A constant factor approximation for minimum λ-edge-connected k-subgraph with metric costs

MohammadAli Safari[⋆] Mohammad R. Salavatipour[⋆⋆]
Department of Computing Science
University of Alberta
Edmonton, Alberta T6G 2E8, Canada
{msafarig,mreza}@cs.ualberta.ca

Abstract. In the (k, λ)-subgraph problem, we are given an undirected graph $G = (V, E)$ with edge costs and two parameters k and λ and the goal is to find a minimum cost λ-edge-connected subgraph of G with at least k nodes. This generalizes several classical problems, such as the minimum cost k-Spanning Tree problem or k-MST (which is a $(k, 1)$-subgraph), and minimum cost λ-edge-connected spanning subgraph (which is a $(|V(G)|, \lambda)$-subgraph). The only previously known results on this problem [12, 5] show that the $(k, 2)$-subgraph problem has an $O(\log^2 n)$-approximation (even for 2-node-connectivity) and that the (k, λ)-subgraph problem in general is almost as hard as the densest k-subgraph problem [12]. In this paper we show that if the edge costs are metric (i.e. satisfy triangle inequality), like in the k-MST problem, then there is an $O(1)$-approximation algorithm for (k, λ)-subgraph problem. This essentially generalizes the k-MST constant factor approximability to higher connectivity.

1 Introduction

Network design is a central topic in combinatorial optimization, approximation algorithms, and operations research. A fundamental problem in network design is to find a minimum cost subgraph satisfying some given connectivity requirements between vertices. Here by a network we mean an undirected graph together with non-negative costs on the edges. For example, with a connectivity requirement $\lambda = 1$ between all the vertices, we have the classical minimum spanning tree problem. For larger values of λ, i.e. finding minimum cost λ-edge-connected spanning subgraphs, the problem is APX-hard. These are special cases of the more general problem of survivable network design problem (SNDP), in which we have a connectivity requirement of r_{uv} between every pair u, v of vertices. Even for this general setting there is a 2-approximation algorithm by Jain [11].

A major line of research in this area has focused on problems with connectivity requirements where one has another parameter k, and the goal is to find a subgraph satisfying the connectivity requirements with a lower bound k on the total number of vertices. The most well-studied problem in this class is the minimum k-spanning tree problem, a.k.a. k-MST. In this problem, we have to find a minimum cost connected

⋆ Supported by Alberta Ingenuity.
⋆⋆ Supported by NSERC and an Alberta Ingenuity New Faculty award.

subgraph spanning at least k-vertices. The approximation factor for this problem was improved from $\sqrt{k}$ and $O(\log^2 k)$ in [14, 1] down to $O(\log n)$ in [13] and to a constant in [3, 9] and recently to 2 [10]. The algorithm of [10] can be used to obtain a constant approximation for the slightly more general setting in which we have a set of nodes T, called terminals, and the goal is to find a minimum cost connected subgraph containing at least k terminals. This is known as the k-Steiner tree problem. The problem of k-TSP, in which one has to find a minimum cost TSP tour containing at least k nodes, can be approximated using very similar technique. We note that in all these problems, the input graph is assumed to be complete and the edge cost function is metric, i.e. satisfies triangle inequality. Most of these problems are motivated from their applications in vehicle routing or profit maximization with respect to a given fixed budget. For example, suppose that we have a battery operated robot and the goal is to find the minimum battery charge required to travel a sequence of at least k nodes in a given graph such that the total length of the tour can be travelled in a single battery charge. See [2, 6] for similar problems.

Recently, Lau et al. [12] considered a very natural common generalization of both the k-MST problem and minimum cost λ-edge-connected spanning subgraph problem, which they called the (k, λ)-subgraph problem. In this problem, we are given a graph $G = (V, E)$ with a (not necessarily metric) cost function $c : E \rightarrow \mathbb{R}^+$ on the edges, a positive integers k, and a connectivity requirement $\lambda \geq 1$; the goal is to find a minimum cost λ-edge-connected subgraph of G with at least k vertices. We should point out that we are not allowed to take more copies of an edge than are present in the graph. Otherwise, a 4-approximate solution can be computed by taking a 2-approximate k-MST solution T, and then taking λ copies of T. It is easy to observe that the (k, λ)-subgraph problem contains, as special cases, the minimum cost λ-edge-connected spanning subgraph problem (it becomes the $(|V(G)|, \lambda)$-subgraph problem), and the k-MST problem (which becomes the $(k, 1)$-subgraph problem). Lau et al. [12] present an $O(\log^2 n)$-approximation for $(k, 2)$-subgraph and show that for arbitrary values of λ, (k, λ)-subgraph is almost as hard as the k-densest subgraph problem[1]. In the k-densest subgraph problem, one has to find a subgraph with k nodes in a given graph G that has the largest number of edges. Despite considerable attempts, the best known approximation algorithm for this problem has ratio $O(n^{\frac{1}{3}-\epsilon})$ for some fixed $\epsilon > 0$ [8]. Chekuri and Korula [5] have recently (and independently of [12]) shown that an algorithm similar to the one in [12] yields an $O(\log^2 n)$-approximation for the $(k, 2)$-subgraph problem even if we want a 2-node-connectivity requirement in the solution.

In light of the result of [12] on the hardness of (k, λ)-subgraph for arbitrary values of λ and general cost functions, it is natural to try to obtain good approximation algorithms for the class of graphs where the edge cost function is metric, i.e. satisfies triangle inequality. Remember that the constant factor approximation algorithms for k-MST and k-TSP are on graphs with metric cost function. Our main result of this paper is the following theorem:

[1] The extended abstract version claimed an $O(\log^3 n)$-approximation but the proof was inaccurate. The full version has the improved result with a complete proof

Theorem 1. *Given a (complete) graph G with metric costs on the edges and two parameters k, λ, there is an $O(1)$-approximation algorithm for finding a (k, λ)-subgraph in G.*

Our algorithm is combinatorial and uses ideas from [4] for metric-cost subset node-connectivity problem as well as the algorithm for k-Steiner tree [7, 10, 14]. The constant factor we obtain is relatively large (between 400 and 500), however, most of our efforts have been to show that the problem has a constant factor approximation rather than trying to obtain the best possible ratio.

The organization of the paper is as follows. We start by some definitions and preliminary bounds used throughout paper in the next section. For the ease of exposition, we first present an algorithm that finds a λ-edge-connected subgraph with at least $k - \lambda/7$ nodes whose cost is at most $O(\text{OPT})$. In Section 3 we show how to extend this solution to a feasible solution to the (k, λ)-subgraph problem while keeping the total cost still bounded by $O(\text{OPT})$. We finish the paper with some concluding remarks.

