The (K, k)-Capacitated Spanning Tree Problem

Esther M. Arkin[1,⋆], Nili Guttmann-Beck[2], and Refael Hassin[3]

[1] Department of Applied Mathematics and Statistics, State University of New York, Stony Brook, NY 11794-3600, USA
estie@ams.sunysb.edu

[2] Department of Computer Science, The Academic College of Tel-Aviv Yaffo, Yaffo, Israel
becknili@mta.ac.il

[3] School of Mathematical Sciences, Tel-Aviv University, Tel-Aviv 69978, Israel
hassin@post.tau.ac.il

Abstract. This paper considers a generalization of the capacitated spanning tree, in which some of the nodes have capacity K, and the others have capacity $k < K$. We prove that the problem can be approximated within a constant factor, and present better approximations when k is 1 or 2.

1 Introduction

Let $G = (V, E)$ be an undirected graph with nonnegative edge weights $l(e)$ $e \in E$ satisfying the triangle inequality. Let $1 \leq k \leq K$ be given integer *capacities*. Assume that $V = \{r\} \cup V_K \cup V_k$, where r is a *root node*, and V_K and V_k are the sets of nodes having capacity K and k, respectively. In THE (K, k) CAPACITATED SPANNING TREE PROBLEM we want to compute a minimum weight tree rooted at r such that for each $v \in V \backslash \{r\}$ the number of nodes in the subtree rooted at v is no bigger than its capacity.

We are motivated by the following: Nodes of the graph correspond to sensors collecting data that must be transported to a given base-station, the root of the tree. Each sensor forwards all of its data to another (single) node, thus forming a tree representing established data paths. Each node v is also responsible to keep an archive (backup, or data repository) for all of the data at all nodes in the subtree rooted at it (in case the link goes down to a child). The node's capacity represents a storage capacity, saying, e.g., how many nodes' worth of data can be stored at node v. So, we must build trees that obey this capacity constraint. Given costs of the edges, the goal is to build short ("cheap") trees.

The (K, k) CAPACITATED SPANNING TREE PROBLEM is NP-hard as it is a generalization of the CAPACITATED MINIMUM SPANNING TREE PROBLEM where $K = k$ (see [12]).
Our results are as follows:

- For $k = 1$:
  - For $K = 2$ we present a way to find the optimal solution.

⋆ Partially supported by NSF CCF-0729019.

B. Chen (Ed.): AAIM 2010, LNCS 6124, pp. 25–34, 2010.

  - We present a $K-1$ simple approximation algorithm, this algorithm is suitable for small values of K.
  - We also present a 6-approximation algorithm which is suitable for all values of K.
- For $k=2$ we present a 10-approximation algorithm, suitable for all values of K.
- We present a 21-approximation algorithm suitable for all values of (K,k).
- We consider a generalization of the problem where each node $v \in V$ has its capacity k_v, we present an $(2+\alpha)$-approximation algorithm for α which bounds the ratio between the maximal and minimal node capacities.

The Capacitated Minimum Spanning Tree Problem has been studied extensively in the Operations Research literature. It arises in practice in the design of local area telecommunication networks. See [5] for a survey. Various generalizations have also been considered, such as [3] who consider different types of edges, with costs depending on the edge type chosen.

Papadimitriou [12] proved that the capacitated spanning tree problem is NP-hard even with $k=3$. In [1] Altinkemer and Gavish proposed a 3-approximation algorithm. Gavish, Li and Simchi-Levi gave in [6] worst case examples for the 3-approximation algorithm showing that the bound is tight. Gavish in [4] presented the directed version of the problem and gave a new linear integer programming formulation of the problem. This formulation led to a new Lagrangean relaxation procedure. This relaxation was used for deriving tight lower bounds on the optimal solution and heuristics for obtaining approximate solutions.

