Approximation Algorithms for the Capacitated Minimum Spanning Tree Problem and its Variants in Network Design*

Raja Jothi and Balaji Raghavachari

University of Texas at Dallas, Richardson, TX 75083.
E-mail: {raja,rbk}@utdallas.edu

Abstract. Given an undirected graph $G = (V, E)$ with non-negative costs on its edges, a root node $r \in V$, a set of demands $D \subseteq V$ with demand $v \in D$ wishing to route $w(v)$ units of flow (weight) to r, and a positive number k, the *Capacitated Minimum Steiner Tree* (CMStT) problem asks for a minimum Steiner tree, rooted at r, spanning the vertices in $D \cup \{r\}$, in which the sum of the vertex weights in every subtree hanging off r is at most k. When $D = V$, this problem is known as the *Capacitated Minimum Spanning Tree* (CMST) problem. Both CMStT and CMST problems are NP-hard. In this paper, we present approximation algorithms for these problems and several of their variants in network design. Our main results are the following.

- We give a $(\gamma\rho_{ST} + 2)$-approximation algorithm for the CMStT problem, where γ is the *inverse Steiner ratio* and ρ_{ST} is the best achievable approximation ratio for the Steiner tree problem. Our ratio improves the current best ratio of $2\rho_{ST} + 2$ for this problem.
- In particular, we obtain $(\gamma + 2)$-approximation ratio for the CMST problem, which is an improvement over the current best ratio of 4 for this problem. For points in Euclidean and Rectilinear planes, our result translates into ratios of 3.1548 and 3.5, respectively.
- For instances in the plane, under the L_p norm, with the vertices in D having uniform weights, we give a non-trivial $(\frac{7}{5}\rho_{ST} + \frac{3}{2})$-approximation algorithm for the CMStT problem. This translates into a ratio of 2.9 for the CMST problem with uniform vertex weights in the L_p metric plane. Our ratio of 2.9 solves the long standing open problem of obtaining a ratio any better than 3 for this case.

1 Introduction

In this paper, we consider the *Capacitated Minimum Steiner Tree* (CMStT) problem, one of the extensively-studied network design problem in telecommunications. The CMStT problem can formally be defined as follows.

CMStT: Given an undirected graph $G = (V, E)$ with non-negative costs on its edges, a root node $r \in V$, a set of demands $D \subseteq V$ with with demand $v \in D$ wishing to route $w(v)$ units of flow (weight) to r, and a positive number k, the *Capacitated minimum Steiner tree* (CMStT) problem asks for a minimum Steiner tree, rooted at r, spanning the vertices in $D \cup \{r\}$, in which the sum of the vertex weights in every subtree hanging off r is at most k.

---

* Full version of the paper available at http://www.utdallas.edu/~raja/Pub/cmst.ps.
Research supported in part by the NSF under grant CCR-9820902.

The capacity constraint k must be at least as much as the largest vertex weight for the CMStTproblem to be feasible. The CMStT problem is NP-hard as the case with $k = \infty$ is the minimum Steiner tree problem, which is NP-hard. When $D = V$, the CMStT problem is the well-known *Capacitated Minimum Spanning Tree* (CMST) problem. The CMST problem is NP-hard [3, 8] even for the case when vertices have unit weights and $k = 3$. The problem is polynomial-time solvable if all vertices have unit weights and $k = 2$ [3]. The problem can also be solved in polynomial time if vertices have 0,1 weights and $k = 1$, but remains NP-hard if vertices have 0,1 weights, $k = 2$ and all edge lengths are 0 or 1 [3]. Even the geometric version of the problem, in which the edge costs are defined to be the Euclidean distance between the vertices they connect, remains NP-hard.

The CMST problem has been well studied in Computer Science and Operations Research for the past 40 years. Numerous heuristics and exact algorithms have been proposed (see full version of paper `http://www.utdallas.edu/~raja/Pub/cmst.ps` for survey on the literature). Although most of the heuristics solve several well known instances close to optimum, they do not provide any approximation guarantee on the quality of the solutions obtained. Exact procedures are limited to solving smaller instances because of their exponential running time. In this paper, we present improved approximation algorithms for the CMStTand CMST problems and their variants.

1.1 Previous results

For the CMST problem with uniform vertex weights, Gavish and Altinkemer [4] presented a modified *parallel savings algorithm* (PSA) with approximation ratio $4 - 1/(2^{\lceil \log k \rceil - 1})$. In 1988, Altinkemer and Gavish [1] gave improved approximation algorithms with ratios $3 - \frac{2}{k}$ and 4 for the uniform and non-uniform vertex weight cases, respectively. They construct a traveling salesman tour (TSP) with length of at most twice the minimum spanning tree (MST), and partition the tour into segments (subtrees) of weight at most k. Partitioned subtrees are then connected to the root vertex using direct edges. Hassin, Ravi and Salman [6] presented algorithms for the 1-cable *Single-Sink Buy-at-Bulk* problem. The algorithms in [1] and [6] can be used to obtain ratios of $2\rho_{ST} + 1$ and $2\rho_{ST} + 2$ for the respective uniform and non-uniform vertex weight CMStT problems.

