An FPTAS for the fractional group Steiner tree problem

Slobodan Jelić[1,*]

[1] *Department of Mathematics, Josip Juraj Strossmayer University of Osijek*
Trg Ljudevita Gaja 6, 31000 Osijek, Croatia
E-mail: ⟨*sjelic@mathos.hr*⟩

Abstract. This paper considers a linear relaxation of the cut-based integer programming formulation for the group Steiner tree problem (FGST). We combine the approach of Koufogiannakis and Young (2013) with the nearly-linear time approximation scheme for the minimum cut problem of Christiano et. al (2011) in order to develop a fully polynomial time approximation scheme for FGST problem. Our algorithm returns the solution to FGST where the objective function value is a maximum of $1+6\varepsilon$ times the optimal, for $\varepsilon \in \langle 0, 1/6]$, in $\tilde{O}(mk(m + n^{4/3}\varepsilon^{-16/3})/\varepsilon^2)$ time, where n, m and k are the numbers of vertices, edges and groups in the group Steiner tree instance, respectively. This algorithm has a better worst-case running time than algorithm by Garg and Khandekar (2002) where the number of groups is sufficiently large.

Key words: approximation algorithm, fully polynomial time approximation scheme, Lagrangean relaxation, group Steiner tree problem, fractional group Steiner tree problem , covering linear program, packing linear program

Received: September 24, 2015; accepted: October 15, 2015; available online: October 31, 2015

DOI: 10.17535/crorr.2015.0039

1. Introduction

Problem definition. The group Steiner tree (GST) problem was introduced by Reich and Widmayer [24], motivated by the problem of wire routing using multiport terminals in a physical VLSI design. We are given an undirected graph $G = (V, E)$, $|V| = n$, $|E| = m$, with edge-weight function $w : E \to \mathbb{R}_+$, and a family of subsets of V, $\mathcal{G} = \{G_1, \dots, G_k\}$, $k \in \mathbb{N}$, $G_i \neq \emptyset$ which are called *groups*. The problem is to find a subtree T such that

$$\sum_{e \in E(T)} w(e)$$

is minimized and $V(T) \cap G_i \neq \emptyset$ for each $i \in [k]$. An instance of the group Steiner tree problem is denoted by $(G, \mathcal{G}, w)$. We consider the algorithm for the *rooted version* where the pre-specified vertex r is required in the solution subtree. Vertex r is called

*Corresponding author.

a *root*. In order to solve the unrooted version, we solve the rooted version for all vertices in the smallest group as possible choices for root r, and take the solution of the smallest weight.

Let $\mathcal{S}_r = \bigcup_{i=1}^{k}\{S \subseteq V \setminus \{r\} : G_i \subseteq S\}$ be a family of all subsets S of $V \setminus \{r\}$ such that the cut $(S, V \setminus S)$ separates a group G_i from the pre-specified root r and $\delta(S) = \{\{s,t\} \in E : s \in S, t \notin S\}$. We are now ready to give a *natural cut-based integer programming formulation* of this problem :

$$\begin{aligned} &\min \textstyle\sum_{e\in E} w(e) z_e \\ s.t. \quad &\textstyle\sum_{e\in\delta(S)} z_e \geq 1, \quad S \in \mathcal{S}_r, \\ &z_e \in \{0,1\}, \quad e \in E. \end{aligned} \tag{1}$$

In this paper, we consider a relaxed version of Problem (1) where the integrality constraints $z_e \in \{0,1\}$, $e \in E$ are replaced by non-negativity constraints $z_e \geq 0$, $e \in E$,

$$\begin{aligned} &\min \textstyle\sum_{e\in E} w(e) z_e \\ s.t. \quad &\textstyle\sum_{e\in\delta(S)} z_e \geq 1, \quad S \in \mathcal{S}_r, \\ &z_e \geq 0, \quad e \in E. \end{aligned} \tag{2}$$

The LP Problem (2) is called the *fractional group Steiner tree problem* (FGST). We provide an interpretation in terms of a flow network by adding new vertices and using the max-flow min-cut theorem. Let us assume the introduction of a new vertex g_i for each group G_i and a directed edge from g_i to each vertex $v \in G_i$ with infinite capacity. The value of variable z_e is interpreted as the capacity of edge e. The conditions in (1) ensure that the capacity of each cut separating some group G_i from root r (or equivalently, a g_i from r) is at least one. Using the max-flow min-cut theorem, it becomes evident that capacities z_e are sufficient to send at least one unit of flow from each vertex g_i, to root r.

Motivation and related problems. The group Steiner tree problem generalizes two important problems: The Steiner tree problem and the set cover problem. The Steiner tree problem is one of the most important NP-hard problems in combinatorial optimization that admits an approximation algorithm with a constant approximation ratio [4, 5]. Actually, it is NP-hard for approximating it within a ratio of less than 96/95 [6]. In the Steiner tree problem, we are given an undirected graph $G = (V, E)$, $|V| = n$, $|E| = m$, with an edge-weight function $w : E \to \mathbb{R}_+$ and a subset of vertices $\emptyset \neq R \subseteq V$ called *terminals*. Vertices in $V \setminus R$ are called *Steiner vertices*. The task is to find a minimum-weight subtree T that spans all terminals. It is obvious that the Steiner tree problem is reducible to a special case of the GST problem, where the size of each group is at most one. The set cover problem is the second one, but no less important since it generalizes a number of other combinatorial problems. We are given a set of elements U and a family $\mathcal{U}$ of subsets of U such that $\bigcup_{S\in\mathcal{U}} = U$. We say that the subfamily $\mathcal{R} \subseteq \mathcal{U}$ is a set cover with respect to the instance $(U, \mathcal{U})$ if every $u \in U$ is covered by at least one set from $\mathcal{R}$. The *set cover problem* introduced by Karp [18] is used to find a subfamily $\mathcal{R}$ of minimum size. The more general version of the problem is typically called the *weighted set cover problem* where each set from the family $\mathcal{U}$ has a nonnegative weight associated with it. It is well known that the set cover problem cannot be approximated by an approximation ratio better than $(1 - o(1)) \ln n$, unless NP contains slightly superpolynomial time algorithms [10]. For a given set cover instance $(U,\mathcal{U})$ we construct the star graph where the leaves are associated with sets in $\mathcal{U}$. Each element in U defines a group. Each vertex belongs to groups that correspond to the elements contained in the set associated with that leaf. The weight of each leaf is equal to the weight of the set associated with it. We have just described the reduction of the set cover problem to the GST problem. Indeed, algorithms for this problem can be used to solve some other related problems [8].