The most closely related model to ours seems to be the one considered by Gouveia and Lopes [7]. In their model, the children of the root are called *first-level nodes* and they are assigned capacities of, say K, while all the other *second level nodes* have smaller capacities, say $k < K$. The main difference between their model and our (K,k) model is that in our case the capacities are attached to the nodes as part of the input, whereas in their model the capacity of a node depends on its position in the solution. Gouveia and Lopes present heuristics and valid inequalities for their model supported by computational results.

Jothi and Raghavachari,[8], study the capacitated minimum spanning network problem, which asks for a minimum cost spanning network such that the removal of r and its incident edges breaks the network into 2-edge-connected components, each with bounded capacity. They show that this problem is NP-hard, and present a 4-approximation algorithm for graphs satisfying triangle inequality.

Jothi and Raghavachari in [9] study the capacitated minimum Steiner tree problem, looking for a minimum Steiner tree rooted at a specific node, in which the sum of the vertex weights in every subtree is bounded.

Könemann and Ravi present in [10] bicriteria approximation algorithms for the degree-bounded minimum cost spanning tree, a problem relevant to the one studied here, since bounding the out-degree of a node may imply bounds on the subtree descending from this node.

Morsy and Nagamochi study in [11] the Capacitated multicast tree routing problem. In this problem we search for a partition $\{Z_1, Z_2, \ldots, Z_l\}$ of

a given terminal set and a set of subtrees $T_1, T_2, \ldots, T_l$ such that Z_i consists of at most k terminals and each T_i spans $Z_i \cup \{s\}$ (where s is the given source). The objective is to minimize the sum of lengths of the trees $T_1, T_2, \ldots, T_l$. They also propose a $(\frac{3}{2} + \frac{4}{3}\rho)$ approximation, where ρ is the best achievable approximation ratio for the Steiner tree problem.

Deo and Kumar in [2] suggest an iterative refinement technique to compute good suboptimal solutions in a reasonable time even for large instance of problems. They discuss how this technique may be effectively used for the capacitated minimum spanning tree problem.

2 The $(K, 1)$ Problem

In this case the nodes of V_k must be leaves of the tree.

2.1 The $(2, 1)$ Problem

An optimal solution can be obtained through a matching algorithm. We match pairs of nodes such that the root can be matched many times but any other node can be matched only once. Matching node v to the root costs $l(v, r)$. Matching non-root nodes u and v costs $l(u, v) + min\{l(r, u), l(r, v)\}$, for $u, v \in V_2$ and $l(u, v) + l(r, u)$ if $u \in V_2$ and $v \in V_1$.

2.2 The $(K, 1)$ Problem with Small K

When $K \leq 6$ the following simple idea gives a better approximation bound than the general one we present in the next subsection.

Remark 1. It follows easily from the triangle inequality that a star (where all the nodes are directly connected to the root) is a K-approximation.

Lemma 1. *The matching solution described for the case $K = 2$ is a $(K - 1)$-approximation.*

2.3 The $(K, 1)$ Problem with General K

We present now an approximation algorithm for the general $(K, 1)$ problem.

Algorithm $(K, 1)$_Tree

1. Compute a minimum weight matching M from the nodes of V_k to $V_K \cup \{r\}$ such that each node in V_K may be assigned at most $K - 1$ nodes, and all the remaining nodes are assigned to r. The matching cost is the weight of the connecting edges in G. M defines a set of stars in the graph, each star is rooted at one of the nodes in $V_K \cup \{r\}$, and the leaves of this star are the nodes from V_k matched to the root of the star (see Figure 1 top left).

2. For every star rooted at a node from V_K with at least $\frac{K}{2}$ nodes, [By Step 1 the number of nodes in this star is at most K.] connect this star to r using the shortest possible edge (see Figure 1 top right). [Later (in Step 5) we will change this connection to be a feasible connection, as the nodes from V_k must be leaves of the tree.]
3. Compute an MST, T_s, on r and the nodes from V_K that were not connected to r in Step 2 (see Figure 1 middle left). [The optimal solution contains a tree T on $V_K \cup \{r\}$. T is a steiner tree on $V(T_s)$, hence $l(T_s) \leq 2l(T)$.] Figure 1 middle right shows $T_s \cup M$, which includes all the connections made so far.

