Theoretical Computer Science 410 (2009) 1093–1098

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

An approximation algorithm to the k-Steiner Forest problem

Peng Zhang [a,*], Mingji Xia [b]

[a] School of Computer Science and Technology, Shandong University, Jinan 250101, China

[b] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, P.O. Box 8718, Beijing, 100080, China

ARTICLE INFO

Keywords:
k-Steiner Forest
Greedy
Set k-cover
Approximation algorithm

ABSTRACT

Given a graph G, an integer k, and a demand set $D = \{(s_1, t_1), \ldots, (s_l, t_l)\}$, the k-Steiner Forest problem finds a forest in graph G to connect at least k demands in D such that the cost of the forest is minimized. This problem was proposed by Hajiaghayi and Jain in SODA'06. Thereafter, using a Lagrangian relaxation technique, Segev et al. gave the first approximation algorithm to this problem in ESA'06, with performance ratio $O(n^{2/3} \log l)$. We give a simpler and faster approximation algorithm to this problem with performance ratio $O(n^{2/3} \log k)$ via greedy approach, improving the previously best known ratio in the literature.

1. Introduction

Given a graph $G = (V, E)$ with costs on edges, an integer $k > 0$, and a demand set $D = \{(s_1, t_1), \ldots, (s_l, t_l)\}$, where each demand (a.k.a. source–sink pair) is distinct but vertices in different demands are not required to be distinct, the k-Steiner Forest (k-Forest for short) problem asks to find a forest in graph G to connect at least k demands in D, such that the cost of the forest is minimized. k-Forest was proposed by Hajiaghayi and Jain when they studied the Prize-collecting Generalized Steiner Tree problem in [9].

If we remove the given parameter k in k-Forest, then we get the classical Steiner Forest problem. The currently best approximation ratio for Steiner Forest is $2(1 - 1/l)$, due to Agrawal, Klein and Ravi [1] and Goemans and Williamson [6]. On the other hand, k-Forest also captures the classical rooted k-MST and rooted k-Steiner Tree problems. If in k-Forest we set $D = \{(r, v) : \forall v \in V - \{r\}\}$ where r is the given vertex used as the root, then k-Forest reduces to the rooted k-MST problem. Similarly, if in k-Forest we set $D = \{(r, v) : \forall v \in R - \{r\}\}$ where R is the set of vertices required to be connected and r is the given vertex used as the root, then k-Forest reduces to the rooted k-Steiner Tree problem. Garg [5] showed that in fact the unrooted k-MST problem is equivalent to the rooted k-MST problem, and gave a 2-approximation algorithm to k-MST, which is also the currently best known approximation to k-MST. On the other hand, both k-MST and k-Steiner Tree have been studied using the Lagrangian relaxation technique by Chudak, Roughgarden and Williamson [3].

Although the basic Steiner Forest problem [1,6] and the Prize-collecting Steiner Forest problem [9] (which is called the Prize-collecting Generalized Steiner Tree problem therein) can be well approximated, it is difficult to approximate k-Forest. Hajiaghayi and Jain [9] proved that if k-Forest can be approximated within α, then the Densest k-Subgraph problem can be approximated within α^2. On the other hand, the best known performance ratio for Densest k-Subgraph is $O(n^{1/3-\epsilon})$ for some small $\epsilon > 0$ and the improvement is known to be difficult [4,10]. It is pointed out in [9] that obtaining performance ratio better than $O(n^{1/6-\epsilon})$ for k-Forest requires substantially new ideas.

Segev et al. [12] gave the first non-trivial approximation algorithm to k-Forest, with performance ratio $O(n^{2/3} \log l)$. Their approach is the Lagrangian relaxation technique. In fact, since the seminar work of Jain and Vazirani [8] for the Facility

* Corresponding author.
E-mail addresses: algzhang@gmail.com (P. Zhang), xmjljx@gmail.com (M. Xia).