2 Preliminaries

As mentioned earlier, we assume we are given a (complete) graph $G = (V, E)$, with a cost function $c : E \rightarrow \mathbb{R}^+$ on the edges that satisfies triangle inequality, and two positive integers k and $\lambda \geq 1$. For every subgraph $F \subseteq G$, we use $c(F)$ to denote the total cost of the edges in F. Throughout, $G^* \subseteq G$ denotes the optimum solution and $\text{OPT} = c(G^*)$ denotes the cost of the optimum solution. We will use two lower bounds on OPT in the analysis of our algorithm. These lower bounds were used earlier in [4] for the problem of minimum cost subset k-node-connectivity. The first lower bound comes from the cost of a minimum spanning tree of the G^*, which we call it T^*. Considering the cut-constraint IP formulation of MST, it is easy to see that $\frac{\lambda}{2} \sum_{e \in T^*} c_e \leq \text{OPT}$. The second lower bound comes from the minimum cost subgraph that has minimum degree at least λ. Note that in a λ-edge-connected subgraph, every vertex has degree at least λ. For any λ-edge-connected subgraph $F \subseteq G$ and any vertex $u \in F$ we let $S_u(F)$ to be the set of λ nearest neighbors of u in F and $s_u(F)$ be $\sum_{v \in S_u(F)} c_{uv}$. Clearly, for any λ-edge-connected subgraph $F \subseteq G$ and any vertex $u \in F$: $s_u(G) \leq s_u(F)$. We often use S_u and s_u instead of $S_u(G)$ and $s_u(G)$, respectively, unless the graph is different from G. It is easy to see that $\frac{1}{2} \sum_{u \in G^*} s_u \leq \frac{1}{2} \sum_{u \in G^*} s_u(G^*) \leq \text{OPT}$. Thus, if T^* is a minimum spanning tree of G^*, then we obtain the following two lower bounds for OPT: (i) $\frac{1}{2} \sum_{u \in T^*} s_u \leq \text{OPT}$, and (ii) $\frac{\lambda}{2} \sum_{e \in T^*} c_e \leq \text{OPT}$, and in particular:

$$\frac{1}{2} \sum_{u \in T^*} s_u + \frac{\lambda}{2} \sum_{e \in T^*} c_e \leq 2\text{OPT}. \tag{1}$$

In our algorithm we will use these two lower bounds frequently, often without referring to them.

We present an algorithm for a modified version of the problem in which along with G, k, and λ, we are also given a vertex $r \in G$ as the root which we are told belongs to the optimum solution G^* and among all the vertices in G^* it has the smallest value s_u.

Clearly if we can solve this rooted version, then we can try every vertex as the root and return the minimum solution among all as the final answer.

Ravi et al [14] showed that any α-approximation for k-MST implies a 2α-approximation for k-Steiner tree. Therefore, together with Garg's algorithm [10], we have a 4-approximation for k-Steiner tree. In fact, we can have a 4-approximation algorithm for the rooted version of the problem, in which a specific vertex r is given as the root and the goal is to find a minimum cost rooted at r Steiner tree containing at least k terminals. Our algorithm will use the best known approximation algorithm for finding a minimum cost rooted k-Steiner tree problem; let us denote the approximation ratio of this algorithm by ρ (by the argument above, we know that $\rho \leq 4$). We denote this approximation algorithm by ST-Alg.

3 Obtaining a low cost $(k - O(\lambda), \lambda)$-subgraph

Observe that to have λ-edge-connectivity, we must have $k \geq \lambda + 1$. We start by presenting an algorithm that returns a λ-edge-connected subgraph (containing root r) that has at least $k - \lambda/7$ nodes and whose cost is within constant factor of OPT. Our algorithm is influenced by the work of Cheriyan and Vetta [4] for minimum cost subset k-node-connectivity.

3.1 Overview of the Algorithm

The main idea of the algorithm is as follows. We create a new graph $G'(V \cup V', E')$ from G by creating a new (dummy) vertex u' (in V') for each vertex $u \in G$ and $E' = E \cup \{uu' | u \in V\}$. Each edge $uu' \in E'$ has weight s_u. For every other edge in G' (that also exists in G) we multiply its weight by λ. Suppose that T^* is an optimum (rooted at r) k-Steiner tree of G' with terminal set V'. We show that $c(T^*) \leq 4\text{OPT}$. We can obtain an approximation of T^*, call it T', by running the ST-Alg. Let us assume that $\tilde{T} \subset G$ is the tree obtained from T' by deleting the dummy vertices and for ease of exposition, suppose that T^* and $\tilde{T}$ are binary trees. For simplicity, suppose that all these sets S_u (for $u \in \tilde{T}$) are disjoint and let us assume that $v_1, \ldots, v_p$ (with $p = k/(\lambda + 1)$) have the smallest s_u values among all the nodes in $\tilde{T}$. Our next steps would be to obtain a λ-edge-connected subgraph by selecting v_i and S_{v_i} (for $1 \leq i \leq p$) and forming a $(\lambda + 1)$-clique on each to get λ-edge connectivity among themselves. The cost of each of these cliques will be at most λs_{v_i}, for $1 \leq i \leq p$, and since $p \approx k/\lambda$ the total cost of all the cliques is at most $X = \sum_{u \in \tilde{T}} s_u$. Considering each of these cliques as a big "blub", we need to establish λ-edge connectivity among these blubs. For that we need to find a tree with the blubs being the nodes and for each edge in the tree between two blubs we add about λ edges between the cliques corresponding to the blubs to maintain λ-edge-connectivity. We can use the structure of $\tilde{T}$ itself to establish a tree over these blubs. Roughly speaking, the total cost of all the λ edges between the blubs will be at most $O(\lambda)$ times the edges in $\tilde{T}$ (using triangle inequality) which is $Y = O(\lambda \sum_{e \in \tilde{T}} c_e)$. Noting that $X + Y \leq O(c(T'))$ implies that we will have a solution within constant factor of the optimum. The main difficulty here will be the possible (or lack of) intersections between S_u and S_v for two vertices $u, v \in T$. A lot of details have been skipped over in this overview and are explained in the next subsection.

3.2 Details of the Algorithm

We build the graph G' from G as described above and compute an approximate (rooted) k-Steiner tree with terminal set $V' = \{v' | v \in V(G)\}$ and root r' (copy of r) using the ST-Alg. Let's call this tree T'.