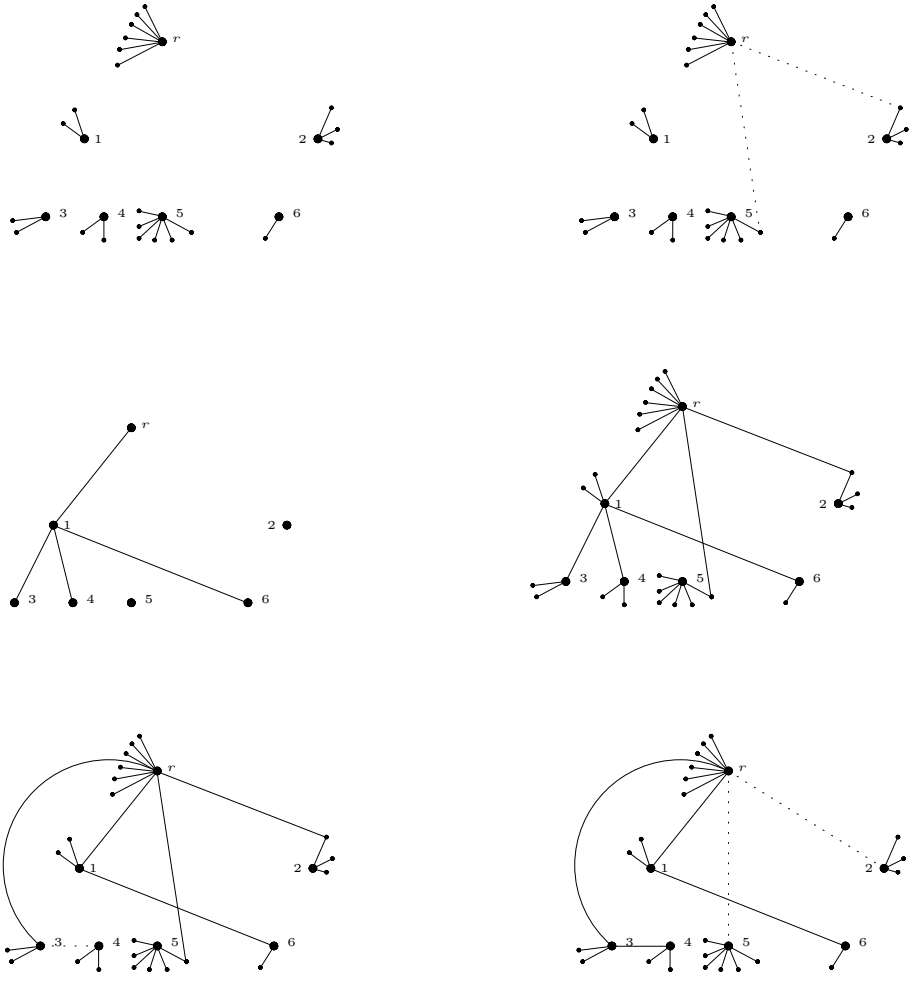

Fig. 1. The different steps in Algorithm $(K, 1)$_Tree with $K = 7, k = 1$

4. Scan T_s from bottom to top and for every node $v \in V_K$ we make the following changes (to guarantee that the subtree rooted at v has at most K nodes):
   - Denote by $y_1, \ldots, y_m \in V_K$ and $u_1, \ldots, u_l \in V_k$ the sons of v, and denote the subtree rooted at y_i by T_i. [Since the tree is scanned from bottom to top $|V(T_i)| \leq K$.]
   - While $\sum_{i=1}^{m} |V(T_i)| \geq \frac{K}{2}$ let p satisfy $\frac{K}{2} \leq \sum_{i=1}^{p} |V(T_i)| \leq K$, disconnect $T_i, (i \in \{1, \ldots, p\})$ from T_s, add the edges $\{(y_q, y_{q+1}) | 1 \leq q \leq p-1\}$, and connect this new tree to r using the shortest possible edge. Renumber the nodes $y_{p+1}, \ldots, y_m$ to $y_1, \ldots, y_{m-p}$ and set $m = m - p$.
     [After the change the number of descendants of v going through V_K nodes is smaller than $\frac{K}{2}$, and the number of sons of v from V_k is smaller than $\frac{K}{2}$, giving that overall v has less than K descendants.]
   - If the subtree rooted at v (including v) contains at least $\frac{K}{2}$ nodes, disconnect this subtree from T_s, and connect to the root using the shortest possible edge.