0304-3975/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2008.10.033

Location and the k-Median problems, in which the Lagrangian relaxation technique was introduced to the design and analysis of approximation algorithm at the first time, the study about approximating the prize-collecting version and k-version of many optimization problems has attracted more attention (see, for instance, [3,8,11]).

1.1. Our results and techniques

We give a simple greedy approximation algorithm to this problem with performance ratio $O(n^{2/3}\log k)$, improving the previously best known ratio $O(n^{2/3}\log l)$ [12]. The optimal solution to k-Forest consists of several disjoint trees, and thus is a forest. Intuitively, a "good" tree in a solution to k-Forest connects some demands by as small as possible cost. In other words, to find a cost-effective tree connecting some demands is a core problem to k-Forest. The cost-effectiveness of a tree in [12] (which is referred to as the density of a tree therein) is defined as the ratio of the cost of the tree and the number of demands connected by this tree. Segev et al. gave a constructive method to find a tree with cost-effectiveness not much larger than that of the most cost-effective tree, and the cost of the found tree not exceeding Δ much more, where Δ is a budget guessed by means of exhaustive searching such that $OPT \le \Delta \le 2 \cdot OPT$, giving OPT as the optimum of the instance of k-Forest. Then they gave a greedy prize-collecting algorithm for the Prize-collecting Steiner Forest problem. Since the prize-collecting algorithm possesses the so-called Lagrangian Multiplier Preserving [7] property, k-Forest can be solved by reducing to Prize-collecting Steiner Forest in the framework of Lagrangian relaxation.

We consider k-Forest in the framework of Set k-Cover, which is the k-version of the Set Cover problem. Formally, Set k-Cover finds a sub-family of the given set family $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ to cover at least k elements in the given grounding set $U = \{e_1, e_2, \ldots, e_n\}$. Slavík [13] gave a greedy approximation algorithm for Set k-Cover with performance ratio H_k, where $H_k = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{k}$ is the k-th harmonic number. The greedy algorithm for Set k-Cover finds the most cost-effective set in each iteration to cover some elements still remaining in U, until at least k elements are covered. In [13] the cost-effectiveness of a set S is carefully defined as the ratio of the cost of S and the number of elements *validly* covered by S. Given k' being the number of elements that need to be covered currently, the number of elements validly covered by S is k' if S covers more than k' new elements, and is the number of new elements covered by S otherwise.

We first prove a coupling lemma which shows that the greedy algorithm coupled with an α-approximation algorithm for finding a cost-effective covering set gives an αH_k-approximation to the Set k-Cover-like problem. This is very useful for the problems that can be cast in the framework of Set k-Cover but it is not known how to find the most cost-effective covering set in polynomial time. One can see that k-Forest can be cast in the framework of Set k-Cover. We define the cost-effectiveness of a covering tree T as the ratio of the cost of T and the numbers validly covered by T, the same as that in [13]. Our newly defined cost-effectiveness of covering tree is different from that in [12]. Then, by implanting the newly defined cost-effectiveness into the constructive method proposed in [12], we give a polynomial time α-approximation algorithm to find a cost-effective covering tree. This eventually leads to an αH_k-approximation algorithm to k-Forest, where $\alpha = O(n^{2/3})$. Our global approach is greedy, which is completely different from that (the Lagrangian relaxation technique) of [12]. Although our algorithm relies on the constructive method proposed therein, we would like to point out that the coupling lemma and the constructive method armed with our cost-effectiveness of covering tree are the key to the success of our algorithm. Our algorithm is of two nested loops and is simpler and faster, while the algorithm in [12] has three nested loops. More importantly, our method gives an improved performance ratio to the problem k-Forest. We believe that our method is of independent interest and may find more applications.