Solving the LP relaxation to the problem is a very common first step in combinatorial optimization algorithms. This is particularly so, if we look at the polylogarithmic approximation algorithm for GST given by Garg et al. [14], where we see that it solves (2) when the input graph is a tree. Afterwards, they extend their result to general graphs using Bartal's technique of tree metrics approximation [2, 9]. Fortunately, solving (2) for optimality is not necessary, but finding $(1+O(\varepsilon))$-approximate solution to (2) for some small $\varepsilon > 0$ is sufficient. We refer to their theorem.

Theorem 1 (Garg et al. [14], page 75, Theorem 4.1.)**.** *There is a randomized polynomial time algorithm that, with a probability at least* $1/2$*, finds a group Steiner tree on an underlying graph which is a tree, at a cost no more than* $O(\log N \log k)$ *times optimal value of FGST in (2), where* N *is the maximum size of the group and* k *is the number of groups.*

Indeed, the polylogarithmic approximation ratio given by Garg et al. gives an upper bound to the integrality gap of relaxation in (2). It is not hard to observe that approximating (2) within a factor $1 + O(\varepsilon)$, for some small $\varepsilon > 0$, does not asymptomatically change the approximation ratio in Theorem 1. It motivates us to find an efficient *fully polynomial time approximation scheme* (FPTAS) for (2). We say that an algorithm for a minimization problem is FPTAS if it returns a solution with cost not exceeding $1 + \varepsilon$ times the cost of the optimal, for arbitrarily small $\varepsilon > 0$, in time that is polynomial in the input size of the problem and $1/\varepsilon$.

2. Previous work

The paper by Garg and Khandekar [11] considers the fractional Steiner Forest Tree and related problems. Their approach leads to an unified framework for finding a minimum hitting set for a collection of *clutters*. A clutter on set X is a family $\mathcal{C}$ of subsets of X such that no set in $\mathcal{C}$ is contained in some other set in $\mathcal{C}$. We say that clutter $\mathcal{C}$ is hit by $T \subset X$ if at least one of its sets is included in T. Their algorithm aims to find a minimum cost subset T of X that hits each clutter in the family of k clutters $\mathcal{C}_1, \ldots, \mathcal{C}_k$ on X. Their approach also uses an oracle $\mathcal{O}_i$, $i \in [k]$, that computes the minimum cost set in clutter C_i and runs in time $T_{\mathcal{O}_i}$. All the problems that they consider can be formulated as the above described *min-hit problem*. They provide an $(1 + \varepsilon)$-approximation of the optimal solution to the min-hit problem in $O((m \log^2 k)/\varepsilon^2 \sum_{i=1}^{k} T_{\mathcal{O}_i})$ time. FGST can be easily interpreted as a min-hit problem for a collection of clutters $\{\mathcal{P}_i\}_{i=1}^{k}$ on the set of edges E. Each clutter $\mathcal{P}_i$,

$i \in [k]$ consists of paths from group G_i, (i.e. a vertex from group G_i) to the root r. We wish to find the minimum cost subset of edges that hits each clutter $\mathcal{P}_i$, $i \in [k]$. In this case, all oracles $\mathcal{O}_i$, $i \in [k]$ are the shortest path algorithms from vertices in G_i to r. Since the root r is fixed, $\mathcal{O}_i$ takes as much time as is taken by a single-source shortest path algorithm. We conclude that the algorithm by Garg and Khandekar in [11] runs in $O(mk(m + n \log n) \log^2 k/\varepsilon^2)$ time[‡]. They pointed out that their algorithm is the only one known for families of clutters that do not satisfy the max-flow min-cut (MFMC) property. According to their approach, the group Steiner tree problem can be viewed as a min-hit problem on a family of clutters that consist of multicommodity paths.

3. FPTAS improvements

Although FGST has a compact flow-based formulation with polynomially many variables and constraints [16], combinatorial algorithms exploiting the special structure of a problem often have better worst-case running time bounds than some known algorithms for general linear programming problems. The linear program (2) exhibits the structure of a fractional covering linear program that is more carefully studied later in this paper. Since (2) has exponentially many constraints, state-of-the-art approaches in [19, 20, 17] cannot be directly applied. Our main task is to adopt their approach and show that it yields a better worst case running time algorithm compared to the previous algorithms, when the number of groups k is large enough.