   (See Figure 1 bottom left)
5. In all cases of connecting a subtree to r by the end edge (r, u) where $u \in V_k$, change this connection to connect the subtree to r using the parent of u. Note that the parent of u is always included in the subtree and is always a node in V_K. (See Figure 1 bottom right.)

Theorem 2. *Denote by* apx *the solution returned by Algorithm* $(K,1)$*_Tree, and let* opt *be the optimal value, then:* $l(\text{apx}) \leq 6\text{opt}$.

3 The $(K, 2)$ Problem

Theorem 3. *Assume* $k = 2$ *and denote by* apx *the solution returned by Algorithm* $(K,1)$*_Tree, and let* opt *be the value of an optimal solution. Then* $\text{apx} \leq 10\text{opt}$.

4 The (K, k) Problem

We now turn to the (K,k) CAPACITATED SPANNING TREE PROBLEM, and consider first a naïve algorithm for the problem: Solve (optimally or approximately) two separate problems. One on $\{r\} \cup V_K$ and the second on $\{r\} \cup V_k$. Then hang the two separate trees on r. This clearly yields a feasible solution.

The following simple example shows that the value of this solution can be as much as $\frac{K-1}{k} + 1$ times the optimal value (even if both separate problems are solved optimally). In this example we assume that $\frac{K-1}{k}$ is integer. The graph has a single node of capacity K, and $K-1$ nodes of capacity k, all at distance 0 from each other, and distance 1 from the root. The first tree is a single edge of length 1, and the second tree includes $\frac{K-1}{k}$ unit length edges from the root.Thus yielding a solution of cost $\frac{K-1}{k} + 1$ while an optimal solution has all nodes in V_k hanging off the single node in V_K, and thus is of cost 1.

In this section we show how to obtain a constant factor approximation algorithm for the (K, k) problem. We first show that any feasible solution F can be transformed into another feasible solution F' with restricted structure, without increasing the weight "too much".

4.1 Ordered Tree

Definition 4. *In an* ordered tree *the capacities of nodes in every path starting at the root are nonincreasing.*

Lemma 5. *Consider a* (K, k) *capacitated spanning tree problem, Let* opt *be the length of an optimal solution. There is a feasible* ordered tree *with length no greater than* 3opt.

Remark 2. There is an instance of the (K, k) capacitated spanning tree problem such that $l(T_o) = 3l(T)$ where T_o is a minimal length feasible ordered tree and T is an optimal solution.

4.2 The Approximation Algorithm

In our algorithm we use the algorithm for the minimum capacitated tree problem described in [1]. This algorithm computes a 3 approximation solution where each subtree is a path.

We offer the following algorithm:
Algorithm (K, k)**_Tree**

1. Compute a minimum spanning tree in the graph induced by $r \cup V_K$, call it T_1. An example of T_1 is shown in Figure 2 top-left.
2. Contract the nodes $r \cup V_K$ into a single node R, and find an approximate capacitated spanning tree on $R \cup V_k$ with capacities k, using the method of [1]. Call this tree T_2. Note that in T_2, each subtree of nodes of V_k hanging on R is a path of length exactly k, except for possibly one shorter path. An example of T_2 is shown in Figure 2 top-right.
3. "Uncontract" the node R in T_2, obtaining a forest in which each connected component is a *rooted-spider*, a node in $r \cup V_K$ with paths of nodes from V_k, all of length k except possibly for one shorter path. Let F_2 denote the forest created from T_2 edges after the 'uncontraction' . Consider Figure 2 middle-left for an example of F_2, where the bold nodes denote $r \cup V_K$.
4. Define a matching problem on a complete bipartite graph $B = (S_1, S_2, S_1 \times S_2)$: In the first side, S_1, of our bipartite graph B, we have a node for each "leg" (path) of a spider. Each node in S_1 should be matched exactly once. In the second side of B we have nodes $S_2 = r \cup V_K$, nodes of V_K have capacity $\lfloor K/k \rfloor - 1$ (meaning that each can be matched at most that many times) and r has unbounded capacity. The cost of matching a node in S_1 to node in S_2 is the length of the edge from a node in the spider leg closest to the destination node.