2. The coupling greedy algorithm for set k-cover

Slavík [13] proposed a greedy algorithm for the Set k-Cover problem, which is denoted by algorithm $\mathcal{A}$ in our setting. Given the grounding set $U = \{e_1, e_2, \ldots, e_n\}$ and set family $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ with each set $S \in \mathcal{S}$ having cost $c(S)$, algorithm $\mathcal{A}$ repeatedly finds the most cost-effective set S_j in each iteration j, until the found sets cover at least k elements in U. Denote by C_j the set of covered elements at the beginning of iteration j. For a set $S \in \mathcal{S}$, define $\text{new}(S) = |S \setminus C_j|$, that is, $\text{new}(S)$ is the number of elements newly covered by set S. Suppose that at the beginning of iteration j the number of elements needs to be covered is k'. Then the cost-effectiveness of S is defined as $\text{cost-ef}_v(S) = \frac{c(S)}{\text{val}(S)}$, where $\text{val}(S) = \min\{k', \text{new}(S)\}$ is the number of elements validly covered by S. Note that in the traditional greedy algorithm (see [14], page 16) for Set Cover, the cost-effectiveness of a set S is defined as $\text{cost-ef}_n(S) = \frac{c(S)}{\text{new}(S)}$. We remind the readers that the function $\text{new}(\cdot)$ and $\text{val}(\cdot)$ are defined with respect to the set of elements currently covered by $\mathcal{A}$.

If the most cost-effective set S can be found in polynomial time, then algorithm $\mathcal{A}$ gives an H_k-approximation for Set k-Cover. There are many problems which can be cast in the framework of Set k-Cover. But for these problems the most cost-effective set cannot be found (or it is not known how to find it) in polynomial time, usually due to that the family of sets is of exponential size. This motivates us to couple the greedy algorithm $\mathcal{A}$ with an approximation algorithm to find the most cost-effective covering set.

Lemma 1 (*The Coupling Lemma*). *If the greedy algorithm $\mathcal{A}$ for Set k-Cover is coupled with an algorithm $\mathcal{H}$, where $\mathcal{H}$ in polynomial time finds an α-approximation to the most cost-effective covering set S_j^* in each iteration j of $\mathcal{A}$, then $\mathcal{A}$ outputs an αH_k-approximation to the problem Set k-Cover in polynomial time.*

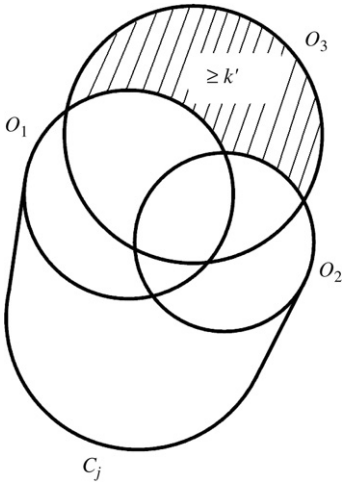

Fig. 1. An example illustrating the sum of $\text{val}(O_u)$ is greater than or equal to k'.

Proof. Without loss of generality, suppose that by reordering the elements in U, the elements covered by algorithm $\mathcal{A}$ in the executing process of $\mathcal{A}$ are $e_1, e_2, \ldots, e_{\hat{k}}$, breaking ties arbitrarily. Notice that $\hat{k} \geq k$. Before giving the detailed analysis, we first introduce some notations. Denote by OPT the optimum of Set k-Cover, and let $O = \{O_1, \ldots, O_r\}$ be an optimal solution. Denote by A_j the set found by algorithm $\mathcal{H}$ in the j-th iteration of $\mathcal{A}$. For each element e_i covered by A_j, define $\text{price}(i) = \frac{c(A_j)}{\text{val}(A_j)}$. Then we know that $A(I) = \sum_j c(A_j) = \sum_{i=1}^{k} \text{price}(i)$, where $A(I)$ is the value of the approximate solution found by $\mathcal{A}$.