Garg and Khandekar in [11] pointed out that techniques in [23, 26] cannot be applied to obtain a FPTAS for the min-hit problem on a collection of clutters since it cannot be formulated as a packing or covering problem. First, we want to present that nearly linear-time FPTAS for explicit fractional packing and covering linear programs by Koufogiannakis and Young in [20] is adoptable to an FPTAS for the FGST problem. This adaptation is very similar to the algorithm given by Garg and Könemann in [12, 13] for the maximum multicommodity flow problem and other fractional packing problems. Second, we point that the running time of our algorithm is $O((m \log m)k(m+n^{4/3}\varepsilon^{-16/3}) \log^c(m+n^{4/3}\varepsilon^{-16/3})/\varepsilon^2)$ where c is some constant that comes from running time of $(1 + \varepsilon)$-approximate min-cut algorithm in [7] (Theorem 3). If we neglect constants in running times, finding a (sub)set of solutions to the following inequality:

$$\frac{m \log^2 k}{\varepsilon^2} k(m + n \log n) > \frac{m \log m}{\varepsilon^2} k(m + n^{4/3}\varepsilon^{-16/3}) \log^c(m + n^{4/3}\varepsilon^{-16/3}), \quad (3)$$

gives us values of k that make our algorithm faster than the algorithm in [11]. We now present the following technical facts that simplify the lower bound on values of k which satisfy (3).

Fact 1.

i) For any $\varepsilon \in \langle 0, 1\rangle$, there is a $n_0 \in \mathbb{N}$ such that

$$\frac{m + n^{4/3}\varepsilon^{-16/3}}{m + n \log n} < \frac{n^{1/3}}{\log n}\varepsilon^{-16/3},$$

[‡] $\tilde{O}(f(n))$ denotes $O(f(n) \log^c f(n))$ for some function f and positive constant c

for all $n \geq n_0$ *and* $n - 1 \leq m \leq n(n-1)/2$.

ii) For any constant c *and* $\varepsilon \in \langle 0, 1/6]$ *there is a* $n_0 \in \mathbb{N}$ *such that*

$$\frac{n^{1/3}}{\log n} > \log^c(m + n^{4/3}\varepsilon^{-16/3}),$$

for all $n \geq n_0$ *and* $n - 1 \leq m \leq n(n-1)/2$.

Now, from (3) follows that

$$\log^2 k > \frac{m + n^{4/3}\varepsilon^{-16/3}}{m + n\log n} \log m \log^c(m + n^{4/3}\varepsilon^{-16/3}). \tag{4}$$

Fact (1) implies the following bound on the right-hand side of (4)

$$\frac{m + n^{4/3}\varepsilon^{-16/3}}{m + n\log n} \log m \log^c(m + n^{4/3}\varepsilon^{-16/3}) < \left(\frac{n^{1/3}}{\log n}\right)^2 \varepsilon^{-16/3}, \tag{5}$$

for $n \geq n_0$ and some $n_0 \in \mathbb{N}$. In order to identify certain values of k that satisfy (3), inequalities (4) and (5) suggest that finding k is sufficient such that:

$$\log^2 k > \left(\frac{n^{1/3}}{\log n}\right)^2 \varepsilon^{-16/3}, \tag{6}$$

which gives

$$k > 2^{\left(\frac{n^{1/3}}{\log n}\right)\varepsilon^{-8/3}}, \tag{7}$$

for $n \geq n_0$ and some $n_0 \in \mathbb{N}_0$. Now, we can conclude that our algorithm from Theorem 3 is faster that the algorithm in [11] when $k > 2^{(n^{1/3}/\log n)\varepsilon^{-8/3}}$ and $n \geq n_0$, for some $n_0 \in \mathbb{N}$. Our approach is independent of the approximate minimum cut computation in Algorithm 2. Some newer and faster nearly linear algorithms [25] also can be used, but all incoming improvements to the approximate max-flow min-cut computation will only decrease the lower bound (7). Unfortunately, improvements of approximate min-cut computation cannot significantly improve the running time of Algorithm 1 by decreasing the dominating polynomial factors.

In Section 2 we give an essential overview of the Lagrangean relaxation algorithm given by Koufogiannakis and Young [19, 20] based on the idea of a two players zero-sum game [15] and non-uniform increments [12, 13]. In the spirit of their algorithm, we present an adaptation that involves the calling of an oracle routine that computes a $(1+\varepsilon)$-approximate cut at each iteration. In Section 3, the analysis of the approximation ratio and running time is given.

3.1. Preliminaries

In order to present the main idea of our approach, let us recall the covering and packing linear programs as is considered in [20]. Let $\mathbf{A} \in \mathbb{R}_+^{m\times n}$, $\mathbf{c} \in \mathbb{R}_+^n$ and

$\mathbf{b} \in \mathbb{R}_+^m$ for $m, n \in \mathbb{N}$. The linear program

$$\begin{aligned} &\min \ \mathbf{c}^T\mathbf{x} \\ s.t. \quad &\mathbf{A}\mathbf{x} \geq \mathbf{b}, \\ &\mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{8}$$

is called a **covering linear program**. The dual of (8)

$$\begin{aligned} &\max \ \mathbf{b}^T\mathbf{y} \\ s.t. \quad &\mathbf{A}^T\mathbf{y} \leq \mathbf{c}, \\ &\mathbf{y} \geq \mathbf{0} \end{aligned} \tag{9}$$

is called a **packing linear program**. W.l.o.g we assume that components of vectors **b** and **c** are all ones.