1.2 Our contributions

In this paper, we solve the long-standing open problem of obtaining better approximation ratios for the CMST problem. Our main results are the following.

- We give a $(\gamma\rho_{ST} + 2)$-approximation algorithm for the CMStT problem, where γ is the *inverse Steiner ratio*[1] and ρ_{ST} is the best achievable approximation ratio for the Steiner tree problem. Our ratio improves the current best ratio of $2\rho_{ST} + 2$ for this problem.
- In particular, we obtain $(\gamma + 2)$-approximation ratio for the CMST problem, which is an improvement over the current best ratio of 4 for this problem.

[1] The Steiner ratio is the maximum ratio of the costs of the minimum cost Steiner tree versus the minimum cost spanning tree for the same instance.

For points in Euclidean and Rectilinear planes, our result translates into ratios of 3.1548 and 3.5, respectively.
- For instances in the plane, under the L_p norm, with the vertices in D having uniform weights, we give a non-trivial $(\frac{7}{5}\rho_{ST} + \frac{3}{2})$-approximation algorithm for the CMStT problem. This translates into a ratio of 2.9 for the CMST problem with uniform vertex weights in the L_p metric plane. Our ratio of 2.9 solves the long standing open problem of obtaining a ratio any better than 3 for this case.
- For the CMST problem, we show how to obtain a 2-approximation for graphs in metric spaces with unit vertex weights and $k = 3, 4$.
- For the *budgeted* CMST problem, in which the weights of the subtrees hanging off r could be up to αk instead of k ($\alpha \geq 1$), we obtain a ratio of $\gamma + \frac{2}{\alpha}$.

Of the above results, the 2.9-approximation result for the CMST problem is of most significance. This is due to the fact that obtaining a ratio any better than 3 for graphs defined in the Euclidean plane (with uniform vertex weights) is not straightforward. There are several ways one can obtain a ratio of 3 for this problem ([1], modified algorithm of [6], our algorithm in Section 3.1). But the question was whether one can ever obtain a ratio smaller than $3 - o(1)$ for this version of the CMST problem. We present an example (in Section 4), which shows that, with the currently available lower bounds for the CMST problem, it is not possible to obtain an approximation ratio any better than 2. We introduce a novel concept of *X-trees* to overcome the difficulties in obtaining a ratio better than 3.

Achieving ratios better than 3 and 4 for the uniform and non-uniform vertex weighted CMST problems, respectively, has been an open problem for 15 years now. One major reason for the difficulty in finding better approximations is that there is no non-trivial lower bound for an optimal solution. There are instances for which the cost of an optimal solution can be as much as $\Omega(n/k)$ times than that of an MST. Inability to find better lower bounds has greatly impeded the process of finding better approximation ratios for this problem. Even though we were not able to completely eliminate the use of MST as a lower bound, we found ways to exploit its geometric structure, thereby achieving better performance ratios. Unlike the algorithms in [1], in which the MST lower bound contributes a factor of 2 to the final ratio, our algorithms minimizes the use of MST lower bound, thereby achieving better ratios.

2 Preliminaries

Let $|uv|$ denote the distance between vertices u and v. Length of an edge is also its cost. The terms points, nodes and vertices will be used interchangeably in this paper. For a given k, let OPT and APP denote optimal and approximate solutions, respectively, and let C_{opt} and C_{app} denote their respective costs. Let C_{mst} and C_{ST} denote the costs of an MST and an optimal Steiner tree, respectively.

In a rooted tree T, let T_v denote the subtree rooted at v. Let C_T denote the cost of tree T. Let $w(v)$ denote the weight of vertex v, and let $w(T_v)$ denote the sum of vertex weights in the subtree rooted at v. For the CMStT problem, the weight of a vertex the is not in D is assumed to be 0. By weight of a subtree, we mean the sum of the vertex weights in that subtree. We call as *spokes*, the edges incident on r of a CMStT. By *level* of a vertex, in a tree T rooted at r, we mean the number of tree edges on its path to r (also known as depth).

By "metric completion" of a given graph (whose edges obey triangle inequality) we refer to a complete graph. Throughout this paper, without loss of generality, we assume that the metric completion of the input graph is available, and that the weights of vertices in $V \backslash D$ is zero. All our algorithms in this paper are for the CMStT problem—a generalization of the CMST problem. The following lemma gives a lower bound on the cost of an optimal solution.

Lemma 1. $C_{opt} \geq \frac{1}{k} \sum_{v \in V} w(v)|rv|$.

3 CMStT algorithms

We first construct a ρ_{ST}-approximate Steiner tree T spanning all the vertices in $D \cup \{r\}$, and then root T at the root vertex r. Next, we prune subtrees of weight at most k in a bottom-up fashion, and add edges to connect r to the closest node in each of the pruned subtrees. In simple terms, we basically cut T into subtrees of weight at most k and connect them to the root vertex.