We argue that for every $1 \leq i \leq k$, $\text{price}(i) \leq \frac{\alpha}{k-i+1} OPT$. This will give the performance ratio αH_k for the coupled greedy algorithm $\mathcal{A}$. Recall that k' is the number of elements that need to be covered at the beginning of iteration j. For clarity, let $a_j = \text{val}(A_j)$. Since the prices are the same for all elements $e_{k-k'+1}, \ldots, e_{k-k'+a_j}$, and $\frac{1}{k'} \leq \frac{1}{k-i+1}$ for each i ranging from $k - k' + 1$ to $k - k' + a_j$, we only need to show that $\text{price}(i) \leq \frac{\alpha}{k'} OPT$, where $i = k - k' + 1$ is the "first" element covered by A_j. We already have that $\text{price}(i) = \frac{c(A_j)}{\text{val}(A_j)} \leq \alpha \cdot \frac{c(S_j^*)}{\text{val}(S_j^*)}$ because A_j is an α-approximation to S_j^*. By the optimality of S_j^*, $\frac{c(S_j^*)}{\text{val}(S_j^*)} \leq \frac{c(O_u)}{\text{val}(O_u)}$ for each set $O_u \in O$ satisfying that $\text{val}(O_u) \neq 0$. So we get

$$\frac{c(S_j^*)}{\text{val}(S_j^*)} \leq \frac{\sum_{O_u \in O:\ \text{val}(O_u) \neq 0} c(O_u)}{\sum_{O_u \in O:\ \text{val}(O_u) \neq 0} \text{val}(O_u)} \leq \frac{OPT}{k'},$$

where the last inequality holds since $\sum_{O_u \in O:\ c(O_u) \neq 0} \text{val}(O_u) \leq OPT$, and at the beginning of iteration j, there are at least k' elements in $(\bigcup_{O_u \in O} O_u) \setminus C_j$ (see Fig. 1), resulting in $\sum_{O_u \in O:\ \text{val}(O_u) \neq 0} \text{val}(O_u) \geq k'$. The proof is completed. □

We give two remarks about the coupling lemma. First, the function $\text{val}(\cdot)$ plays an important role in the proof of the coupling lemma, since we define $\text{price}(i) = \frac{c(A_j)}{\text{val}(A_j)}$ for every element i covered by A_j, and this, in turn, guarantees the performance ratio αH_k for the coupled greedy algorithm. Secondly, for the problem that can be cast into the framework of Set k-Cover, the coupling lemma gives a general framework to obtaining an αH_k-approximation to the problem.

Our proof of the coupling lemma is based on the proof of the greedy algorithm for Set Cover in [14], and is different to the original proof of Slavík [13]. We should point out that Alfandari and Paschos [2] considered coupling the classical greedy algorithm for Set Cover with an approximation algorithm of finding an approximate covering set with respect to cost-effectiveness $\text{cost-ef}_n(S)$, instead of $\text{cost-ef}_v(S)$.

3. Approximating *k*-forest

3.1. Finding cost-effective covering tree

Then we turn back to the k-Forest problem. k-Forest can be viewed as a Set k-Cover-like problem in which the grounding set $U = \{d_1, d_2, \ldots, d_l\}$ where each $d_i = (s_i, t_i)$ denotes a demand, and the set family $\mathcal{S}$ is the collection of all such edge set E_j that the edges in E_j form a tree in G met some demands from U. In general $\mathcal{S}$ has exponential size.

In the framework of Set k-Cover for k-Forest, one of the key issues is to find the most cost-effective covering tree. Since there are exponential trees in G, it is obvious that one can not hope to find the most cost-effective covering tree by enumerating them one by one. Then we try to find an approximately optimal covering tree. Denote by F_j the collection

of trees selected by the algorithm at time j (actually, at the beginning of iteration j), and by R_j the set of demands not yet connected at time j. Let $C_j = D \backslash R_j$ be the set of demands already connected by the algorithm at this point. For succinctness, when the time j is clear in the context, F_j, C_j, R_j are also written as F, C, R, respectively. For a tree T in G, we define new(T) to be the number of demands in R connected by T, similarly to that in the setting of Set k-Cover. Since the goal is to connect at least k demands in D, given the already connected demands set C and selected tree collection F, adding T to F may connect more than k demands. The quantity that is beyond k is useless for the goal. Under this consideration, the function val$(\cdot)$ is defined as