After the transformation of Problem (2), we obtain the following fractional covering program

$$\begin{aligned} \min \quad & \textstyle\sum_{e\in E} x_e \\ s.t. \quad & \textstyle\sum_{e\in\delta(S)} \frac{1}{w(e)} x_e \geq 1, \quad S \in \mathcal{S}_r, \\ & x_e \geq 0, \quad e \in E, \end{aligned} \tag{10}$$

while the dual is the fractional packing linear program

$$\begin{aligned} \max \quad & \textstyle\sum_{S\in\mathcal{S}_r} y_S \\ s.t. \quad & \textstyle\sum_{S\in\mathcal{S}_{r,e}} \frac{1}{w(e)} y_S \leq 1, \quad e \in E, \\ & y_S \geq 0, \quad S \in \mathcal{S}_r, \end{aligned} \tag{11}$$

where $\mathcal{S}_{r,e} = \{S \in \mathcal{S}_r : e \in \delta(S)\}$. Now it is clear that (10) and (11) are representable in the form of (8) and (9), respectively, where matrix $\mathbf{A}$ is given by

$$A_{S,e} = \begin{cases} 1/w(e) & , e \in \delta(S), \\ 0 & , \text{otherwise.} \end{cases} \tag{12}$$

3.2. Algorithm

3.2.1. Description of the algorithm

We present the Lagrangian relaxation algorithm based on ideas described in the papers of Koufogiannakis and Young [19, 20] that iteratively improves the solution of the Primal (10). In order to present the main idea of the algorithm, we recall how to obtain a Lagrangian relaxation of the Dual (11)

$$\max_{\mathbf{A}^T\mathbf{y}\leq\mathbf{1},\mathbf{y}\geq\mathbf{0}} \mathbf{1}^T\mathbf{y} \leq \min_{\mathbf{x}\geq\mathbf{0}} \max_{\mathbf{y}\geq\mathbf{0}} \left(\mathbf{1}^T\mathbf{y} + \mathbf{x}^T(\mathbf{1} - \mathbf{A}^T\mathbf{y})\right). \tag{13}$$

Let $\tilde{y}_e(t)$ be $\sum_{S\in\mathcal{S}_{r,e}} y_S(t)/w(e)$, for $e \in E$. We conclude in (13) that the primal variable x_e plays the role of the penalty for the violation of the dual constraint $\sum_{S\in\mathcal{S}_{r,e}} y_S/w(e) \leq 1$. Let $x_e(t)$, $e \in E$ and $y_S(t)$, $S \in \mathcal{S}_r$ be the primal and dual variables at iteration $t \geq 0$. At the beginning, we have $x_e(0) = 0$, $e \in E$ and $y_S(0) = 0$, $S \in \mathcal{S}_r$. If $\tilde{y}_e(t)$ is "large", and we suppose that the violation of the dual constraint is also "large". According to (13) increasing x_e to mimic the increasing the penalty for a violation of the corresponding dual constraint seems reasonable. In order to quantify the violation, we use $\tilde{y}_e(t)$. Following the idea of exponential potential function methods in [3], we increase each variable $x_e(t)$ by a quantity that is exponential in violation of $\tilde{y}_e(t-1)$ and divided by the sum of exponentials in all dual constraint violations. More precisely, our algorithm iteratively improves the primal solution $x_e(t)$ quantitatively which is proportional to $p_e(t)/|\mathbf{p}(t)|$ where

$$p_e(t) = (1+\varepsilon)^{\tilde{y}_e(t-1)}. \tag{14}$$

Instead of random sampling of the primal and dual variables, we increase deterministically one variable in the dual and all variables in the primal. At each iteration in line 13 of Algorithm 1, we increment the dual variable $y_{\hat{S}(t)}$ by $\hat{W}(t)$, where $(\hat{S}(t), V \setminus \hat{S}(t))$ is the $(1+\varepsilon)$-approximate minimum cut with respect to the capacities

$$c_e(t) := \frac{1}{w(e)} \frac{p_e(t)}{|\mathbf{p}(t)|}, \qquad \text{for } e \in E,$$

in graph G that separates some group G_i from r, and $\hat{W}(t) := \min_{e\in\delta(\hat{S}(t))} w(e)$. More details for determining the cut $(\hat{S}(t), V \setminus \hat{S}(t))$ are found in Section 3.2.2.

We also note, at line 11 of Algorithm 1, that $\hat{W}(t)$ is the length of the step that increases the primal variable x_e in the direction $p_e(t)/|\mathbf{p}(t)|$, for $e \in E$. In other words, $\hat{W}(t)$ is distributed among all primal variables according to the weight vector $p(t)/|\mathbf{p}(t)|$. We also update $\tilde{y}_e(t)$ and $p_e(t)$ only for $e \in \delta(\hat{S}(t))$ at each iteration. Our algorithm terminates when $M(t) \geq T$ where

$$M(t) := \max_{e\in E} \tilde{y}_e(t). \tag{15}$$

Scaling the primal solution $\mathbf{x}$ by the capacity of the $(1+\varepsilon)$-approximate minimum cut gives a feasible primal solution.

3.2.2. $(1+\varepsilon)$-approximate minimum cut oracle

In this section we describe the algorithm that computes a $(1+\varepsilon)$-approximate minimum cut $(\hat{S}(t), V \setminus \hat{S}(t))$ that separates some group G_i from r with respect to the capacities $c_e(t)$, $e \in E$, at iteration $t \geq 0$. We solve that problem as a sequence of k $(1+\varepsilon)$-approximate minimum cut problems in the following group Steiner network.