It is safe to assume that nodes have integer weights. The assumption is not restrictive as any CMStT problem with rational weights can be converted to an equivalent problem with integer node weights. The optimal solution for the scaled problem is identical to that of the original problem [1].

Since our algorithm for the uniform vertex weights case is quite complex, we first present the algorithm for the general case (non-uniform vertex weights), which will help in an easier understanding of our algorithm for the uniform vertex weights case. Note that all our algorithms start with a ρ_{ST}-approximate Steiner tree of constant degree. Before we proceed to the algorithms, we present the following important lemma.

Lemma 2. *For a given graph $G = (V, E)$, a set of demands $D \subseteq V$, $r \in V$, and a k, let T_f be a feasible* CMStT *and let $t_1, t_2, \ldots, t_m$ be the subtrees hanging off r in T_f. Let $w(t_q)$ be the weight of a minimum weight subtree t_q hanging off r. For all i, if the cost of the edge connecting subtree t_i to r is minimal, then the cost C_{sp} of all the edges incident on r (spokes) in T_f is at most $k/w(t_q)$ times the cost of an optimal solution.*

Proof. Let Γ be the set of vertices in $t_1, \ldots, t_m$. For all i, let v_i be the vertex in t_i through which t_i is connected to r. Recall that edge rv_i is a spoke, and that it is a minimal cost edge crossing the cut between r and t_i. Then,

$$|rv_i| \leq \frac{\sum_{v \in t_i} w(v)|rv|}{\sum_{v \in t_i} w(v)} \leq \frac{\sum_{v \in t_i} w(v)|rv|}{w(t_q)}.$$

The cost of the all the edges incident on r is given by

$$C_{sp} = \sum_{i=1}^{m} |rv_i| \leq \frac{\sum_{v \in \Gamma} w(v)|rv|}{w(t_q)} = \frac{k}{w(t_q)} \times \frac{\sum_{v \in D} w(v)|rv|}{k}$$

$$\leq \frac{k}{w(t_q)} \times C_{opt}. \quad \text{(by Lemma 1)}$$

3.1 Non-uniform vertex weights

The algorithm given below outputs a feasible CMStT for a given instance, whose edges obey triangle inequality. Note that during the course of the algorithm, we replace real vertices with *dummy* vertices of zero weight. These dummy vertices can be thought of as Steiner points. In the algorithm, we use c_i to denote the subtree rooted at child i of vertex v, and p_v to denote v's parent.

Algorithm CMStT-NonUniform
Input: ρ_{ST}-approximate Steiner tree T rooted at r.

1. Choose a maximum level vertex $v \neq r$ such that $w(T_v) \geq k$. If there exists no such vertex then STOP.
2. If $w(T_v) = k$, then replace the Steiner tree edges incident on the vertices in T_v with edges of a minimal cost tree τ spanning only the vertices in $T_v \cap D$. Add a new edge connecting r to the closest vertex in τ.
3. Else if, for some i, $w(c_i) \geq k/2$, then replace the Steiner tree edges incident on the vertices in c_i with edges of a minimal cost tree τ spanning only the vertices in $c_i \cap D$. Add a new edge connecting r to the closest vertex in τ.
4. Else if $\sum w(c_i) < k/2$, which means $w(v) > k/2$, then replace v with a dummy vertex. In the final solution, add v and an edge connecting v to r.
5. Else collect a subset s of subtrees, each of which is rooted at one of v's children, such that $k/2 \leq w(s) \leq k$. Replace the Steiner tree edges incident on the vertices in s with edges of a minimal cost tree τ spanning only the vertices in $s \cap D$. Add a new edge connecting r to the closest vertex in τ.
6. Go to step 1.

It can be verified that our algorithm outputs a feasible CMStT for a given k.

Theorem 1. *For a given* CMStT *instance, Algorithm* CMStT-NonUniform *guarantees an approximation ratio of* $(\gamma\rho_{ST} + 2)$.

Proof. We show that the cost of the tree output by Algorithm CMStT-NonUniform is at most $\gamma\rho_{ST} + 2$ times the cost of an optimal CMStT. The input to the algorithm is a ρ_{ST}-approximate Steiner tree T.

It can be easily verified from the algorithm that all the new edges added to the original tree T are either new spokes, or edges that interconnect vertices within the subtrees for which the new spokes were added. In what follows, we account for the cost of the new spokes added to T, followed by the cost of other edges in the final solution output by the algorithm.

A new spoke, incident on a subtree, is added to the original Steiner tree if and only if the weight of the subtree it connects is at least $k/2$. Notice that the algorithm outputs a tree with each subtree hanging off r being disjoint and the weight of every such subtree, for which a new spoke was added, is at least $k/2$. Let C_{sp} be the cost of the spokes that the algorithm "adds" to the Steiner tree. Note that C_{sp} does not include the cost of the spokes that are already in the Steiner tree that was given as input to the algorithm. By Lemma 2, $C_{sp} \leq 2 \times C_{opt}$.