Lemma 1. $c(T') \leq 4\rho\text{OPT}$.

Proof. Consider the optimal solution G^* to the (k, λ)-subgraph problem on G and let T^* be a MST of G^* (we assume that $r \in T^*$). Then $\tilde{T} = T^* \cup \{uu' \in G' | u \in T^*\}$ is clearly a Steiner tree in G' containing at least k terminals with total cost at most 4OPT (using the bound in (1) for T^*). The lemma follows by observing that the ST-Alg (for k-Steiner tree) has approximation ratio ρ.

Without loss of generality, we can assume that T' has exactly k terminals, as if it has more we can safely delete them. Let $T_0 \subset G$ be the tree obtained from $T' \subset G'$ by deleting the dummy nodes (i.e. the nodes in V') and V_0 be the vertex set of T_0. Note that by Lemma 1:

$$\lambda \sum_{e \in T_0} c_e + \sum_{u : u' \in T'} s_u \leq 4\rho\text{OPT}. \tag{2}$$

We should also point out that V_0 might have some vertices $v \in V$ (and therefore $v \in T'$) but $v' \notin T'$. We obtain another tree $T_1 = (V_1, E_1) \subset G$ from T_0 with the following properties: (i) $V_1 \subseteq V_0$, (ii) $c(T_1) \leq 2c(T_0)$, and (iii) for every vertex $v \in V_1$, the corresponding vertex $v' \in G'$ belongs to T'. To do this, we duplicate every edge of T_0 and do an Eulerian walk of T_0; now shortcut over every vertex $v \in T_0$ with $v' \notin T'$. It is easy to see that we are left with a tree T_1 whose cost is at most $2c(T_0)$ and every vertex $v \in T_1$ has its copy v' in T'. Also, property (iii) implies that T_1 has exactly k vertices. Thus:

Lemma 2. *$V_1 \subseteq V_0$, with $|V_1| = k$ and $c(T_1) \leq 2c(T_0)$.*

Suppose that we have an ordering of the vertices of T_1, say $v_1 = r, v_2, \ldots, v_k$, such that $s_{v_2} \leq s_{v_3} \leq \cdots \leq s_{v_k}$. Note that although r has the smallest s_u value among all vertices $u \in G^*$, it is not necessarily the case in T_1. For each $1 \leq i \leq k$, let $\mu_i = \frac{s_{v_i}}{\lambda}$. We call S_{v_i} the ball of v_i and the core of S_{v_i}, denote by B_{v_i}, is the set of nodes in S_{v_i} with distance at most $2\mu_i$ to v_i. By a simple averaging argument, one can easily show that $|B_{v_i}| \geq \lambda/2$. We partition the nodes of T_1 into two sets of *active* and *inactive* nodes using the following procedure to cluster the balls. Initially, all the nodes of T_1 are active and we have $\mathcal{S} = \emptyset$ ($\mathcal{S}$ will contain the centers of active balls). For each $1 \leq i \leq k$, if v_i is active and there is no $v_j \in \mathcal{S}$ (with $j < i$) such that $c_{ij} \leq 4\mu_i + 2\mu_j$ then add v_i to $\mathcal{S}$ and make all the nodes in S_{v_i} inactive (except v_i itself). Note that S_{v_i} might include some vertices not in T_1. So at the end, for every two active nodes $v_i, v_j \in \mathcal{S}$ (with $j < i$) we have $c_{ij} > 4\mu_i + 2\mu_j$ and $B_{v_i} \cap B_{v_j} = \emptyset$. Now for every value of $1 \leq i \leq k$ such that v_i is active but $v_i \notin \mathcal{S}$, there exists a $j < i$ such that $v_j \in \mathcal{S}$ and $c_{ij} \leq 4\mu_i + 2\mu_j$. Let j^* be the smallest such index and define $p(i) = j^*$, meaning that v_i is assigned to ball $S_{v_{j^*}}$. So each active node v_i is either the center of an active ball (and it belongs to $\mathcal{S}$) or is assigned to a ball $S_{p(i)}$ with $p(i) \in \mathcal{S}$, and all the remaining nodes (that are inside the balls with centers in $\mathcal{S}$) are inactive. Thus:

Lemma 3. *Every core B_{v_i}, for $v_i \in \mathcal{S}$, is disjoint from any other B_{v_j} (with $v_j \in \mathcal{S}$, $j \neq i$) and $|B_{v_i}| > \frac{\lambda}{2}$.*

For every value of i, consider the union of active nodes v_j and their ball S_{v_j} (if $v_j \in \mathcal{S}$), for all $j \leq i$, and define this set of vertices U_i, i.e.

$$U_i = \{\text{active nodes } v_j \text{ with } j \leq i\} \cup \bigcup_{\substack{\text{active } v_j,\, j \leq i, \\ v_j \in \mathcal{S}}} S_{v_j}. \tag{3}$$

Let $i^* \leq k$ be the smallest index such that $|U_{i^*}| \geq k - \lambda/7$. It is easy to see that from the definition of U_i and the choice of i^*:

Lemma 4. *$k - \frac{\lambda}{7} \leq |U_{i^*}| \leq k + \frac{6\lambda}{7}$ and if $|U_{i^*}| > k$ then $v_{i^*} \in \mathcal{S}$ and $S_{v_{i^*}}$ has at least $\frac{\lambda}{7} + 2$ vertices not in U_{i^*-1}.*

The solution of our algorithm will be a graph on vertex set U_{i^*}.

Let V_2 be the set of active nodes in $\mathcal{S}$ with index at most i^*. We compute a tree $T_2 = (V_2, E_2)$ starting from T_0 as follows. Duplicate every edge in T_0 and find an Eulerian tour. Shortcut all the edges that go through vertices that were deleted while computing T_1 from T_0 or those vertices of T_1 that are not in V_2. The cost of T_2 is clearly at most $2c(T_0)$, using triangle inequality. Also, it only contains a subset of vertices of T_1, namely V_2. Thus:

Lemma 5. *$V_2 \subseteq V_1$ and $c(T_2) \leq 2c(T_0)$.*

Note that T_2 is in fact a path, so the maximum degree of every vertex in T_2 is at most 2. The next steps of the algorithm would be to make a $(\lambda+1)$-clique over $S_{v_i} \cup \{v_i\}$, for each $v_i \in T_2$ which are precisely those $v_i \in \mathcal{S}$ with $i \leq i^*$. For each active node $v_i \notin \mathcal{S}$, we connect v_i to all the λ vertices in $S_{p(i)}$. It is easy to observe that each ball S_{v_i} with $v_i \in \mathcal{S}$ together with all the active nodes assigned to it will form a λ-edge-connected subgraph. The final step is to make good connectivity between these balls. For that, we look at every edge $v_i v_j \in T_2$; note that both $v_i, v_j \in \mathcal{S}$. Let $a = |S_{v_i} \cap S_{v_j}|$. We add an arbitrary matching (of size $\lambda - a$) between the $\lambda - a$ vertices in $S_{v_i} - S_{v_j}$ and $S_{v_j} - S_{v_i}$. The full description of the algorithm, is given in Figure 1, and Figure 2 illustrates the approximate Steiner tree computed and the balls around the active nodes and some of the edges added to make the graph λ-edge-connected.