Definition 1. *Let $(G, \mathcal{G}, w)$ be a group Steiner tree instance and $i \in [k]$. A group Steiner network with respect to the capacities $\mathbf{c} \in \mathbb{R}_+^{|E|}$ is the network $(\vec{G}_i, \mathbf{c})$ where $\vec{G}_i$ is a directed graph obtained from G as follows:*

Algorithm 1 FPTAS for the fractional group Steiner tree problem

```
1: $x_e(0) := 0$, $\tilde{y}_e(0) = 0$ for all $e \in E$, $y_S(0) := 0$, $S \in \mathcal{S}_r$,$M(0) \leftarrow 0$, $t := 0$,
   $T := \frac{\ln m}{\varepsilon^2}$
2: $p_e(1) := 1$, $e \in E$
3: while $M(t) < T$ do
4:     $t \leftarrow t + 1$
5:     for $e \in E$ do
6:         $c_e(t) \leftarrow \frac{1}{w(e)} \frac{p_e(t)}{|\mathbf{p}(t)|}$
7:     end for
8:     calculate $\hat{S}(t)$ with respect to the capacities $\mathbf{c}(t)$ using Algorithm 2
9:     $\hat{W}(t) \leftarrow \min_{e \in \delta(\hat{S}(t))} w(e)$
10:    for $e \in E$ do
11:        $x_e(t) \leftarrow x_e(t-1) + \hat{W}(t) \cdot \frac{p_e(t)}{|\mathbf{p}(t)|}$
12:    end for
13:    $y_{\hat{S}(t)}(t) \leftarrow y_{\hat{S}(t)}(t-1) + \hat{W}(t)$
14:    $M(t) \leftarrow M(t-1)$
15:    for $e \in E$ do
16:        if $e \in \delta(\hat{S}(t))$ then
17:            $\tilde{y}_e(t) \leftarrow \tilde{y}_e(t-1) + \frac{\hat{W}(t)}{w(e)}$
18:            $p_e(t+1) \leftarrow p_e(t)(1+\varepsilon)^{\frac{\hat{W}(t)}{w(e)}}$
19:            if $\tilde{y}_e(t) > M(t)$ then
20:                $M(t) \leftarrow \tilde{y}_e(t)$
21:            end if
22:        else
23:            $\tilde{y}_e(t) \leftarrow \tilde{y}_e(t-1)$
24:            $p_e(t+1) \leftarrow p_e(t)$
25:        end if
26:    end for
27: end while
28: calculate $S'$ with respect to the capacities $\mathbf{x}(t)$ using Algorithm 2
29: $m(t) \leftarrow \sum_{e \in \delta(S')} x_e(t)$
30: return $\mathbf{x}(t)/m(t)$
```

- *for each edge $e = \{u, v\} \in E$, we introduce two directed edges (u, v) and (v, u) where each of them has the capacity c_e,*
- *for group $G_i \in \mathcal{G}$, we introduce a new vertex g_i,*
- *for each $v \in G_i$, we introduce a directed edge (g_i, v) with infinite capacity.*

At each iteration l we compute a $(1+\varepsilon)$-approximate minimum cut that separates group G_l and the root r in the group Steiner network $(\vec{G}_l, \mathbf{c})$ by using the algorithm of Christiano et al. in [7]. Upon completing we take the one with the minimum capacity. It is easy to see that we computed a $(1+\varepsilon)$-approximate cut that separates group G_j from r, for some $j \in [k]$. The running time of Algorithm 2 is given in the following Proposition.

Proposition 1. *Algorithm 2 returns $\hat{S} \subseteq V \setminus \{r\}$, where $(\hat{S}, V \setminus \hat{S})$ is $(1+\varepsilon)$-approximate min-cut that separates some group G_i from r, in $\tilde{O}(k(m + n^{4/3}\varepsilon^{-16/3}))$ time.*

Proof. The construction of the group Steiner network in Definition 1 does not asymptotically enlarge the size of the network because only one vertex is added and at most $2m+n$ directed edges are introduced[§]. Therefore each computation of $(1+\varepsilon)$-approximate min-cut takes at most $\tilde{O}(m + n^{4/3}\varepsilon^{-16/3})$ time. Since there are k such computations, the statement follows easily. □

Algorithm 2 $(1+\varepsilon)$-approximate min-cut oracle

```
**Input:** the capacities $\mathbf{c}$, $\varepsilon > 0$
**Output:** $(1+\varepsilon)$-approximate min-cut $\hat{S}$ that separates $g_i$ from $r$ in the $(\vec{G}_i, \mathbf{c})$ for some $i \in [k]$
$\hat{C}, \hat{C}' \leftarrow \infty$
$\hat{S}, \hat{S}' \leftarrow \emptyset$
**for** $l = 1, \ldots, k$ **do**
    compute a $(1+\varepsilon)$-approximate $(g_l, r)$ min-cut $\hat{S}'$ in the network $(\vec{G}_l, \mathbf{c})$ by using algorithm in [7]
    $\hat{C}' \leftarrow \sum_{e \in \delta(\hat{S}')} c(e)$
    **if** $\hat{C}' < \hat{C}$ **then**
        $\hat{S} \leftarrow \hat{S}'$
        $\hat{C} \leftarrow \hat{C}'$
    **end if**
**end for**
**return** $\hat{S}$
```

4. Analysis of the Algorithm 1

In this section we prove that Algorithm 1 returns a feasible solution to (2) with a total cost not exceeding $1 + O(\varepsilon)$ times the optimal. We will also prove that running time is indeed that which was previously mentioned in Section 3.

4.1. Approximation ratio analysis

First, we give some technical facts (without proof) that are necessary for an analysis.

Claim 1.

i) $(1+\varepsilon)^x \leq 1 + \varepsilon x$, *for all* $0 < \varepsilon < 1$ *and* $0 \leq x \leq 1$

ii) $(1+\varepsilon x) \leq e^{\varepsilon x}$, *for all* $\varepsilon, x \in \mathbb{R}$

In order to prove that Algorithm 1 returns the $(1+O(\varepsilon))$-approximate solution to (2), we present some useful facts.