Now, we account for the cost of other edges in the final solution. These edges are either the Steiner tree edges or the edges that replaced the Steiner tree edges. We show that the total cost of all these edges together is at most γ times the cost of the initial Steiner tree. To prove this, it suffices to prove that the cost of the edges that replace the Steiner tree edges is at most γ times the cost of the Steiner tree edges that it replaces. For every subtree formed, notice that the algorithm replaced the edges of the Steiner tree spanning the vertices in that subtree by the edges of an MST spanning only the non-zero weight vertices in that subtree. Since γ was defined to be the inverse Steiner ratio (ratio of the cost of an MST versus the cost of an optimal Steiner tree), by Steiner ratio argument, the cost of the MST spanning only the non-zero weight vertices in a subtree is at most γ times the cost of an optimal Steiner tree spanning the non-zero weight vertices in that subtree. Thus, we can conclude that the cost of the new edges is at most γ times the cost of the ρ_{ST}-approximate Steiner tree edges it replaces. The final cost of the tree output by the algorithm is given by

$$C_{app} \leq C_{sp} + \gamma\rho_{ST}C_{ST} \leq 2C_{opt} + \gamma\rho_{ST}C_{opt} \leq (\gamma\rho_{ST} + 2)C_{opt}.$$

Corollary 1. *For the* CMStT *problem with uniform vertex weights, Algorithm* CMStT-NonUniform *with little modification guarantees a* $(\rho_{ST}+2)$*-approximation ratio.*

Proof. Since we are dealing with uniform vertex weights, without loss of generality, we can assume that they are of unit weight, and thus we can eliminate Step. 4 from Algorithm CMStT-NonUniform. Therefore no dummy vertices are introduced by the algorithm. Once a subtree t of size at least $k/2$ is found, instead of replacing the Steiner tree spanning the vertices in t with a MST spanning the non-zero weight vertices in t, we can just use the edges in t, minus the edge that connects t to its parent, as they are. This eliminates the γ from the final ratio.

Corollary 2. *For the* CMST *problem, Algorithm* CMStT-NonUniform *guarantees a* $(\gamma+2)$*-approximation ratio. In particular, for points in Euclidean and rectilinear planes, it guarantees a ratio of 3.1548 and 3.5, respectively.*

3.2 Uniform vertex weights

Although our algorithm for uniform vertex weights case is similar to Algorithm CMStT-NonUniform at the top-level, contrary to expectations, there are some complicated issues that have to be handled in order to obtain an approximation ratio strictly less than $\rho_{ST} + 2$. From our analysis for the non-uniform vertex weights case, we can see that the weight of the minimum weight subtree hanging off r plays a crucial role in the calculation of the approximation ratio. An obvious heuristic is to prune subtrees of weight as close as possible to k, so that the ratio drops considerably. We will soon see why pruning subtrees of weight strictly greater than $k/2$ is more difficult than pruning subtrees of weight greater than or equal to $k/2$. To overcome the difficulty of pruning subtrees of size strictly greater than $k/2$, we introduce the concept of *X-trees*, which we define below. We call a subtree, T_v, rooted at vertex v as an *X-tree*, x, if all of the following properties are satisfied (follow Fig. 1).

- $k < w(T_v) < \frac{4}{3}k$.
- Weight of no subtree hanging off v is between $\frac{2}{3}k$ and k.
- Sum of the weights of no two subtrees hanging off v is between $\frac{2}{3}k$ and k.
- Sum of the weights of no three subtrees hanging off v is between $\frac{2}{3}k$ and k.

The following proposition follows from the definition of an X-tree.

Proposition 1. *Let v_1 be a maximum level vertex in an X-tree rooted at v such that T_{v_1} is also an X-tree (v_1 could be v itself). If there is no subtree (non-X-tree) of weight greater than k rooted at one of v_1's children, then there always exist two subtrees, t_α and t_β, hanging off v_1 such that $k < w(t_\alpha) + w(t_\beta) < \frac{4}{3}k$ and $\frac{1}{3}k < w(t_\alpha), w(t_\beta) < \frac{2}{3}k$.*

Since the vertices are of uniform weight, without loss of generality, we can assume that they are of unit weight, and scale k accordingly. We also assume that a ρ_{ST}-approximate Steiner tree is given as part of the input. Note that we are trying to solves instances in L_p metric plane. Even though, the maximum nodal degree in a Steiner tree on a plane is 3, we will continue as if it is 5. This is to ensure that our algorithm solves CMST instances on a plane, as the maximum degree of an MST on a L_p plane is 5 [7, 9]. Note that every vertex but root in a tree, with vertex degrees at most 5, has at most 4 children. The algorithm given below finds a feasible CMStT for instances defined on a L_p plane. In the algorithm, we use c_i to denote the subtree rooted at child i of vertex v, and x_j to denote the X-tree rooted at child j of vertex v.