3.3 Analysis of Algorithm

It is easy to see that H contains exactly those active nodes v_i with $i \leq i^*$ as well as all the nodes in $\bigcup_{v_{i \leq i^*} \in \mathcal{S}} S_{v_i}$; which is exactly set U_{i^*}. Thus, by Lemma 4:

Lemma 6. *$k - \frac{\lambda}{7} \leq |H| \leq k + \frac{6\lambda}{7}$ and if $|H| > k$ then at the iteration in which v_{i^*} is added to H, $S_{v_{i^*}} \cup \{v_{i^*}\}$ adds at least $\frac{\lambda}{7} + 2$ new vertices to H.*

In the remaining of this subsection we show that H has the required connectivity while its cost is bounded by $O(\textsc{opt})$.

1 Build graph $G' = (V' \cup V, E')$ by starting from G and creating a new vertex u' for every $u \in G$ and adding the edge uu' to G'; define $c_{u'u} = s_u$ and the cost of every other edge in G' (that also belongs to G) is multiplied by λ.
2 Compute an approximate k-Steiner tree T' (using ST-Alg) with terminals V'.
3 Compute tree $T_1 = (V_1, E_1)$ from T' (as described) which only consist of vertices v s.t. $v' \in T'$.
4 Compute the clustering of active balls $\mathcal{S}$; let i^* be the first index such that the union of active nodes v_i (and their ball S_{v_i} if it belongs to $\mathcal{S}$), for all $i \leq i^*$, is at least $k - \lambda/7$, and let $V_2 = \{v_i \in \mathcal{S} | i \leq i^*\}$.
5 Compute a tree $T_2 = (V_2, E_2)$ out of T_0 s.t. $c(T_2) \leq 2c(T_0)$; we first duplicate edges in T_0, find a Eulerian tour, and shortcut all the edges that go through vertices not in V_2.
6 Let H be an empty graph on vertex set consisting of union of all active nodes v_i with $i \leq i^*$. and $\bigcup_{v_i \in \mathcal{S}} B_{v_i}$.
7 **foreach** *active node* v_i *with* $i \leq i^*$ **do**
8   **if** $v_i \in \mathcal{S}$ **then**
9   Add v_i and every $u \in S_{v_i}$ to H as well as every edge uv, with $u, v \in S_{v_i} \cup \{v_i\}$
10   **else**
11   Add v_i to H and every edge uv_i, with $u \in S_{p(i)}$
12   **end**
13 **end**
14 **foreach** *edge* $v_i v_j \in T_2$ **do**
15   Add an arbitrary matching of size $\lambda - |S_{v_i} \cap S_{v_j}|$ from $S_{v_i} - S_{v_j}$ to $S_{v_j} - S_{v_i}$ in H
16 **end**
17 return H

Fig. 1. Algorithm 1, which is an approximation algorithm for low cost $(k - \lambda/7, \lambda)$-subgraph

Lemma 7. *Solution H returned by the algorithm is λ-edge-connected.*

Proof. For every $v \in H$, let us define the hub for v, denoted by $h(v)$, to be (i) v itself if $v \in \mathcal{S}$, (ii) $p(i)$ if $v = v_i$ is an active node but not in $\mathcal{S}$, and (iii) $v_\ell \in \mathcal{S}$ if v_ℓ is the first vertex added to $\mathcal{S}$ with $v \in S_{v_\ell}$. Observe that the set of hub nodes are precisely the nodes in $\mathcal{S}$ with index at most i^*, which is the same as the set of nodes of T_2. First it is easy to see that each v has λ-edge connectivity to its hub (for case (iii) we have made a clique out of $h(v)$ and all the vertices in its ball including v, and for case (ii) v is adjacent to λ vertices in the clique made from the ball of $h(v)$). So it is enough to show that we have λ-edge-connectivity between the hub vertices. For any two adjacent vertices $v_i, v_j \in T_2$, the matching edges added between the balls of v_i and v_j (together with possible nodes in $S_{v_i} \cap S_{v_j}$) establish λ-edge-connectivity between v_i and v_j. By transitivity, we have λ-edge-connectivity between any pair of nodes $v_i, v_j \in T_2$.

Lemma 8. *The cost of edges of H added in line 15 is at most $8\rho\text{OPT}$.*

Proof. Let $v_i v_j$ be an edge in T_2, thus $v_i, v_j \in \mathcal{S}$. For any edge xy with $x \in S_{v_i}$ and $y \in S_{v_j}$ that we add in line 15: $c_{xy} \leq c_{xv_i} + c_{v_i v_j} + c_{v_j y}$. Since the matching added between the balls of v_i and v_j has size at most λ, the cost of this matching is at most

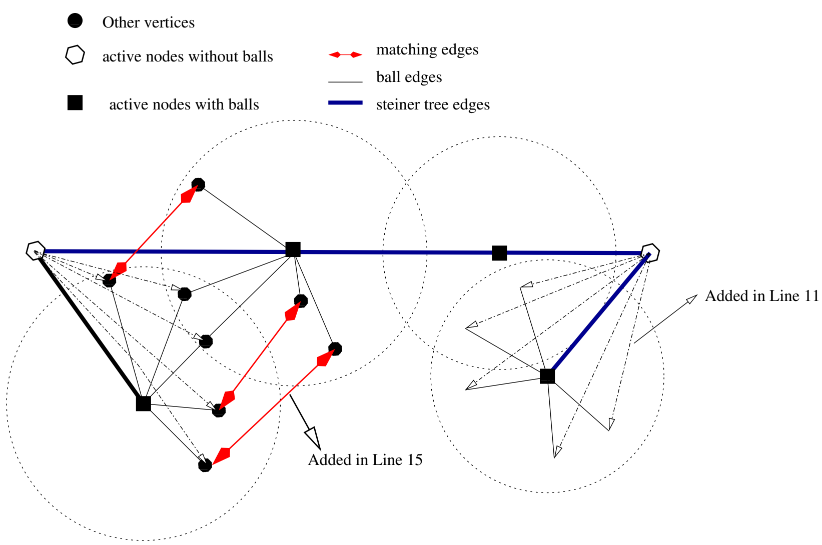

Fig. 2. A sample showing how the edges between the balls of active nodes are added and how the active nodes without a ball are connected to the balls of other active nodes.