[§]since each group contains at most n vertices

Proposition 2. *For all $t \geq 1$ Algorithm 1 maintains the following invariants:*

i) $|\mathbf{x}(t)| = |\mathbf{y}(t)|$, $t \geq 0$

ii) $\max_{e \in E}(\tilde{y}_e(t) - \tilde{y}_e(t-1)) = 1$

Proof.
i) This statement follows easily from the facts that only one dual variable corresponding to $\hat{S}(t)$ is increased by $\hat{W}(t)$ and that the sum of all increments in primal variables equals $\hat{W}(t)$.

ii)

$$\begin{aligned}
\max_{e \in E}(\tilde{y}_e(t) - \tilde{y}_e(t-1)) &= \max_{e \in E} \sum_{S \in \mathcal{S}_{r,e}} \frac{1}{w(e)}(y_S(t) - y_S(t-1)) \\
&= \max_{e \in \delta(\hat{S}(t))} \frac{1}{w(e)}(y_{\hat{S}(t)}(t) - y_{\hat{S}(t)}(t-1)) \\
&= \max_{e \in \delta(\hat{S}(t))} \frac{\hat{W}(t)}{w(e)} = \frac{\hat{W}(t)}{\min_{e \in \delta(\hat{S}(t))} w(e)} \\
&= 1
\end{aligned}$$

□

The crucial fact in the analysis of the approximation ratio of Algorithm 1 is the upper bound for the value of $\Phi(t+1)$, where Φ is the potential function given by

$$\Phi(t) = |\mathbf{p}(t)|.$$

This approach is widely used in designing fast combinatorial approximation algorithms for similar problems. A very nice overview of potential function methods can be found in [3].

Lemma 1. *For any $t \geq 1$*

$$\Phi(t+1) \leq m \exp\left(\varepsilon \sum_{t'=1}^{t} \hat{W}(t')\hat{C}(t')\right),$$

where $\hat{C}(t)$ is the capacity of a $(1+\varepsilon)$-approximate minimum cut $(\hat{S}(t), V \setminus \hat{S}(t))$.

Proof.

$$\begin{aligned}
\Phi(t+1) &= \sum_{e \in E} p_e(t+1) = \sum_{e \in E} (1+\varepsilon)^{\tilde{y}_e(t)} \\
&= \sum_{e \in \delta(\hat{S}(t))} (1+\varepsilon)^{\tilde{y}_e(t-1) + \frac{\hat{W}(t)}{w(e)}} + \sum_{e \notin \delta(\hat{S}(t))} (1+\varepsilon)^{\tilde{y}_e(t-1)} \\
&= \sum_{e \in \delta(\hat{S}(t))} p_e(t)(1+\varepsilon)^{\frac{\hat{W}(t)}{w(e)}} + \sum_{e \notin \delta(\hat{S}(t))} p_e(t) \qquad (16)
\end{aligned}$$

From Fact *ii)* in Proposition 2, Claim 1 and (16) it follows

$$\begin{aligned}
\Phi(t+1) &\le \sum_{e\in\delta(\hat{S}(t))} p_e(t)\left(1+\frac{\hat{W}(t)}{w(e)}\varepsilon\right) + \sum_{e\notin\delta(\hat{S}(t))} p_e(t) \\
&= |\mathbf{p}(t)|\left(1+\varepsilon\sum_{e\in\delta(\hat{S}(t))}\frac{p_e(t)}{|\mathbf{p}(t)|}\cdot\frac{\hat{W}(t)}{w(e)}\right) \\
&= \Phi(t)\left(1+\varepsilon\sum_{e\in\delta(\hat{S}(t))} c_e(t)\hat{W}(t)\right) \\
&\le \Phi(t)\exp\left(\varepsilon\sum_{e\in\delta(\hat{S}(t))} c_e(t)\hat{W}(t)\right) \qquad (17)
\end{aligned}$$

Iterating (17), we obtain the statement of the lemma

$$\begin{aligned}
\Phi(t+1) &\le \Phi(1)\exp\left(\varepsilon\sum_{t'=1}^{t}\sum_{e\in\delta(\hat{S}(t'))} c_e(t')\hat{W}(t')\right) \\
&= m\exp\left(\varepsilon\sum_{t'=1}^{t}\sum_{e\in\delta(\hat{S}(t'))} c_e(t')\hat{W}(t')\right) \\
&= m\exp\left(\varepsilon\sum_{t'=1}^{t}\hat{W}(t')\sum_{e\in\delta(\hat{S}(t'))} c_e(t')\right) \\
&= m\exp\left(\varepsilon\sum_{t'=1}^{t}\hat{W}(t')\hat{C}(t')\right).
\end{aligned}$$

□

Lemma 2. *Let* $m(t)$ *be* $\min_{S\in\mathcal{S}_r}\sum_{e\in\delta(S)}\frac{x_e(t)}{w(e)}$, *for* $t\ge 1$, *then*

$$\sum_{t'=1}^{t}\hat{C}(t')\hat{W}(t') \le (1+\varepsilon)m(t).$$

Proof. Let us observe from line 11 of Algorithm 1 that

$$x_e(t) = \sum_{t'=1}^{t}\hat{W}(t')\frac{p_e(t')}{|\mathbf{p}(t')|}. \qquad (18)$$