Algorithm CMStT-UNIFORM

Input: ρ_{ST}-approximate Steiner tree T rooted at r

1. Choose a maximum level vertex $v \neq r$ such that T_v is a non-X-tree with $w(T_v) \geq k$. If there exists no such vertex then go to step 11.
2. If $w(T_v) = k$, then add a new edge connecting r to the closest node in T_v. Remove edge vp_v from T.
3. Else if, for some i, $2k/3 \leq w(c_i) \leq k$, then add a new edge connecting r to the closest node in c_i. Remove the edge connecting v to c_i from T.
4. Else if, for some i and j $(i \neq j)$, $2k/3 \leq w(c_i) + w(c_j) \leq k$, then replace edges vc_i and vc_j by a minimal cost edge connecting c_i and c_j, merging the two subtrees into a single tree s. Add a new edge to connect r to the closest node in s.
5. Else if, for some i, j and z $(i \neq j \neq z)$, $2k/3 \leq w(c_i) + w(c_j) + w(c_z) \leq k$, then replace the Steiner tree edges incident on the vertices in c_i, c_j and c_z by a minimal cost tree s spanning all the vertices in c_i, c_j and c_z. Add a new edge to connect r to the closest node in s.
6. Else if, for some i, j and z $(i \neq j \neq z)$, $4k/3 \leq w(c_i) + w(c_j) + w(c_z) \leq 2k$, then do the following.
   Let E_i be the set of edges incident on vertices in c_i. We define E_j (E_z) with respect to c_j (c_z resp.) analogously. Without loss of generality, let E_j be the low-cost edge set among E_i, E_j and E_z. Use DFS on c_j to partition the vertices in c_j into two sets g_1 and g_2 such that the total weight of

vertices in $(c_i \cup g_1) \cap D$ is almost the same as the total weight of vertices in $(c_z \cup g_2) \cap D$. Remove all the edges incident on the vertices in subtrees c_i, c_j and c_z. Construct a minimal cost spanning tree s_1 comprising the vertices in c_i and g_1. Similarly, construct a minimal cost spanning tree s_2 comprising the vertices in c_z and g_2. Add new edges to connect r to the closest nodes in s_1 and s_2.

7. Else if, for some i and j $(i \neq j)$, $2k < w(x_i)+w(x_j) < 8k/3$, do the following. Let v_1 and v_2 be two maximum level vertices in X-trees x_i and x_j respectively, such that T_{v_1} and T_{v_2} are X-trees themselves (see Fig. 2). Recall, by Proposition 1, that there exist two subtrees t_{α_1} and t_{β_1} (t_{α_2} and t_{β_2}), hanging off v_1 (v_2 resp.) such that $k < w(t_{\alpha_1}) + w(t_{\beta_1}) < \frac{4}{3}k$ ($k < w(t_{\alpha_2}) + w(t_{\beta_2}) < \frac{4}{3}k$ resp.).
   Let E_1 represent the set of edges incident on vertices in t_{α_1} (see Fig. 3). Let E_2 represent the set of edges incident on vertices in t_{β_1}. We define E_4 (E_5) with respect to t_{α_2} (t_{β_2} resp.) analogously. Let E_3 be the set of edges incident on vertices in x_i and x_j minus the edges in E_1, E_2, E_4 and E_5.
   Let $G_1 = \{E_1, E_2\}$, $G_2 = \{E_3\}$, and $G_3 = \{E_4, E_5\}$ be three groups. Out of $\{E_1, E_2, E_3, E_4, E_5\}$, double two low-cost edge sets such that they belong to different groups.
   (a) If E_i and E_j were the two edges sets that were doubled, with E_i in G_1 and E_j in G_3, then form three minimal cost subtrees s_1, s_2 and s_3 spanning the vertices in x_i and x_j as follows. Without loss of generality, let E_2 and E_4 be the two low-cost edge sets that were doubled (Fig. 4). Use shortcutting to form s_1 spanning all vertices in t_{α_1} and a subset of vertices in t_{β_1}, form s_3 spanning all vertices in t_{β_2} and a subset of vertices in t_{α_2}, and form s_2 with all the left-over vertices. Remove edge vp_v. Since $k < w(t_{\alpha_1})+w(t_{\beta_1}) < 4k/3$, $k < w(t_{\alpha_2})+w(t_{\beta_2}) < 4k/3$, and $2k \leq w(x_i) + w(x_j) \leq 8k/3$, we can form s_1, s_2 and s_3 of almost equal weight with $2k/3 \leq w(s_1), w(s_2), w(s_3) \leq k$.
   (b) If E_i and E_j were the two edges sets that were doubled, with E_i in G_1 or G_3, and E_j in G_2, then form three minimal cost subtrees s_1, s_2 and s_3 spanning the vertices in x_i and x_j as follows. Without loss of generality, let E_2 and E_3 be the two low-cost edge sets that were doubled (see Fig. 5). From t_{α_2} and t_{β_2} find a vertex w such that $|wr|$ is minimum. Without loss of generality, let t_{α_2} contain w. Use shortcutting to form s_3 spanning all the vertices in x_j minus the vertices in t_{β_2} (see Fig. 6). Note that $k/3 < w(s_3) < k$, as x_j and T_{v_2} are X-trees and $k/3 < w(t_{\alpha_2}), w(t_{\beta_2}) < 2k/3$. Also, since $k/3 < w(t_{\beta_2}) < 2k/3$ and $k < w(x_i) < 4k/3$, subtrees s_1 and s_2 together will be of weight at least $4k/3$ and at most $2k$ (see Fig. 6). Form subtrees s_1 and s_2, using the ideas in Step. 6, such that $2k/3 \leq w(s_1), w(s_2) \leq k$ and $4k/3 \leq w(s_2) + w(s_3) \leq 2k$.
   (c) Add new edges to connect r to the closest nodes in s_1, s_2 and s_3.
8. Else if, for some i and j $(i \neq j)$, $4k/3 \leq w(x_i)+w(c_j) < 2k$, do the following. Let v_1 be a maximum level vertex in X-tree x_i such that T_{v_1} is an X-tree itself. Recall, by Proposition 1, that there exist two subtrees t_{α_1} and t_{β_1}, hanging off v_1 such that $k < w(t_{\alpha_1}) + w(t_{\beta_1}) < \frac{4}{3}k$.