$\lambda c_{v_i v_j} + s_{v_i} + s_{v_j}$. Noting that the degree of each vertex $v_i \in T_2$ is at most 2, there are at most two vertices like v_j with $v_i v_j \in T_2$ for which we have to add a matching between the balls of S_{v_i} and S_{v_j}. So the total cost of all the edges added in line 15 is at most: $\lambda \sum_{e \in T_2} c_e + 2 \sum_{v_i \in T_2} s_{v_i}$. Using Equation (2) and noting that $V_2 \subseteq V_0$ and $c(T_2) \leq 2c(T_0)$ (by Lemma 5), the total cost of the edges added in line 15 is at most $2\lambda \sum_{e \in T_0} c_e + 2 \sum_{v_i \in T_0} s_{v_i} \leq 8\rho\text{OPT}$.

In order to bound the cost of the edges added in lines 9 and 11 we need the following lemma.

Lemma 9. *For every $v_i \in \mathcal{S}$, with $i \geq 2$, (that is every node in $\mathcal{S}$ except the root $r = v_1$) and every node $v_j \in T_1$ with $c_{v_i v_j} \leq 2\mu_i$ such that v_j became inactive once we added v_i to $\mathcal{S}$: $\mu_i \leq 2\mu_j$.*

Proof. If $i < j$ (i.e. v_i was considered before v_j) then clearly $s_{v_i} \leq s_{v_j}$ and therefore $\mu_i \leq \mu_j$. Now suppose that $i > j$. It means that v_j was an active node but not in $\mathcal{S}$ at the time v_i was examined. This can happen only if there is $\ell < j$ with $v_\ell \in \mathcal{S}$ and

$$c_{v_j v_\ell} \leq 4\mu_j + 2\mu_\ell. \tag{4}$$

On the other hand, since v_i was not inactivated by v_ℓ:

$$c_{v_i v_\ell} > 4\mu_i + 2\mu_\ell \tag{5}$$

Using triangle inequality:

$$c_{v_i v_\ell} \leq c_{v_i v_j} + c_{v_j v_\ell}. \tag{6}$$

Combining (4), (5), and (6), together with the assumption that $c_{v_i v_j} \le 2\mu_i$ implies that: $4\mu_i + 2\mu_\ell < c_{v_i v_\ell} \le c_{v_i v_j} + c_{v_j v_\ell} \le 2\mu_i + 4\mu_j + 2\mu_\ell$; therefore $\mu_i \le 2\mu_j$ as wanted.

Lemma 10. *The cost of edges of H added in lines 9 and 11 is at most* $(2 + 28\rho)\text{OPT}$.

Proof. We will charge the cost of the edges added to H to the vertices incident to them (at the time those edges and vertices are added) and show that the total charge is bounded by $O(\text{OPT})$. To achieve this goal, we make sure that for every vertex $v_i \in H \cap T_1$, the charge assigned to v_i is no more than $7s_{v_i}$ and for every vertex $u \in H - T_1$ (which implies it is added by adding S_{v_i} for some $v_{i \le i^*} \in \mathcal{S}$), the charge assigned to u is no more than $2s_{v_i}$. We consider the following cases.

Case 1: First consider an active node $v_{i \le i^*} \notin \mathcal{S}$; so $i \ge 2$ and we have added vertex v_i to H plus every edge uv_i with $u \in S_{p(i)}$ in line 11. Let us assume $p(i) = j^*$. Note that $c_{uv_i} \le c_{uv_{j^*}} + c_{v_i v_{j^*}}$. So the total cost of edges added at line 11 (for adding vertex v_i to H) is at most $\lambda c_{v_i v_{j^*}} + s_{v_{j^*}}$. By definition of j^*: $c_{v_i v_{j^*}} \le 4\mu_i + 2\mu_{j^*}$. Noting that $s_{v_{j^*}} \le s_{v_i}$ (and therefore $\mu_{j^*} \le \mu_i$), the total cost of edges added for v_i is at most $6\lambda\mu_i + s_{v_i} \le 7s_{v_i}$. We charge this cost to v_i.

Case 2: Now consider an active node $v_i \in \mathcal{S}$ for which we add all the vertices in $S_{v_i} \cup \{v_i\}$ to H and make a $(\lambda{+}1)$-clique on these vertices in line 9. For any two vertices $x, y \in S_{v_i} \cup \{v_i\}$: $c_{xy} \le c_{v_i x} + c_{v_i y}$. Since each vertex x is incident with λ edges in this clique, the total cost of the edges of the clique is at most $\lambda \sum_{y \in S_{v_i}} c_{v_i y} = \lambda s_{v_i}$. Now we show how to pay for this cost by charging the vertices in $S_{v_i} \cup \{v_i\}$.

Sub-case 2a: In this sub-case we assume $i = 1$, i.e. the case of $v_1 = r$. In this case, we are adding λ new vertices in S_{v_1} in line 9 at a cost of at most λs_{v_1}. Assume that $v'_1, v'_2, v'_3, \ldots, v'_k$ are the vertices of the optimum solution where $v'_1 = v_1 = r$. Without loss of generality, and using the assumption that $v'_1 = r$ has the smallest $s_{v'_i}$ value among all the nodes in the optimum solution we assume that $s_r \le s_{v'_2} \le \ldots \le s_{v'_k}$. Using the first lower bound given for OPT in the previous section: $\sum_{1 \le i \le k} s_{v'_i} \le 2\text{OPT}$. Thus, using the fact that $k \ge \lambda + 1$: $\lambda s_r < \sum_{1 \le i \le k} s_{v'_i} \le 2\text{OPT}$. So if we charge the root by 2OPT, we can pay for the cost of edges added for the ball of the root in line 9.