For any arbitrary $S\in\mathcal{S}_r$ using (18) we have

$$\begin{aligned}\sum_{e\in\delta(S)}\frac{x_e(t)}{w(e)} &= \sum_{e\in\delta(S)}\frac{1}{w(e)}\sum_{t'=1}^{t}\hat{W}(t')\frac{p_e(t')}{|\mathbf{p}(t')|}\\ &= \sum_{t'=1}^{t}\hat{W}(t')\sum_{e\in\delta(S)}\frac{1}{w(e)}\frac{p_e(t')}{|\mathbf{p}(t')|}\\ &= \sum_{t'=1}^{t}\hat{W}(t')\sum_{e\in\delta(S)}c_e(t'), \qquad (19)\end{aligned}$$

where $c_e(t')$ is given in line 6 of Algorithm 1. Since a $(1+\varepsilon)$-approximate minimum cut $(\hat{S}(t'), V\setminus\hat{S}(t'))$ is calculated at each iteration t' with respect to the capacities $c_e(t')$, $e\in E$, from (19) it is obvious that

$$(1+\varepsilon)\sum_{e\in\delta(S)}\frac{x_e(t)}{w(e)} \geq \sum_{t'=1}^{t}\hat{W}(t')\hat{C}(t'),$$

which proves the statement of the lemma. □

The following theorem gives the upper bound on the approximation ratio of Algorithm 1.

Theorem 2. *After termination, for any* $\varepsilon\in\langle 0,1/6]$*, Algorithm 1 returns a* $(1+6\varepsilon)$*-approximate solution to the LP relaxation of the group Steiner tree (10).*

Proof. Statements of Lemmas 1 and 2 imply

$$(1+\varepsilon)^{\tilde{y}_e(t)} \leq m\exp(\varepsilon(1+\varepsilon)m(t)), \quad \forall e\in E,$$

which, after taking the natural logarithm of both sides, becomes

$$\tilde{y}_e(t)\ln(1+\varepsilon) \leq \ln m + \varepsilon(1+\varepsilon)m(t), \quad \forall e\in E. \qquad (20)$$

Since inequality in (20) is valid for all $e\in E$, it follows that

$$M(t)\ln(1+\varepsilon) \leq \ln m + \varepsilon(1+\varepsilon)m(t), \quad \forall e\in E. \qquad (21)$$

Our algorithm terminates when inequality $M(t)\geq T$ is valid. If we use this fact in Inequality (21), we obtain

$$\frac{m(t)}{M(t)} \geq 1-3\varepsilon,$$

which, for any $\varepsilon\in\langle 0,1/6]$, gives

$$\frac{M(t)}{m(t)} \leq 1+6\varepsilon. \qquad (22)$$

Let $\hat{\mathbf{x}}$ be the vector returned by Algorithm 1 and $\mathbf{x}^*$ the optimal solution to (10). From Fact *i)* in Proposition 2, the strong duality theorem and (22), it follows that

$$\frac{|\hat{\mathbf{x}}|}{|\mathbf{x}^*|} \leq \frac{\frac{|\mathbf{x}|}{m(t)}}{\frac{|\mathbf{y}|}{M(t)}} = \frac{M(t)}{m(t)} \leq 1 + 6\varepsilon.$$

□

4.2. Running time analysis

In this section, we give an analysis of the running time of Algorithm 1.

Theorem 3. *Algorithm 1 returns a* $(1+6\varepsilon)$*-approximate solution to (10) in* $\tilde{O}(mk(m+n^{4/3}\varepsilon^{-16/3})/\varepsilon^2)$ *time.*

Proof. The update operations inside the while loop (lines 3-27), excluding line 8, in Algorithm 1 take $O(m)$ time. From Proposition 1 it follows that Algorithm 1 takes $\tilde{O}(k(m+n^{4/3}\varepsilon^{-16/3}))$ time per iteration since Algorithm 2 runs inside the while loop at line 8 and it dominates the time of all update operations. It remains to show that there are at most $(m \ln m)/\varepsilon^2$ iterations until termination. There are at most m iterations until $M(t)$ is increased by 1 because at least one edge is a maximizer of the left-hand side term in *ii*) of Proposition 2. It follows that $M(t)$ is at least T after at most mT iterations of the while loop. □

5. Conclusion

We presented a simple modification of the fully polynomial time approximation scheme (FPTAS) in [19, 20] for explicit fractional packing and covering linear programs that is very close to the approach in [12, 13]. In this way, we obtained Algorithm 1 that computes a $(1+\varepsilon)$-approximate solution to the fractional group Steiner tree problem whose running time outperforms the algorithm in [11] when the number of groups k is large enough. Since all algorithms in [11] for the min-hit problem on a family of clutters that do not satisfy the max-flow min-cut property are reported as the only known algorithms, further research in this area can be undertaken to improve the running times of these algorithms using an approach similar to the one presented in this paper.

On the other hand, the FPTAS for the fractional group Steiner tree (FGST) problem, presented in this paper, can be applied for approximate solving of large-scale group Steiner tree (GST) instances that naturally arise from the various problems that are of great interest in Operations Research community. The FGST can be found as the part of approximation algorithms for GST problem [14]. Furthermore, the solution to the FGST problem can be used as a good lower bound in the optimal solution to the GST problem in some general linear integer programming techniques. Besides being a very common application for the problem associated with wire routing and multiport terminals in physical VLSI design [24], the GST problem can be applied to find teams of experts in social networks that are supposed to have solved some of the specific tasks [21, 1, 22]. This application requires extremely fast algorithms, since social networks grows rapidly. This fact provides motivation for a further improvements of running time for the FPTAS and FGST problems.

Acknowledgements

The author wishes to thank Domagoj Matijević for useful discussion.

References

[1] Anagnostopoulos, A., Becchetti, L., Castillo, C., Gionis, A. and Leonardi, S. (2012). Online team formation in social networks. In: Proceedings of the 21st international conference on World Wide Web - WWW '12 (pp. 839–848). New York, USA. ACM Press.