Let E_1 represent the set of edges incident on vertices in t_{α_1}. Let E_2 represent the set of edges incident on vertices in t_{β_1}. Let E_3 be the set of edges incident on vertices in x_i and c_j minus the edges in E_1 and E_2. Form subtrees s_1 and s_2 using the ideas in Step. 6. Add new edges to connect r to the closest nodes in s_1 and s_2.

9. Else if, $4k/3 \leq w(T_v) \leq 2k$, do the following. Let v_1 be a maximum level vertex in X-tree x_i such that T_{v_1} is an X-tree itself. Recall, by Proposition 1, that there exist two subtrees t_{α_1} and t_{β_1}, hanging off v_1 such that $k < w(t_{\alpha_1}) + w(t_{\beta_1}) < \frac{4}{3}k$.
   Let E_1 represent the set of edges incident on vertices in t_{α_1}. Let E_2 represent the set of edges incident on vertices in t_{β_1}. Let E_3 be the set of edges incident on vertices in T_v minus the edges in E_1 and E_2. Form subtrees s_1 and s_2 using the ideas in Step. 6. Add new edges to connect r to the closest nodes in s_1 and s_2.
10. Go to step 1.
11. While there is an X-tree, x, hanging off r, pick a maximum level vertex v_1 in x such that T_{v_1} is also an X-tree. Out of the two subtrees, t_α and t_β, hanging off v_1 (by Proposition 1), without loss of generality, let t_α be the subtree that is closer to r. Remove the edge connecting t_α to v_1, and add a new edge to connect r to the closest node in t_α.

Theorem 2. *For a given* CMStT *instance on a L_p plane, Algorithm* CMStT-Uniform *guarantees an approximation ratio of* $(\frac{7}{5}\rho_{ST} + \frac{3}{2})$.

Proof. We show that the cost of the tree output by Algorithm CMStT-Uniform is at most $(\frac{7}{5}\rho_{ST} + \frac{3}{2})$ times the cost of an optimal CMStT. The input to the algorithm is a ρ_{ST}-approximate Steiner tree T with maximum nodal degree at most 5.

The algorithm "adds" a new spoke to the tree whenever it prunes a subtree of weight at least $2k/3$. There are certain situations (Steps 6 and 11) where the algorithm adds a spoke for pruned subtrees of weight less than $2k/3$. We continue our analysis as if all of the pruned subtrees are of weight at least $2k/3$. This supposition makes the analysis of spoke cost simpler. We will soon justify this supposition (in Cases 5 and 8) in a manner that it does not affect the overall analysis in any way.

The cost of the spokes that were added to the initial Steiner tree is given by $C_{sp} \leq \frac{3}{2} \times C_{opt}$ by an argument analogous to that proving the cost of the spokes that the algorithm adds to the initial Steiner tree in Theorem 1. The above inequality follows immediately from the fact that a new spoke is added to the tree if and only if the subtree it connects to r is of weight at least $2k/3$.

Now, we account for the cost of other edges—all the edges in the final solution, except for the spokes added by the algorithm—in the final solution. We show that the cost of these edges is at most 7/5 times the cost of the Steiner tree edges that the algorithm started with. To prove this, it suffices to show that the cost of the edges that replace the Steiner tree edges is at most 7/5 times the cost of the edges that are replaced. In what follows, we show this by presenting a case-by-case analysis depending upon which step of the algorithm was executed.

Case 1. Steps 1, 2, 3 and 10 do not add any non-spoke edges. The weight of the subtrees for which Steps 1 and 2 adds spokes to the tree is at least $2k/3$.

Case 2. The minimal cost edge connecting c_i and c_j in Step 4 is at most the sum of the two Steiner tree edges that connects c_i and c_j to v (by triangle inequality). Hence no additional cost is involved.