Sub-case 2b: In this sub-case we consider other active nodes $v_i \in \mathcal{S}$ with $i \ge 2$ that are added (in line 9). As mentioned earlier, the total cost of the edges of the clique added in line 9 is at most λs_{v_i}. We will charge this cost to the vertices in B_{v_i}. Using Lemma 3, there are at least $\lambda/2$ nodes in B_{v_i} (the core of S_{v_i}) that do not belong to any B_{v_j} for $v_j \in \mathcal{S}$ with $j \ne i$. So the vertices in B_{v_i} that we charge the total cost of λs_{v_i} to, are not charged any cost in a different core. We can pay for this (at most) λs_{v_i} cost if we charge every node in B_{v_i} by $2s_{v_i}$ and that is what we do. Remember that we want to ensure that for every vertex $v_j \in H \cap T_1$, the total charge for it is no more than $7s_{v_j}$. For every vertex $v_j \in B_{v_i} \cap T_1$ that we charge $2s_{v_i}$, if $j \ge i$ then clearly the charge $2s_{v_i}$ assigned to v_j is no more than $7s_{v_j}$. But if $j < i$, it means that v_j was de-activated when adding v_i to $\mathcal{S}$; in this case the charge of $2s_{v_i}$ which is assigned to it is upper bounded by $4s_{v_j}$ for the following reason. Note that $c_{v_j v_i} \le 2\mu_i$ (by definition of core B_{v_i}); so using Lemma 9: $\mu_i \le 2\mu_j$, which implies $s_{v_i} \le 2s_{v_j}$. Thus the total charge assigned to v_j is bounded by $2s_{v_i} \le 4s_{v_j}$.

Hence, the charge for every node $v_i \in H \cap T_1$ ($i \ge 2$) analyzed in Cases (1) and (2b) is at most $7s_{v_i}$ and the charge of every node $u \in H - T_1$, which means $u \in S_{v_i}$

for some active node $v_{i\leq i^*} \in \mathcal{S}$, is at most $2s_{v_i}$; i.e. the property we required at the beginning of the lemma holds. We show the total charge is at most $O(\text{OPT})$, which together with the charge of root, which is 2OPT, is still $O(\text{OPT})$.

Let $\tilde{H}$ be $H - (S_{v_1} \cup \{v_1\}) - (S_{v_{i^*}} \cup \{v_{i^*}\})$, and $|\tilde{H}| = \ell$. By definition of i^*: $\ell < k - \frac{\lambda}{7} - 1$. We define a one-to-one function π from the vertices of $\tilde{H}$ to vertices $v_2, \ldots, v_{\ell+1}$ in T_1 in the following way: every vertex $v_i \in \tilde{H} \cap T_1$ is mapped to itself (therefore the charge assigned to v_i is at most $7s_{\pi(v_i)}$). Every other vertex $u \in \tilde{H} - T_1$, which is added in line 9 by adding $S_{v_i} \cup \{v_i\}$ (for some $v_{i\leq i^*-1} \in \mathcal{S}$), is mapped to a vertex $v_{j\geq i} \in T_1$ to which no other vertex of $\tilde{H}$ is mapped to already (in this case the charge assigned to u is at most $2s_{v_i}$, which is at most $2s_{\pi(u)}$). Thus, the total charge assigned to the vertices in $\tilde{H}$ is at most:

$$7 \sum_{2\leq i\leq \ell+1} s_{v_i} \leq 7 \sum_{2\leq i<k-\lambda/7} s_{v_i} \tag{7}$$

where the inequality follows from the fact that $\ell < k - \frac{\lambda}{7} - 1$. Noting that $i^* \leq k - \frac{\lambda}{7}$, the total charge of the nodes in $S_{v_{i^*}} \cup \{v_{i^*}\}$ is at most

$$\lambda s_{v_{i^*}} \leq \lambda s_{v_{k-\lambda/7}} \leq 7 \sum_{k-\lambda/7\leq i\leq k} s_{v_i}. \tag{8}$$

Using Equations (7) and (8), together with the bound of 2OPT for the charge of v_1 in Sub-case 2b, the total charge of the nodes in H is at most

$$2\text{OPT} + 7 \sum_{2\leq i<k-\lambda/7} s_{v_i} + 7 \sum_{k-\lambda/7\leq i\leq k} s_{v_i} \leq 2\text{OPT} + 7 \sum_{1\leq i\leq k} s_{v_i}.$$

Using Equation (2) and the fact $V_1 \subseteq V_0$: $7\sum_{1\leq i\leq k} s_{v_i} \leq 28\rho\text{OPT}$. Thus, the total charge of the nodes in H is at most $(2 + 28\rho)\text{OPT}$.

Theorem 2. *Algorithm 1 (in Figure 1) returns a graph of size at least $k - \frac{\lambda}{7}$ which is λ-edge-connected and has cost at most $(2 + 36\rho)\text{OPT}$.*

Proof. By Lemma 6 and Lemma 7, H is a λ-edge-connected graph with at least $k - \frac{\lambda}{7}$ nodes. Using Lemmas 8 and 10: $c(H) \leq 8\rho\text{OPT} + (2 + 28\rho)\text{OPT} = (2 + 36\rho)\text{OPT}$.

4 From size $k - O(\lambda)$ to size k

As mentioned in the previous section, graph H computed by Algorithm 1 has at least $k - \lambda/7$ vertices and has non-empty intersection with G^* (at least root r belongs to both). If $|H| \geq k$ then we are done. Otherwise, we in this section show how to augment H to have at least k vertices without loosing its edge-connectivity. For every vertex $u \in G \backslash H$, let the distance of u to H, denoted by $d(u, H)$, be the cost of the cheapest edge from u to a vertex in H. We compute two different graphs $H_1 \supseteq H$ and $H_2 \supseteq H$ that are λ-edge-connected and return whichever has at least k nodes and the least cost. The description of the algorithm is given in Figure 3.

1 If $|H| < k$ then compute H_1 and H_2 as described below. If $H_1 = H$ then return H_2. Otherwise, return the one with the smallest cost among H_1 and H_2.
2 H_1:
3 Start with $H_1 = H$.
4 If there is a vertex $u \in G \backslash H$ such that S_u contains at least $\lambda/7$ vertices in $G \backslash H$ then:
5 Find such a vertex $u \in G \backslash H$ with the smallest $s_u + d(u, H)$ value.
6 If $S_u \cap H = \emptyset$ then find the cheapest edge from u to a vertex in H, say v. Add all the edges from u to the λ nearest neighbors of v in H_1.
7 Add all the vertices in $S_u \cup \{u\}$ to H_1 (if they do not already belong to H_1) and make $S_u \cup \{u\}$ a clique by adding all the necessary edges to H_1.
8 H_2:
9 Start with $H_2 = H$.
10 Find a minimum weight matching M of size $k - |H|$ between $G \backslash H$ and H, let Y be the set of vertices in $G \backslash H$ that participate in this matching.
11 Add each $y \in Y$ to H_2 and all the edges between y to the λ closest neighbors of $M(y)$ in H.