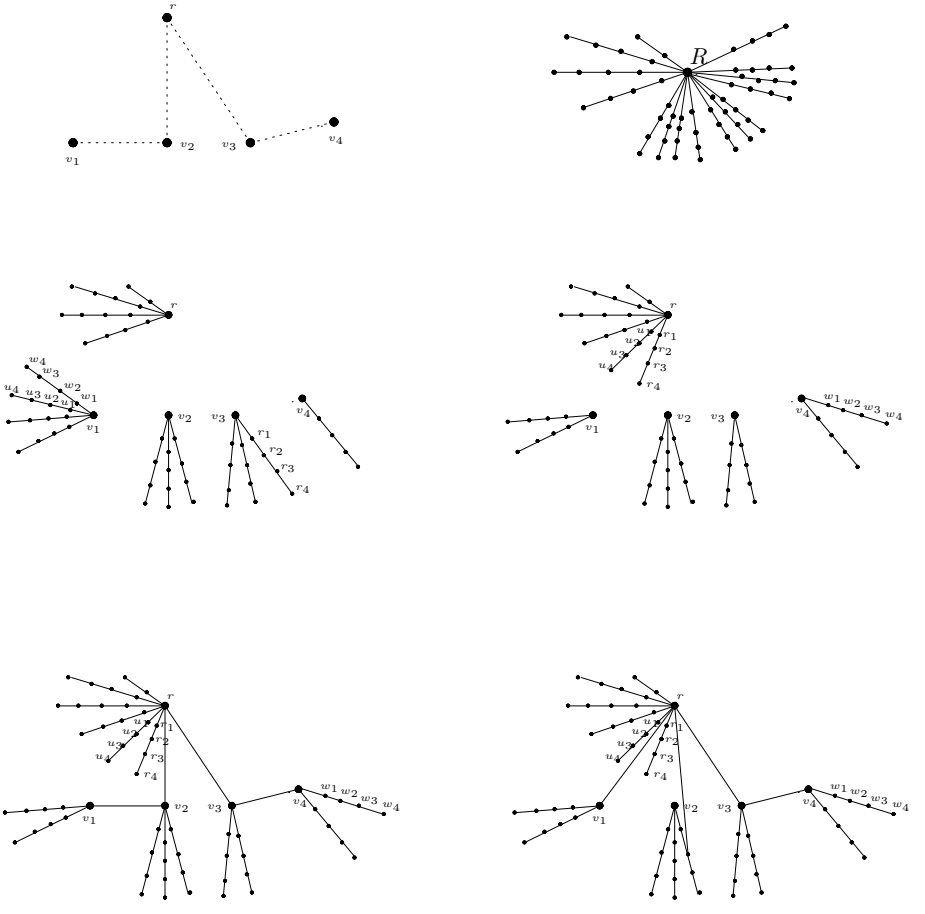

Fig. 2. The different steps in Algorithm (K, k)_Tree with $K = 18$ and $k = 4$

5. Solve the matching problem and change F_2 in the following way: Each spider leg will be attached to the node in $r \cup V_K$ it is assigned to it in the matching problem. The attachment is done by connecting the node in the path closest to the node (i.e., the edge which defines the cost used in the matching). Denote the new forest as F_2'. The forest F_2' is illustrated in Figure 2 middle-right.
6. Consider $T_1 \cup F_2'$ (this graph can be shown in Figure 2 bottom-left). For every $v \in V_K$ with legs $P_1, \ldots, P_l \in V_k$ and $\sum_{i=1}^{l} |V(P_i)| \geq \frac{K}{2} - 1$, disconnect $v \cup \{P_1, \ldots, P_l\}$ from $T_1 \cup F_2'$. [By the way the algorithm works $\sum_{i=1}^{l} |V(P_i)| \leq K - 1$.] Connect this subtree to r using the shortest possible edge. This step is applied to the subtree rooted at v_2 in the bottom figures of Figure 2.
7. The tree T_1 was disconnected in the previous step, reconnect it using only edges between nodes in $V_K \cup \{r\} \setminus \{$nodes that were disconnected in previous

step }. Denote the new tree induced on V_K as T_3. The graph after applying this change to v_1 is shown in Figure 2 bottom-right.

8. Finally, to turn this into a feasible solution for all nodes of V_K, we follow Steps 4,5 of Algorithm $(K,1)$_*Tree* in Section 2.3.

Theorem 6. *Denote by* opt *the value of an optimal solution, and* apx *the solution returned by* Algorithm (K,k)_Tree*, then* $l(\text{apx}) \leq 21\text{opt}$.

Proof: By construction, $l(T_1) \leq 2\text{opt}$ and $l(T_2) \leq 3\text{opt}$, $l(T_3) \leq 2l(T_1) \leq 4\text{opt}$. Next, we bound the length of the edges in the matching. Consider another bipartite graph $B' = (S_1, S_2', E)$, with the same nodes on the first side as B, namely S_1, and nodes on the second side S_2' each corresponding to maximal subtrees induced by V_k in opt$'$ where opt$'$ is the best feasible ordered tree. By Lemma 5 $opt' \leq 3opt$. There is an edge in this bipartite graph between a node in S_1 and a node in S_2' if the two sets of nodes (the leg and the subtree) have at least one node in common (B' is their intersection graph). We now show that B' has a matching in which all nodes of S_1 are matched, using Hall's Theorem. Recall that all legs of apx are of length exactly k, except possibly for one shorter leg, whereas all the subtrees from opt$'$ have length at most k.

