greedy approach to $k$-GMC(G), the greedy algorithm for the problem have to compute a multiway cut for each terminal set $S_i$ in graph $G$ and output as a solution to $k$-GMC(G) the union of the first $k$ cheapest multiway cuts among all the found cuts. As the Multiway Cut problem in undirected graphs is NP-hard [8], the greedy algorithm actually finds a $\rho$-approximate solution to each instance $(G, S_i)$ of the Multiway Cut problem. The current best value for the approximation ratio $\rho$ is $1.3438$ [16].

Denote by $\text{apx}(G, S)$ the cost of the solution found on Multiway Cut instance $(G, S)$. Rearrange the $q$ costs such that $\text{apx}(G, \bar{S}_1) \leq \text{apx}(G, \bar{S}_2) \leq \cdots \leq \text{apx}(G, \bar{S}_q)$. Without loss
of generality, assume the terminal sets in $D$ are just in the order that $OPT_{MWC}(G, S_1) \leq OPT_{MWC}(G, S_2) \leq \cdots \leq OPT_{MWC}(G, S_q)$. We have $\sum_{i=1}^{k} \text{apx}(G, \tilde{S}_i) \leq \sum_{i=1}^{k} \text{apx}(G, S_i) \leq \rho \sum_{i=1}^{k} OPT_{MWC}(G, S_i)$. Then using similar argument as that in Theorem 2.1, we know that the greedy algorithm for $k$-GMC(G) outputs a $\rho k$-approximate solution in polynomial time.

The LP relaxation for $k$-GMC(G), denoted by $(LP_k^G)$, is the same as the LP relaxation $(LP_k^T)$ for $k$-GMC(T), except that the LP is defined on graph $G$. Note that in $(LP_k^G)$, $P_j$ is the set of all possible paths for every terminal pair $(s, s')$ coming from terminal set $S_i$. Hence in $P_j$ there may be exponential number of paths. But $(LP_k^G)$ can still be optimally solved in polynomial time using the Ellipsoid method, since there is a polynomial time separation oracle. For every terminal set $S_i$ and every terminal pair $(s, s')$ from $S_i$ (there are $\binom{|S_i|}{2}$ such pairs), compute the shortest $s$-$s'$ path using $\{x_e\}$ as distance labels on edges. If there is a pair $(s, s')$ from $S_i$ whose shortest path length is less than $1 - z_i$, then we find a violated inequality for $(LP_k^G)$.

Similarly, we can get the LP relaxation $(LP_{sG})$ for the Generalized Multicut in undirected graphs problem according to $(LP_{sT})$. There are at most $\binom{n}{2}$ terminal pairs coming from all terminal sets in $D$. So by the region-growth approximation algorithm due to Garg et al. [10], we get an integral solution to $(LP_{sG})$ in polynomial time such that its cost is at most $\beta = O(\log n)$ times $OPT_{T}(LP_{sG})$. Suppose that $\beta = c_0 \log n$ for some constant $c_0$. Using the LP-rounding method in algorithm $B$, for any small number $\epsilon > 0$, we can get a solution to $k$-GMC(G) which cuts at least $(1 - \epsilon)k$ terminal sets in $D$ and whose cost is at most $c_0 \log n \cdot q \frac{1 - \epsilon}{k}$ times the optimum of the $k$-GMC(G) problem.

Then we are ready to describe the final approximation algorithm for $k$-GMC(G). If $k \leq \sqrt{q \log n}$, we return the solution found by the greedy algorithm. The performance ratio is $\rho \cdot k = O(\sqrt{q \log n})$ in this case. Else we apply the LP-rounding plus greedy approach as in algorithm $C$ by setting $\epsilon = \frac{\sqrt{n \log n}}{k}$. The performance ratio is $O(\sqrt{q \log n})$ in this case. So at last we compute an $O(\sqrt{q \log n})$-factor approximation for $k$-GMC(G) in polynomial time.

### 3.2 $k$-Forest

Given an undirected graph $G = (V, E)$ with costs $\{c_e\}$ defined on edges, a terminal pair collection $D = \{(s_1, 1), (s_2, t_2), \ldots, (s_q, t_q)\}$, and a positive integer $k$, the $k$-Forest problem asks to find a subgraph $F$ of $G$ at the minimum cost $c(F) = \sum_{e \in E(F)} c_e$ such that at least $k$ terminal pairs are connected in $F$. In general the subgraph $F$ is a forest. This problem is proposed by Hajiaghayi and Jain [15]. We give an $O(\sqrt{q})$-approximation algorithm for $k$-Forest using the LP-rounding plus greedy approach.