Fig. 3. Algorithm 2, which augments H (the result of Algorithm 1) to have at least k vertices

Figure 4 shows how graphs H_1 and H_2 are built from expanding H. To perform line 10 of the algorithm, we can use one of the known minimum weighted (bipartite) matching algorithms or a minimum cost flow algorithms (see [15]). In what follows we show that both H_1 and H_2 are λ-edge-connected and at least one of them has at least k vertices and cost at most $O(\text{OPT})$.

Lemma 11. *If $H_1 \neq H$ then it is λ-edge connected and has at least k vertices. Also, H_2 is λ-edge-connected and has at least k vertices.*

Proof. By the description of Algorithm 2 (in Figure 3), if $H_1 \neq H$ then we have added at least $\lambda/7$ new vertices (belonging to $S_u \cup \{u\}$) to H, so the size will be at least k (given that $|H| \geq k - \lambda/7$). For H_2, we add the vertices of Y and $|Y| = k - |H|$, so $|H_2| = k$. For λ-edge-connectivity, note that H was originally λ-edge-connected. If $H_1 \neq H$ then we have added a vertex u together with S_u. The vertices in $S_u \cup \{u\}$ form a clique, so are λ-edge-connected among themselves. Also, if $S_u \cap H = \emptyset$, u is λ-edge connected to some vertex $v \in H$ by the λ edges added between u and the λ nearest neighbors of v (in H). By transitivity, this implies the λ-edge connectivity of H_1. For the connectivity of H_2, every new vertex $y \in Y$ is connected to at least λ vertices in H which makes it λ-edge connected to all the vertices in H.

Lemma 12. *If H is the solution of Algorithm 1, then the solution of Algorithm 2 has cost at most* $\max\{12\text{OPT} + 3c(H), 13\text{OPT} + 2c(H)\}$.

Proof. We prove this by considering the following two cases.

Case 1: $|G^* \backslash H| \leq \lambda/3$

In this case we show that cost of H_2 is at most $O(\text{OPT})$. Every vertex $u \in G^* \backslash H$

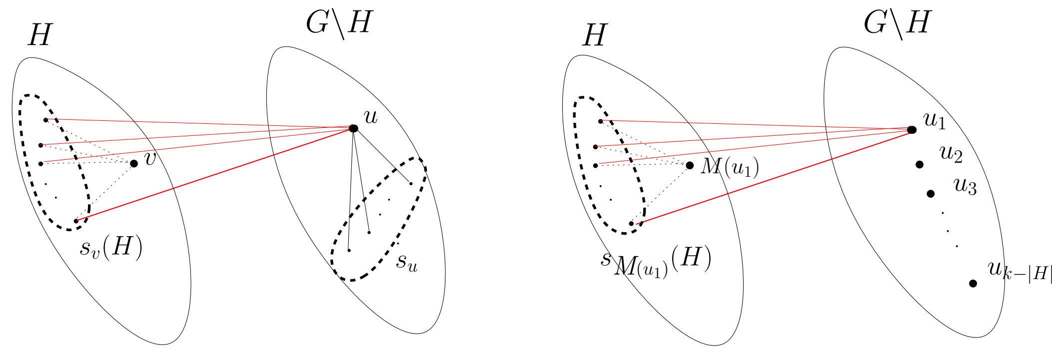

Fig. 4. Constructing H_1 (in the left picture) and H_2 (in the right picture)

is connected, in G^*, to at least $2\lambda/3$ vertices in $G^* \cap H$ and (by a simple averaging argument) the distance of u to at least $\lambda/3$ of them is at most $3s_u(G^*)/\lambda$ (recall that $s_u(G^*)$ is the sum of distances from u to its λ closest neighbors in G^*). Therefore there is a matching $\tilde{M}$ between $G^*\backslash H$ and $G^* \cap H$ such that $d(u, \tilde{M}(u)) \leq 3s_u(G^*)/\lambda$ for every $u \in G^*\backslash H$, and $|\tilde{M}| = |G^*\backslash H| \geq k - |H| = |M|$, where M is the matching we find in the algorithm. Since M is a minimum weight matching:

$$c(M) \leq c(\tilde{M}) \leq \frac{3}{\lambda} \sum_{u \in G^*\backslash H} s_u(G^*) \leq \frac{6\text{OPT}}{\lambda}, \tag{9}$$

where we use the lower bound of $\sum_{u \in G^*\backslash H} s_u(G^*) \leq \sum_{u \in G^*} s_u(G^*) \leq 2\text{OPT}$. Connecting u to the λ nearest neighbors of $M(u)$ costs at most $\lambda \cdot d(u, M(u)) + s_{M(u)}(H)$, by triangle inequality. Thus, if $c(M)$ and $c(H)$ denote the cost of edges of matching M and graph H, respectively, the total cost of the edges added in line 11 is at most:

$$\begin{aligned}\sum_{u \in G^*\backslash H} (\lambda \cdot d(u, M(u)) + s_{M(u)}(H)) &\leq \lambda \cdot c(M) + 2c(H) \\ &\leq 6\text{OPT} + 2c(H)\end{aligned}$$

where the first inequality follows from the fact that $\cup_{u \in G^*\backslash H} S_{M(u)}(H)$ counts every edge of H at most twice and, therefore, its cost is at most twice as much as the cost of H. Thus, $c(H_2) \leq 3c(H) + 6\text{OPT}$.

Case 2: $|G^*\backslash H| > \lambda/3$

In this case (again by an averaging argument) there is a set Y of $\lambda/6$ vertices in $G^*\backslash H$ such that $s_y \leq 12\text{OPT}/\lambda$ for each $y \in Y$ (Otherwise, the remaining at least $\lambda/6$ vertices would have total s value more than $\lambda/6 \times 12\text{OPT}/\lambda \geq 2OPT$ which is a contradiction). If every vertex in Y is connected, in G^*, to at least $2\lambda/6$ vertices in $G^* \cap H$ then, with an argument similar to the previous case, we can upper bound the cost of H_2 by:

$$
\begin{aligned}
c(H_2) &\leq c(H) + \lambda \sum_{y \in Y} d(y, M(y)) + 2c(H) \\
&\leq \lambda \sum_{y \in Y} \frac{6 s_y}{\lambda} + 3c(H) \\
&= 6 \sum_{y \in Y} s_y + 3c(H) \\
&\leq 6 \cdot \frac{\lambda}{6} \cdot \frac{12\textsc{opt}}{\lambda} + 3c(H) \\
&= 12\textsc{opt} + 3c(H),
\end{aligned}
$$

where the 2nd inequality follows from the fact that Y has $\lambda/6$ nodes and each $y \in Y$ has at least $\lambda/3$ neighbors in $G^* \cap H$.