[2] Bartal, Y. (1998). On approximating arbitrary metrices by tree metrics. In: Proceedings of the thirtieth annual ACM symposium on Theory of computing - STOC '98 (pp. 161–168). New York, USA. ACM Press.

[3] Bienstock, D. (2002). Potential Function Methods for Approximately Solving Linear Programming Problems: Theory and Practice. Springer.

[4] Byrka, J., Grandoni, F., Rothvoß, T. and Sanità, L. (2010). An improved LP-based approximation for Steiner tree. In: STOC '10 Proceedings of the 42nd ACM symposium on Theory of computing (pp. 583–592). New York, USA. ACM Press.

[5] Byrka, J., Grandoni, F., Rothvoss, T. and Sanità, L. (2013). Steiner tree approximation via iterative randomized rounding. Journal of the ACM, 60(1), 1–33. doi:10.1145/2432622.2432628.

[6] Chlebík, M. and Chlebíková, J. (2008). The Steiner tree problem on graphs: Inapproximability results. Theoretical Computer Science, 406(3), 207–214. doi:10.1016/j.tcs.2008.06.046.

[7] Christiano, P., Kelner, J. A., Madry, A., Spielman, D. A. and Teng, S. H. (2011). Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In: Proceedings of the 43rd annual ACM symposium on Theory of computing - STOC '11 (pp. 273–282). New York, USA. ACM Press.

[8] Elbassioni, K., Jelić, S. and Matijević, D. (2012). The relation of connected set cover and group Steiner tree. Theoretical Computer Science, 438, 96–101. doi:10.1016/j.tcs.2012.02.035.

[9] Fakcharoenphol, J., Rao, S. and Talwar, K. (2004). A tight bound on approximating arbitrary metrics by tree metrics. Journal of Computer and System Sciences, 69(3), 485–497. doi:10.1016/j.jcss.2004.04.011.

[10] Feige, U. (1998). A threshold of ln n for approximating set cover. Journal of the ACM, 45(4), 634–652. doi:10.1145/285055.285059.

[11] Garg, N. and Khandekar, R. (2002). Fast approximation algorithms for fractional Steiner forest and related problems. In: FOCS '02 Proceedings of the 43rd Symposium on Foundations of Computer Science (pp. 500–509). IEEE Computer Society.

[12] Garg, N. and Konemann, J. (1998). Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In: Proceedings 39th Annual Symposium on Foundations of Computer Science - FOCS'98. (pp. 300–309). Palo Alto, CA, USA. IEEE Computer Society.

[13] Garg, N. and Könemann, J. (2007). Faster and simpler algorithms for multicommodity flow and other fractional packing problems. SIAM Journal on Computing, 37(2), 630–652. doi:10.1137/s0097539704446232.

[14] Garg, N., Konjevod, G. and Ravi, R. (2000). A polylogarithmic approximation algorithm for the group Steiner tree problem. Journal of Algorithms, 37(1), 66–84. doi:10.1006/jagm.2000.1096.

[15] Grigoriadis, M. D. and Khachiyan, L. G. (1995). A sublinear-time randomized approximation algorithm for matrix games. Operations Research Letters, 18(2), 53–58. doi:10.1016/0167-6377(95)00032-0.

[16] Halperin, E., Kortsarz, G., Krauthgamer, R., Srinivasan, A. and Wang, N. (2007). Integrality ratio for group Steiner trees and directed Steiner trees. SIAM Journal on Computing, 36(5), 1494–1511. doi:10.1137/s0097539704445718.

[17] Jelić, S., Laue, S., Matijević, D. and Wijerama, P. (2015). A fast parallel implementation of a PTAS for fractional packing and covering linear programs. International Journal of Parallel Programming, 43(5), 840–875. doi:10.1007/s10766-015-0352-y.

[18] Karp, R. M. (1972). Reducibility among combinatorial problems. In: Miller, R. E., Thatcher, J. W., and Bohlinger, J. D., editors, Complexity of Computer Computations (pp. 85–103). Springer US.

[19] Koufogiannakis, C. and Young, N. E. (2007). Beating simplex for fractional packing and covering linear programs. In: FOCS'07 Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (pp. 494–504). IEEE.

[20] Koufogiannakis, C. and Young, N. E. (2013). A nearly linear-time PTAS for explicit fractional packing and covering linear programs. Algorithmica, 70(2), 648–674. doi:10.1007/s00453-013-9771-6.

[21] Lappas, T., Liu, K. and Terzi, E. (2009). Finding a team of experts in social networks. In: KDD '09 Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 467–176). New York, USA. ACM Press.

[22] Li, C. T., Shan, M. K. and Lin, S. D. (2015). On team formation with expertise query in collaborative social networks. Knowledge and Information Systems, 42(2), 441–463. dio:10.1007/s10115-013-0695-x.

[23] Plotkin, S. A., Shmoys, D. B. and Tardos, E. (1995). Fast approximation algorithms for fractional packing and covering problems. Mathematics of Operations Research, 20(2), 257–301. doi:10.1287/moor.20.2.257.

[24] Reich, G. and Widmayer, P. (1990). Beyond Steiner's problem: A VLSI oriented generalization. In: Proceedings of the Fifteenth International Workshop on Graph-theoretic Concepts in Computer Science, WG '89 (pp. 196–210). Springer-Verlag, New York, Inc.

[25] Sherman, J. (2013). Nearly maximum flows in nearly linear time. In: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, volume 2013 (pp. 263–269). IEEE Computer Society.

[26] Young, N. (2001). Sequential and parallel algorithms for mixed packing and covering. In: Proceedings 2001 IEEE International Conference on Cluster Computing (pp. 538–546). IEEE Computer Society.