Case 3. In Step 5, the cost of the tree s spanning all the vertices in c_i, c_j and c_z is at most the cost of the tree obtained by doubling the minimum cost edge out of the 3 Steiner tree edges that connect the 3 subtrees to v (see Fig. 7(a)). Hence, we can conclude that the cost of the tree constructed in Step 5 is at most 4/3 times the cost of the Steiner tree edges it replaces.

Case 4. In Step 6, the total cost of the trees s_1 and s_2 spanning all the vertices in c_i, c_j and c_z is at most the total cost of the trees t_1 and t_2 obtained by doubling the minimum cost edge set out of the 3 edge sets that are incident on the vertices in c_i, c_j and c_z, respectively (see Fig. 7(b)). Hence, we can conclude that the cost of the tree constructed in Step 6 is at most 4/3 times the cost of the Steiner tree edges it replaces.

Case 5. Step 7 forms three subtrees s_1, s_2 and s_3 from X-trees x_i and x_j. Since s_1, s_2 and s_3 can be formed by doubling two low-cost edge sets (belonging to two different groups) out of the 5 possible edge sets and shortcutting, we can conclude that the cost of the subtrees s_1, s_2 and s_3 constructed in Step 7 is at most 7/5 times the cost of the Steiner tree edges it replaces.

Accounting for the cost of the spokes added to the Steiner tree requires that each subtree pruned from the Steiner tree is of weight at least $2k/3$. We already proved that the cost of the spokes added to the Steiner tree is at most 3/2 times the cost of an optimal solution. Without loss of generality, the requirement that each pruned subtree is of weight at least $2k/3$ can be interpreted as that of "charging" the spoke cost incident on a subtree to at least $2k/3$ vertices. Notice that this interpretation is valid only if the spoke connecting the subtree to the root is of minimal cost (r is connected to the closest node in the subtree).

Step 7(a) of the algorithm constructs three subtrees s_1, s_2 and s_3, each containing at least $2k/3$ vertices. This ensures that there are at least $2k/3$ vertices to which each of these subtrees can charge their spoke cost. This is not the case with Step 7(b) of the algorithm. As can be seen, subtree s_3 might be of weight less than $2k/3$. Since s_2 contains at least $2k/3$ vertices and $w(s_2)+w(s_3) \geq 4k/3$, and w is a vertex in x_j such that $|wv|$ is minimum, we can always charge the spoke costs of s_2 and s_3 to at least $4k/3$ vertices. Hence, our initial assumption that every pruned subtree is of weight at least $2k/3$ does not affect the analysis since there are at least $2k/3$ vertices for every spoke to charge.

Case 6. Analysis for Steps 8 and 9 are similar to that for Step 6 (Case 4).

Case 8. Step 11 prunes one subtree off X-tree x. The cost of the spoke $|rw|$ to connect t_α to r can be charged to all the vertices in the X-tree x as per the following argument. After disconnecting t_α from the X-tree, we are left with a subtree of $w(x) - w(t_\alpha) < k$ vertices. We do not need a new spoke for the left-over subtree as it is already connected to r using the Steiner tree edge. Hence, even for this case, our initial assumption that every pruned subtree is of weight at least $2k/3$ does not affect the analysis since there are at least $\frac{2}{3}k$ vertices to charge for the spoke added.

In all of the above cases, the cost of the edges that replace the Steiner tree edges is at most $7/5$ times the cost of the Steiner tree edges that the algorithm started with. Thus, the total cost of the tree output by the algorithm is

$$C_{app} \leq \frac{7}{5}\rho_{ST}C_{ST} + \frac{3}{2}\ C_{opt} \leq \Big(\frac{7}{5}\rho_{ST} + \frac{3}{2}\Big)C_{opt}.$$

Corollary 3. *For the* CMST *problem in L_p plane with uniform vertex weights, Algorithm* CMStT-UNIFORM *guarantees a* 2.9*-approximation ratio.*

4 Conclusion

Our ratios are, certainly, not tight. We believe that there is room for improvement, at least for the CMST problem with uniform vertex weights, for which we obtain a ratio of 2.9. The cost of an optimal CMST can be lower bounded by one of the following two quantities: (i) the MST cost and (ii) the spoke lower bound (Lemma 1). Consider Fig. 8, which contains $\alpha^2 k$ points in a unit-spaced grid. MST cost of the points in the grid alone is $\alpha^2 k - 1$. Let k be the distance between r and the closest node in the grid. For capacity constraint k, the cost of an optimal solution would be $2\alpha^2 k - \alpha^2$, whereas the MST cost would be $(\alpha^2+1)k - 1$ and the spoke lower bound would be $\alpha^2 k$. This shows that with the current lower bounds, one cannot get a ratio any better than 2. It should be interesting to see whether we can find a unified lower bound by combining the MST cost and the spoke cost in a some way, instead of just analyzing them separately. We do not see a reason why our of ratio of 2.9 cannot be improved to 2.