$$\text{val}(T) = \min\{k', \text{new}(T)\},$$

where $k' = k - |C|$. That is, val(T) is the number of demands validly connected by T. Then the cost-effectiveness of tree T is defined as

$$\text{cost-ef}_v(T) = \frac{c(T)}{\text{val}(T)}.$$

The problem of finding the most cost-effective covering tree is to find a tree T such that cost-ef$_v(T)$ is minimized over all possible trees in G. Similarly, we also define the cost-effectiveness of a tree T with respect to the function new$(\cdot)$ as

$$\text{cost-ef}_n(T) = \frac{c(T)}{\text{new}(T)}.$$

Given the unconnected demand set R, suppose that T^* is the most cost-effective covering tree with respect to the function cost-ef$_v(\cdot)$. For clarity, T^* is also called the optimal covering tree. Segev et al. [12] gave a constructive method to find a covering tree, which is approximately optimal with respect to the function cost-ef$_n(\cdot)$ (which is referred to as the density function therein). We adapt their method to make it fit our settings, that is, to find a covering tree which is approximately optimal with respect to the function cost-ef$_v(\cdot)$. Before giving the algorithm, we first prove Lemma 2. We state the rooted quota-MST problem here, which is used in the proof of Lemma 2. Given a graph G with costs on edges and each vertex $v \in V(G)$ having profit$(v) \in \mathbb{Z}^+$, a vertex $r \in V(G)$, and a nonnegative quota Q, the rooted quota-MST problem asks to find a spanning tree $\widetilde{T}$ with minimized cost satisfying $\sum_{v \in V(\widetilde{T})} \text{profit}(v) \geq Q$. If each profit$(v)$ is polynomially bounded, this problem can be reduced to the k-MST problem and thus can be approximated within factor 2 by the work of Garg [5]. In Lemma 2, T^* is an optimal covering tree and is not known in advance.

Lemma 2. *Given $q = \text{val}(T^*)$, an arbitrary vertex $r \in T^*$, and an arbitrary tree T that contains r, then an augmented tree $T^+ \supseteq T$ can be constructed in polynomial time such that at least one of the two properties*

(1) $|V(T^+)| \geq |V(T)| + \sqrt{q}/2$, *and*
(2) $\text{val}(T^+) \geq 3q/8$

holds.

Proof. Suppose that $\text{new}(T) \geq q/2$. If $\text{new}(T) \leq k'$, then we have $\text{val}(T) = \min\{k', \text{new}(T)\} = \text{new}(T) \geq q/2$. If $\text{new}(T) > k'$, since $q = \text{val}(T^*) = \min\{k', \text{new}(T^*)\} \leq k'$, we know that $\text{val}(T) = \min\{k', \text{new}(T)\} = k' \geq q$. So T itself is a tree satisfying property (2) in the lemma and we are done.

So in the following we deal with the case $\text{new}(T) < q/2$.

If $|V(T^*) \setminus V(T)| \geq \sqrt{q}/2$ then one can construct a quota spanning tree $\widetilde{T}$ rooted at r by approximating the rooted quota-MST instance on G, in which profit$(v) = 1$ for all $v \in V(G) \setminus V(T)$, profit$(v) = 0$ for all vertices in $V(T)$, and the quota is $\sqrt{q}/2$. Now $T^+ = T \cup \widetilde{T}$ is a feasible tree satisfying the property (1) in the lemma. Such a tree $\widetilde{T}$ must exist, since T^* itself is a feasible quota spanning tree rooted at r.

Otherwise we have $|V(T^*) \setminus V(T)| < \sqrt{q}/2$. Once again, a quota spanning tree $\widetilde{T}$ rooted at r can be constructed by approximating the rooted quota-MST instance on G, in which profit(v) for all $v \in V(G) \setminus V(T)$ is equal to the number of demands in R consisting of v and an additional vertex from T, all vertices in $V(T)$ have zero profit, and the quota is $3q/8$.