In our graph $B' = (S_1, S_2', E)$, we want to show that Hall's condition holds, and therefore there is a matching saturating all nodes of S_1. Let $X \subseteq S_1$. X represents $|X|$ disjoint paths each of length k except possibly one shorter, therefore it represents more than $k(|X|-1)$ nodes of V_k. Call this set $V_k(X)$, and so we have $|V_k(X)| > k(|X|-1)$. Similarly, $N(X)$ also represents disjoint subtrees of nodes in V_k, each subtree contains at most k nodes. Call this set $V_k(N(X))$. Therefore $|V_k(N(X))| \leq k|N(X)|$. By construction $V_k(X) \subseteq V_k(N(X))$ and therefore $k(|X|-1) < |V_k(X)| \leq |V_k(N(X))| \leq k|N(X)|$, resulting in $|X| - 1 < |N(X)|$, or equivalently $|X| \leq |N(X)|$ as required by Hall's Theorem.

Observe that our graph G can be thought of as a subgraph of the graph for which the algorithm finds a matching, simply merge subtrees (nodes of S_2') that are attached to the same node in $r \cup V_K$ to obtain S_2. Thus, the matching in graph G is a feasible solution to the matching found by our approximation algorithm, and our algorithm picked the best such matching. Thus the connections in the last step of our algorithm have total length $l(conn)$ which is at most opt$'$.

When disconnecting subtrees from the tree and connecting them directly to r we add three kinds of edges:

- Connecting brothers (adding edges (y_i, y_{i+1}) to the tree). The sum of lengths of these edges can be bounded by the length of T_3.
- We add edges connecting trees with at least $\frac{K}{2}$ nodes to r. As in the proof of Theorem 2 we can bound the length of the edges with 2opt.
- In the last step we change some of the connecting edges from $(r,u), u \in V_k$ to (r, A_u), $u \in V_k$, where A_u is the closest ancestor of u which is in V_K. By the triangle inequality $l(r, A_u) \leq l(r,u) + l(u, A_u)$, where (u, A_u) is a part of leg added to the tree in the matching. Thus, this step adds at most the length of all the edges from nodes in V_k to their ancestors in V_K, with total length at most $l(conn)$.