References

1. K. Altinkemer and B. Gavish, *Heuristics with constant error guarantees for the design of tree networks*, Management Science **34**, pp. 331-341, 1988.
2. L.R. Esau and K.C. Williams, *On teleprocessing system design*, IBM Sys. Journal, **5**, pp. 142-147, 1966.
3. M.R. Garey and D.S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*, W.H. Freeman, San Francisco, 1979.
4. B. Gavish and K. Altinkemer, *Parallel savings heuristics for the topological design of local access tree networks.*, Proc. IEEE INFOCOM, pp. 130-139, 1986.
5. M. Goemans and D. Williamson, *A General Approximation Technique for Constrained Forest Problems*, SIAM J. on Comput., **24**, pp. 296-317, 1995.
6. R. Hassin, R. Ravi and F.S. Salman, *Approximation algorithms for capacitated network design problems*, APPROX, pp. 167-176, 2000.
7. C. Monma and S. Suri, *Transitions in geometric minimum spanning trees*, Disc. Comput. Geom., **8**, pp. 265-293, 1992.
8. C.H. Papadimitriou, *The complexity of the capacitated tree problem*, Networks, **8**, pp. 217-230, 1978.
9. G. Robins and J. S. Salowe, *Low-degree minimum spanning trees*, Disc. Comput. Geom., **14**, pp. 151-166, 1995.

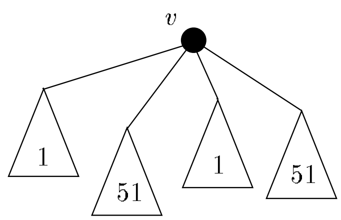

Fig. 1. *An X-tree with $k = 100$.*

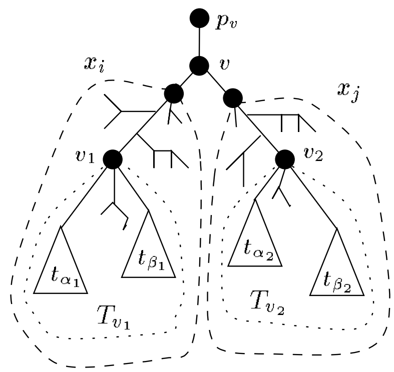

Fig. 2.

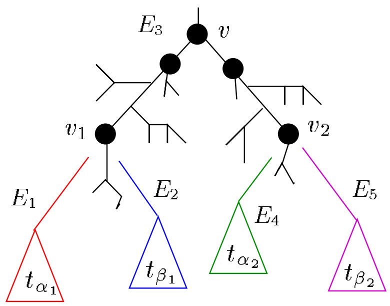

Fig. 3.

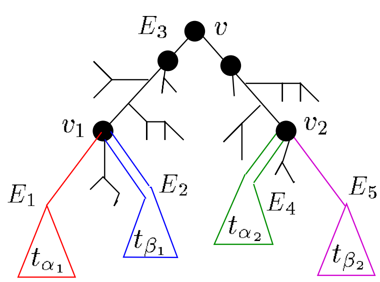

Fig. 4.

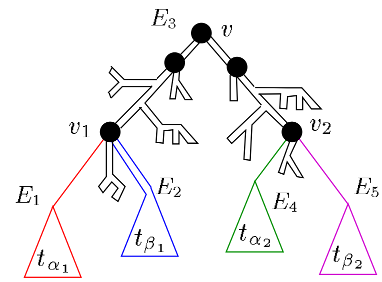

Fig. 5.

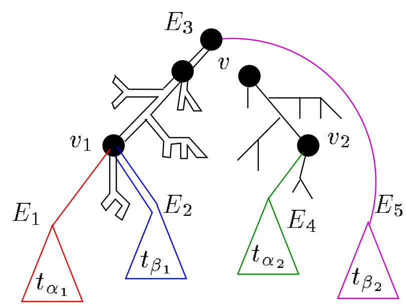

Fig. 6.

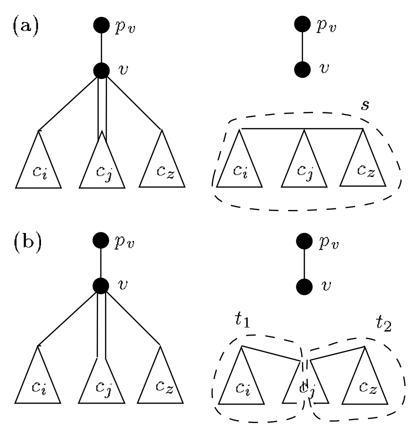

Fig. 7. *Illustration (a)Step 5 (b) Step 6*

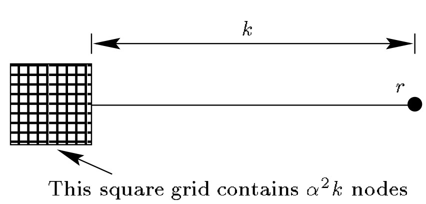

Fig. 8. *A tight example.*