The greedy algorithm for $k$-Forest simply outputs the union of paths that are the first $k$ shortest among the $q$ shortest paths for every terminal pair. Thus we get a $k$-approximation for the $k$-Forest problem, because the cost (length) of the path that is the $j$-th shortest among all the shortest paths for terminal pairs in $D$ is at most the cost of the path that is the $j$-th shortest among the shortest paths for the terminal pairs connected in $F^*$, an optimal solution to $k$-Forest.

The LP relaxation for the $k$-Forest problem, denoted by $(LP_k^F)$, is similar to $(LP_k^T)$ for $k$-GMC(T): just replace $E(T)$ by $E(G)$ and constraint (1) by the following constraint (3).

$$\sum_{e \in \delta(S)} x_e + z_i \geq 1, \quad \forall 1 \leq i \leq q, \forall S \in S_i$$

(3)
In constraint (3), \( S_i \) is the collection of all vertex subsets \( S \) such that \((S, V(G) - S)\) forms an \( s_i-t_i \) cut, and \( \delta(S) \) is the set of all edges in the cut \((S, V(G) - S)\). Notice that there may be exponential number of inequalities in constraint (3).

\( \text{(LP}_k\text{-F)} \) can be optimally solved by the Ellipsoid algorithm since there is a polynomial time separation oracle for it. For each \( 1 \leq i \leq q \), we compute a minimum \( s_i-t_i \) cut using \( \{x_e\} \) as edge capacities. If for some \( i \) the capacity of the minimum \( s_i-t_i \) cut is strictly less than \( 1 - z_i \), then we obviously find a violated inequality. Otherwise we have a feasible solution to the LP.

Using the same method as in algorithm \( \mathcal{B} \), we can compute a \( \left( \frac{2(q-k+\epsilon_k)}{2}, 1-\epsilon \right) \)-factor bicriteria approximate solution to the \( k \)-Forest problem, as the primal-dual algorithm due to Goemans and Williamson [12] computes an integral solution \( \{\hat{x}_e\} \) to the following LP relaxation (LP\(_{s-F}\)) for the Steiner Forest problem, satisfying that \( \sum_{e \in E(G)} c_e \hat{x}_e \leq 2 \cdot \text{OPT}_{f}(\text{LP}_{s-F}) \).

\[
\min \left\{ \sum_{e \in E(G)} c_e x_e \mid \sum_{e \in \delta(S)} x_e \geq 1, \forall i, \forall S \in \mathcal{S}_i; x_e \geq 0, \forall e \right\} \quad \text{(LP}_{s-F}\text{)}
\]

Then we can get an \( O(\sqrt{q}) \)-approximation in polynomial time for \( k \)-Forest using the LP-rounding plus greedy approach as what does in algorithm \( \mathcal{C} \).

4 Concluding Remarks

We study the \( k \)-GMC(T) problem in this paper. Based on a greedy approach and an LP-rounding with scaling approach, we give an \( O(\sqrt{q}) \)-approximation algorithm for the \( k \)-GMC(T) problem. Our approach applies to a class of \text{NP}-hard partial optimization problems. As its extensions, we give an \( O(\sqrt{q \log n}) \)-approximation algorithm for the \( k \)-GMC(G) problem and an \( O(\sqrt{q}) \)-approximation algorithm for the \( k \)-Forest problem.

By the reduction presented in [21, Subsection 3.1], an \( O(\sqrt{q}) \)-approximation algorithm for \( k \)-GMC(T) implies an \( O(\sqrt{m}) \)-approximation algorithm for \( MkEC \) (we omit the details here). However, the \( O(\sqrt{m}) \)-approximation algorithm for \( MkEC \) cannot lead to nontrivial approximation algorithm for \( DkS \) due to the somewhat large factor \( \sqrt{m} \). Furthermore, since \( q \) is the number of terminal sets, it is incomparable to the edge number \( m \) and the vertex number \( n \). Thus to obtain approximation algorithm for \( k \)-GMC(T) (and \( k \)-GMC(G)) whose performance ratio is sub-linear in \( m \) or \( n \) remains an interesting open problem.

Acknowledgement

The authors are very grateful to the anonymous referees for their valuable comments that significantly improve the presentation of the paper.

Part of the work was done when the first author visited Department of Mathematical and Computing Sciences, Tokyo Institute of Technology. The author is very grateful to Prof. Osamu Watanabe and his research group for their helpful discussion and hospitality.

References