Summing all this, $l(\text{apx})$ consists of:

- $l(T_1) \leq 2\text{opt}$, $l(T_2) \leq 3\text{opt}$, $l(T_3) \leq 4\text{opt}$.
- $l(conn) \leq opt' \leq 3\text{opt}$.
- The edges added connecting brothers with length $\leq l(T_3) \leq 4\text{opt}$.
- Edges connecting subtrees to r with length $\leq 2\text{opt}$.
- Changing the connecting edges to ancestors from V_K with maximal length $l(conn) \leq 3\text{opt}$.

Altogether, $l(\text{apx}) \leq 21\text{opt}$. ∎

5 Concluding Remarks: General Capacities

A natural extension of our model allows more than two capacity types. In the extreme case, each node v may have a different capacity, k_v. We leave this generalized problem for future research, and observe that a straightforward extension of the naïve algorithm of Section 4 is possible, as follows: Let k_M be the maximal capacity bound and k_m the minimal capacity bound, and let $\alpha = \frac{k_M}{k_m}$. W.l.o.g., assume that $\frac{|V|}{k_m}$ is an integer, otherwise add an appropriate number of nodes with zero distance from r without affecting the solution.

The algorithm: Compute an MST, T. Double its edges to create an Eulerian cycle. By shortcutting the cycle form a Hamiltonian cycle in the standard way. Partition the cycle into subpaths, each containing k_m nodes. Connect each subpath to r using the shortest possible edge. (This is actually the approximation algorithm suggested in [1] for $k_M = k_m$.)

Theorem 7. *Denote by* opt *the value of the optimal solution and by* apx *the approximation solution, then* $l(\text{apx}) \leq (2 + \alpha)\text{opt}$.

References

1. Altinkemer, K., Gavish, B.: Heuristics with constant error guarantees for the design of tree networks. Management Sci. 34, 331–341 (1988)
2. Deo, N., Kumar, N.: Computation of constrained spanning trees: a unified approach. Lecture Notes in Econ. and Math. Systems, vol. 450, pp. 194–220. Springer, Berlin (1997)
3. Gamvros, I., Raghavan, S., Golden, B.: An evolutionary approach to the multi-level Capacitated Minimum Spanning Tree problem. Technical report (2002)
4. Gavish, B.: Formulations and algorithms for the capacitated minimal directed tree problem. J. Assoc. Comput. Mach. 30, 118–132 (1983)
5. Gavish, B.: Topological design of telecommunication networks - local access design methods. Annals of Operations Research 33, 17–71 (1991)
6. Gavish, B., Li, C.L., Simchi-Levi, D.: Analysis of heuristics for the design of tree networks. Annals of Operations Research 36, 77–86 (1992)
7. Gouveia, L., Lopes, M.J.: Using generalized capacitated trees for designing the topology of local access networks. Telecommunication Systems 7, 315–337 (1997)

8. Jothi, R., Raghavachari, B.: Survivable network design: the capacitated minimum spanning network problem. Inform. Process. Let. 91, 183–190 (2004)
9. Jothi, R., Raghavachari, B.: Approximation algorithms for the capacitated minimum spanning tree problem and its variants in network design. ACM Trans. Algorithms 1, 265–282 (2005)
10. Könemann, J., Ravi, R.: Primal-dual meets local search: approximating MSTs with nonuniform degree bounds. SIAM J. Comput. 34, 763–773 (2005)
11. Morsy, E., Nagamochi, H.: An improved approximation algorithm for capacitated multicast routings in networks. Theoretical Comput. Sci. 390, 81–91 (2008)
12. Papadimitriou, C.H.: The complexity of the capacitated tree problem. Networks 8, 217–230 (1978)