Otherwise, let $y \in Y$ be a vertex such that S_y has more than $4\lambda/6 > \lambda/7$ vertices in $G^* \backslash H$. In this case we show that $H_1 \neq H$ and $c(H_1) = O(\textsc{opt})$. First we claim that the clique on vertices $S_y \cup \{y\}$ costs at most $\lambda \cdot s_y \leq 12\textsc{opt}$. The reason is, for each pair $u, v \in S_y$: $c_{uv} \leq c_{uy} + c_{yv}$. Since edge uy participates in λ such inequalities, we get that the total cost of the clique is at most λs_y, and because each vertex $y \in Y$ has $s_y \leq 12\textsc{opt}/\lambda$, we get the upper bound of $12\textsc{opt}$. Furthermore, since $H \cap G^*$ is non-empty (at least $r \in H$), y must have distance at most $\textsc{opt}/\lambda$ to some vertex $v \in H$ (because there are at least λ edge disjoint paths between y and v and the cost of each is at least c_{yv} by the triangle inequality). Thus, with an argument similar to the previous case, it costs at most $\lambda c_{yv} + s_v(H) \leq \textsc{opt} + c(H)$ to connect y to the λ nearest neighbors of v in H. So the cost of building the clique on $S_y \cup \{y\}$ and connecting y to the λ nearest neighbors of v in H is at most $13\textsc{opt} + c(H)$. This implies that H_1 will have at least k vertices and costs at most $13\textsc{opt} + 2c(H)$

Combining the two Algorithms 1 (in Figure 1) and 2 (in Figure 3), and using Lemmas 12 and 11, and Theorem 2 we have an algorithm that returns a λ-edge connected subgraph on at least k vertices with cost at most $\max\{12\textsc{opt} + 3c(H), 13\textsc{opt} + 2c(H)\} \leq 3(2 + 36\rho)\textsc{opt} + 12\textsc{opt} = (18 + 108\rho)\textsc{opt}$. Thus, we have the following theorem which is essentially Theorem 1:

Theorem 3. *There is a polynomial time algorithm for the (k, λ)-subgraph problem on graphs with metric edge costs which has approximation factor at most $18 + 108\rho$, with $\rho \leq 4$ being the best approximation factor for the k-Steiner tree problem.*

5 Concluding Remarks

In this paper, we proved that the (k, λ)-subgraph problem with metric costs has a polynomial time $O(1)$-approximation algorithm. However, the approximation ratio of our algorithm is relatively large (between 400 and 500). Although it is very likely that one can achieve an approximation ratio close to 100 using the same algorithm by fine tuning the parameters, getting a small constant factor approximation seems challenging, for general values of λ.

For general cost functions, the only known results on this problem (that we are aware of) are the papers [12, 5] which prove that the $(k, 2)$-subgraph problem (on general graphs) has $O(\log^2 n)$-approximation, even if we require 2-node-connectivity in the solution (instead of 2-edge-connectivity). Even for the special case of $\lambda = 3$, there is no known non-trivial approximation algorithm or lower bound (hardness result).

Acknowledgments: The second author thanks Joseph Cheriyan for some initial discussions on the problem.

References

1. B. Awerbuch, Y. Azar, A. Blum and S. Vempala, *New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen*, SIAM J. Computing 28(1):254-262, 1999.
2. A. Blum, S. Chawla, D. Karger, T. Lane, A. Meyerson, and M. Minkoff, *Approximation Algorithms for Orienteering and Discounted-Reward TSP*, SIAM J. on Computing 28(1):254-262, 1999. Earlier version in Proc of STOC 1995.
3. A. Blum, R. Ravi, and S. Vempala, *A constant-factor approximation algorithm for the k-MST problem*, J. Comput. Syst. Sci. 58(1): 101-108, 1999. Earlier in Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC '96), pp. 442-448.
4. J. Cheriyan and A Vetta, *Approximation algorithms for network design with metric costs*, In Proceedings of the thirty-seventh annual ACM symposium on Theory of computing (STOC) 2005, 167–175.
5. C. Chekuri and N. Korula, *Min-Cost 2-Connected Subgraphs with k Terminals*, manuscript 2008, available at http://arxiv.org/abs/0802.2528.
6. C. Chekuri, N. Korula, and M. Pál, *Improved Algorithms for Orienteering and Related Problems*, In Proc of ACM-SIAM SODA, 2008.
7. F. Chudak, T. Roughgarden, and D. P. Williamson, *Approximate MSTs and Steiner trees via the primal-dual method and Lagrangean relaxation*, Math. Program. 100(2):411-421, 2004.
8. U. Feige, G. Kortsarz and D. Peleg, *The dense k-subgraph problem*, Algorithmica, 29(3): 410-421, 2001. Preliminary version in the Proc. 34-th IEEE Symp. on Foundations of Computer Science (FOCS) pp 692-701, 1993.
9. N. Garg, *A 3-Approximation for the minim tree spanning k vertices*, In Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS), 302-309, 1996.
10. N. Garg, *Saving an epsilon: a 2-approximation for the k-MST problem in graphs*, In Proceedings of the thirty-seventh annual ACM symposium on Theory of computing (STOC), 396 - 402, 2005.
11. K. Jain, *A factor 2 approximation algorithm for the generalized Steiner network problem*, Combinatorica, 21:39-60, 2001.
12. L. Lau, S. Naor, M. Salavatipour, and M. Singh, *Survivable Network Design with Degree or Order Constraints*, Submitted to SIAM J. on Computing. Earlier version in Proceedings of the thirty-nineth annual ACM symposium on Theory of computing (STOC), 2007.
13. S. Rajagopalan and V. Vazirani, *Logarithmic approximation of minimum weight k trees*, unpublished manuscript, 1995.
14. R. Ravi, R. Sundaram, M.V. Marathe, D.J. Rosenkrants, and S.S. Ravi, *Spanning trees short or small*, SIAM Journal on Discrete Mathematics, 9(2):178-200, 1996.
15. A. Schrijver, Combinatorial Optimization, Springer, 2003.