We argue that such a spanning tree $\widetilde{T}$ must exist. Denote by $D(T)$ the set of demands covered by a tree T. For $0 \leq j \leq 2$, define set $N_j = \{(s_i, t_i) \in R \cap D(T^*) : |\{s_i, t_i\} \cap V(T)| = j\}$, that is, N_j is just the set of demands newly covered by T^* with exactly j endpoints in $V(T)$. Since even if all the vertices in $V(T^*) \setminus V(T)$ form demands newly covered by T^*, there are at most $\binom{|V(T^*) \setminus V(T)|}{2} \leq \binom{\lfloor \sqrt{q}/2 \rfloor}{2} \leq q/8$ new demands entirely in $V(T^*) \setminus V(T)$, eventually we have that $|N_0| \leq q/8$. Notice that $\text{new}(T^*) \geq \min\{k', \text{new}(T^*)\} = \text{val}(T^*) = q$ and $|N_2| \leq \text{new}(T) < q/2$. So we have

$$|N_1| = \text{new}(T^*) - |N_0| - |N_2| \geq q - |N_0| - |N_2| \geq q - \frac{q}{8} - \frac{q}{2} = \frac{3q}{8}.$$

Since $\sum_{v \in V(T^*) \setminus V(T)} \text{profit}(v) \geq |N_1| \geq 3q/8$, T^* itself is a feasible tree to the rooted quota-MST instance.

Now we get a tree $T^+ = T \cup \widetilde{T}$ with $\text{new}(T^+) \geq 3q/8$. Again, if $\text{new}(T^+) \leq k'$ then we have $\text{val}(T^+) = \min\{k', \text{new}(T^+)\} = \text{new}(T^+) \geq 3q/8$. If $\text{new}(T^+) > k'$ then we have $\text{val}(T^+) = \min\{k', \text{new}(T^+)\} = k' \geq q$ since $q = \text{val}(T^*) = \min\{k', \text{new}(T^*)\} \leq k'$. So, we always have that $\text{val}(T^+) \geq 3q/8$ and hence T^+ satisfies the property (2) in the lemma.

The lemma follows since we can solve the two instances separately and preferably pick the output on the second instance as the final output when the both instances are feasible, although T^* is unknown to us. □

We shall point out that although in the proof of Lemma 2 we adapt the constructive method in [12], our result $\text{val}(T^+) \geq 3q/8$ in Lemma 2 is stronger than that in [12], which is crucial to the success of our whole algorithm.

Algorithm $\mathcal{B}$

input: Graph G, integer $k' > 0$, and the uncovered demands set R.
output: A tree T covering some demands in R.

(denote by T^* an optimal covering tree)
(1) guess $q = \text{val}(T^*)$ from $\{1, \ldots, k'\}$ and an arbitrary vertex $r \in T^*$ from $V(G)$
(2) **if** $q < n^{2/3}$ **then return** the shortest path connecting any demand in R and **stop**
(else we have that $q \geq n^{2/3}$)
(3) **let** $T \leftarrow \{r\}$
(4) repeatedly extend T by Lemma 2 until $\text{val}(T) \geq 3q/8$
(5) **return** T

For algorithm $\mathcal{B}$ we have Lemma 3.

Lemma 3. *In polynomial time, algorithm $\mathcal{B}$ finds a covering tree T satisfying that $\text{cost-ef}_v(T) \leq \alpha \cdot \text{cost-ef}_v(T^*)$, where $\alpha = O(n^{2/3})$ and T^* is an optimal covering tree with respect to the function $\text{cost-ef}_v(\cdot)$.*

Proof. Suppose that algorithm $\mathcal{B}$ guesses the right $q = \text{val}(T^*)$ and $r \in T^*$. If $q < n^{2/3}$, the algorithm returns the shortest path connecting a new demand in R as the covering tree T. Obviously in this case $\text{val}(T) = 1$. So we get

$$\text{cost-ef}_v(T) = \frac{c(T)}{\text{val}(T)} = c(T) \leq c(T^*) \leq n^{2/3} \cdot \frac{c(T^*)}{q} = \alpha \cdot \text{cost-ef}_v(T^*),$$

where the first inequality holds since T^* connects at least one new demand.

Next we consider the case $q \geq n^{2/3}$. By the constructive proof of Lemma 2, we know that T is extended by approximating the problem rooted quota-MST, which can be approximated with factor 2 by the work of [5]. Before $\text{new}(T) \geq 3q/8$, at least $\sqrt{q}/2$ new vertices are added to T in each iteration of step 4. So the number of iterations is at most $\frac{n}{\sqrt{q}/2} + 1 \leq 2n^{2/3} + 1$, and T is extended at most $2n^{2/3} + 1$ times from the trivial tree $\{r\}$, whose cost is zero. Thus we have $c(T) \leq (2n^{2/3} + 1) \cdot 2c(T^*)$. By step 4 of the algorithm, we have

$$\text{cost-ef}_v(T) = \frac{c(T)}{\text{val}(T)} \leq \frac{c(T)}{3q/8} \leq \frac{8}{3}(2n^{2/3} + 1)\frac{2c(T^*)}{q} = \alpha \cdot \text{cost-ef}_v(T^*)$$

provided $q \geq n^{2/3}$.

Finally, trying all the possible $1 \leq q \leq k'$ and $r \in V(G)$ and picking the tree with minimal $\text{cost-ef}_v(\cdot)$ completes the proof. □

3.2. The greedy algorithm for k-forest

Now we are ready to deduce the greedy algorithm for k-Forest. The algorithm is given as algorithm $\mathcal{C}$. In each iteration of algorithm $\mathcal{C}$, the algorithm finds an approximate covering tree T and adds T to F, until at least k demands have been connected. At last, the algorithm outputs F as the solution. As before, the notation k' denotes the number of demands that need to be covered at the beginning of iteration j.

Algorithm $\mathcal{C}$

input: Graph G, demand set $D = \{(s_1, t_1), \ldots, (s_l, t_l)\}$, integer $k > 0$.
output: A collection of trees F that connects at least k demands in D.

(1) **let** $F \leftarrow \emptyset, C \leftarrow \emptyset, R \leftarrow D$
(2) **while** $k' = k - |C| > 0$ **do**
(3) find a covering tree T by algorithm $\mathcal{B}$
(4) **let** $F \leftarrow F \cup \{T\}, C \leftarrow C \cup D(T), R \leftarrow R \setminus D(T)$
(5) **end**
(6) **return** F

In step 4 of algorithm $\mathcal{C}$, the notation $D(T)$ denotes the set of demands connected by T. For algorithm $\mathcal{C}$, we have the main theorem of this paper.

Theorem 4 (*The Main Theorem*). *k-Forest can be approximated within factor $O(n^{2/3} \log k)$ in polynomial time.*

Proof. By Lemma 3, in each iteration of algorithm $\mathcal{C}$ an approximate covering tree T is found in polynomial time, such that cost-ef$_v(T) \leq \alpha \cdot$ cost-ef$_v(T^*)$ where $\alpha = O(n^{2/3})$ and T^* is an optimal covering tree at this point. By the coupling lemma, algorithm $\mathcal{C}$ yields an αH_k-approximation to k-Forest. Also, algorithm $\mathcal{C}$ is of polynomial time, since there are at most k iterations in total. The theorem follows. □

It is clear that algorithm $\mathcal{C}$ has at most k iterations. In each iteration of algorithm $\mathcal{C}$, algorithm $\mathcal{B}$ is called and has to try $O(kn)$ guesses. The time used in each guess of algorithm $\mathcal{B}$ is $O(n^{2/3}t)$, where t is the time used to build a quota-MST by the algorithm in [5].[1] So the total time complexity of algorithm $\mathcal{C}$ is $O(k^2n^{5/3}t)$.

4. Discussion

Segev et al. proposed a separate construction method in [12] similar to the one in Lemma 2. For the details of the construction the readers may refer to [12]. Using the method in [12], we are able to construct an approximately optimal covering tree T such that cost-ef$_v(T) \leq O(\sqrt{l}) \cdot$ cost-ef$_v(T^*)$. Then, by the coupling lemma, we know that k-Forest can also be approximated within factor $O(\sqrt{l}\log k)$ in polynomial time.

We obtain an improved performance ratio $O(n^{2/3}\log k)$ for k-Forest via greedy approach, while the previous best known ratio in the literature is $O(n^{2/3}\log l)$, obtained by Segev et al. [12] through the Lagrangian relaxation technique. Our global framework is completely different from that in [12]. Another difference is that our algorithm is simpler and faster. We believe that the coupling lemma for Set k-Cover has its own independent interest. To improve the performance ratio further for k-Forest in the framework of Set k-Cover, it seems that entirely new constructive method for building a covering tree is needed. Furthermore, the constructive method used in this paper does not take advantage of the special structure of the underlying graph, implying that approximating the problem k-Forest in trees, which is also proposed by Hajiaghayi and Jain [9], still remains open.

Acknowledgements

The authors are grateful to their supervisor Prof. Angsheng Li for advice and encouragement. The authors are partially supported by NSFC Grants No. 60325206 and No. 60310213.

References

[1] A. Agrawal, P. Klein, R. Ravi, When trees collide: An approximation algorithm for the generalized Steiner problem on networks, SIAM Journal on Computing 24 (3) (1995) 440–456.
[2] L. Alfandari, V. Paschos, Master-slave strategy and polynomial approximation, Computational Optimization and Applications 16 (2000) 231–245.
[3] F. Chudak, T. Roughgarden, D. Williamson, Approximate k-MSTs and k-Steiner trees via the primal-dual method and Lagrangean relaxation, Mathematical Proramming 100 (2) (2004) 411–421.
[4] U. Feige, G. Kortsarz, D. Peleg, The dense k-subgraph problem, Algorithmica 29 (2001) 410–421.
[5] N. Garg, Saving an epsilon: A 2-approximation for the k-MST problem in graphs, in: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, STOC'05, 2005, pp. 302–309.
[6] M. Goemans, D. Willamson, A general approximation technique for constrained forest problems, SIAM Journal on Computing 24 (2) (1995) 296–317.
[7] K. Jain, M. Mahdian, E. Markaksi, A. Saberi, V. Vazirani, Greedy facility location algorithms analyzied using dual fitting with factor-revealing LP, Journal of the ACM 50 (6) (2003) 795–824.
[8] K. Jain, V. Vazirani, Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation, Journal of the ACM 48 (2) (2001) 274–296.
[9] M. Hajiaghayi, K. Jain, The prize-collecting generalized Steiner tree problem via a new approach of primal-dual schema, in: Proceedings of the 17th Annual ACM–SIAM Symposium on Discrete Algorithms, SODA'06, 2006, pp. 631–640.
[10] S. Khot, Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique, in: Proceedings of the 44th Annual IEEE Syposium on the Foundations of Computer Science, FOCS'04, 2004, pp. 136–145.
[11] A. Levin, D. Segev, Partial multicuts in trees, in: Proceedings of the 3rd International Workshop on Approximation and Online Algorithms, WAOA'05, 2005, pp. 320–333.
[12] D. Segev, G. Segev, Approximate k-Steiner Forests via the Lagrangian relaxation technique with internal preprocessing, in: Proceedings of the 14th Annual European Symposium on Algorithms, ESA'06, 2006, pp. 600–611.
[13] P. Slavík, Improved performance of the greedy algorithm for partial cover, Information Processing Letters 64 (1997) 251–254.
[14] V. Vazirani, Approximation Algorithms, 2nd edition, Springer-Verlag, Berlin, Heidelberg, 2003.

[1] The time complexity t is not explicitly given in [5]. Since the algorithm in [5] uses the Goemans–Williamson algorithm [6] as its underlying algorithm, which has time complexity $O(n^2\log n)$, it is believed that for algorithm $\mathcal{B}$ the time used in step 4 dominates the time used in step 2 ($O(n^2)$).