A Simple Approximation Algorithm for the Design of Tree Networks

Raja Jothi*

Abstract

We present a simple 4-approximation algorithm for the extensively-studied *Capacitated Minimum Spanning Tree* problem. The approximation ratio of our algorithm matches the current best ratio of 4, due to Altinkemer and Gavish. Our algorithm is simpler, and easier to analyze when compared to the current best approximation algorithm.

Keywords: Capacitated minimum spanning trees, Approximation algorithms, Network design, Combinatorial optimization.

1 Introduction

A well-studied problem in network design is the *Capacitated Minimum Spanning Tree* (CMST) problem. Consider a given undirected vertex-weighted graph $G = (V, E)$ with non-negative costs on its edges, root node $r \in V$, and capacity constraint k. The cost matrix on the edges is symmetric and satisfies the triangle inequality. The CMST problem asks for a minimum cost spanning tree rooted at r, in which the sum of the vertex weights in every subtree connected to r is at most k. The CMST problem is NP-hard [5].

The CMST problem is one of the extensively studied network design problem in Computer Science and Operations Research. Numerous heuristics and exact algorithms have been proposed over the past 40 years. For a detailed survey on the literature, we refer the reader to [1, 4, 6]. In this paper, we restrict ourselves to approximation algorithms for the CMST problem.

An *α-approximation algorithm* for a combinatorial optimization problem is a polynomial time algorithm, which finds a feasible solution of cost at most α times the cost of an optimal solution. We call α to be the *approximation ratio* of the algorithm. The value of α is greater than or equal to 1 for minimization problems, and less than or equal to 1 for maximization problems. The closer the value of α to 1, the better the approximation ratio.

For the CMST problem, Gavish and Altinkemer [3] were the first ones to present an algorithm with any constant approximation ratio. In [2], they presented a $(3 - \frac{2}{k})$-approximation algorithm for the special case of the CMST problem, in which the vertex weights are all the same. They

*Department of Computer Science, University of Texas at Dallas, P.O. Box 830688; EC 3.1, Richardson, TX 75083, USA. Email: raja@utdallas.edu

used this algorithm as a black-box to present a 4-approximation algorithm for the original CMST problem with arbitrary weights. Henceforth, we will be referring Altinkemer and Gavish by their initials AG.

In this paper, we present a new 4-approximation algorithm for the CMST problem. Our algorithm is simpler and easier to analyze when compared to current best approximation algorithm [2] available for the problem. The approximation ratio guaranteed by our algorithm matches the current best ratio of 4 [2]. We show that there are instances for which the cost of the solution produced by our algorithm is about one third of that produced by AG's algorithm.

2 Preliminaries

We use the terms nodes and vertices interchangeably. Let $|uv|$ denote the cost of an edge connecting vertices u and v. We use r to denote the root vertex. For a given CMST instance, let OPT denote an optimal solution with C^* being its cost, and let C_{mst} denote the cost of a minimum spanning tree (MST).

Lemma 2.1 ([2]) $C^* \geq \max\{C_{mst}, \sum_{v \in V} \frac{w(v)|rv|}{k}\}$.

3 Overview of AG's algorithm

AG's algorithm [2] converts a given CMST instance with arbitrary vertex weights and capacity k into a *new* instance with unit vertex weights and capacity $k/2$, by replacing each vertex $v \in V$ of weight w_v with w_v unit weight vertices. They construct a traveling salesman tour (TSP) visiting all the vertices in the new instance, and partition the tour into segments (subtrees) of weight $k/2$. One can perform an optimal partitioning of the tour, instead of settling for an arbitrary partitioning, by choosing the best partitioning among all possible partitionings. If one chooses to do an optimal partitioning, there may be up to 2 segments that are of weight less than $k/2$.

After partitioning the tour into segments, they connect r to the closest node in each segment. They then show how to convert the solution for the unit weight instance with capacity $k/2$ into a feasible solution for the given instance with capacity k. While their algorithm for the CMST problem with unit vertex weights is straightforward, converting the solution for the unit vertex-weighted CMST instance with capacity $k/2$ into a feasible solution for the arbitrary vertex-weighted CMST instance with capacity k is quite involved. We refer the reader to [2] for more details.

As per their algorithm, the final cost of the solution comprises two components: (i) the cost of the edges in the TSP, and (ii) the cost of the other edges (edges that connect the segments to r). Since a TSP can be constructed by doubling the edges of an MST, for any given instance, the cost of the TSP is at most $2C^*$ (by Lemma 2.1). They show that the cost of connecting each segment to r is at most $2C^*$. This translates into an overall cost of at most $4C^*$.

4 Our Algorithm

Our algorithms improve on the ideas of AG's algorithm. Unlike AG's algorithm, our algorithms can directly be applied to a given instance without having to do any conversions as in [2], thereby reducing the running time. Our algorithm for the CMST problem is given below.

Algorithm PARTITION-TOUR (V^*, r, k)

```
1. Construct a TSP tour $t$ on $V^* \cup \{r\}$.
2. set segmentWeight $= 0$.
3. Starting from $r$, traverse the tour in clockwise direction.
4. Let $p$ and $s$ be the vertices that are lying just before and after $v$ on the tour.
5. While not all vertices in $t$ are traversed and segmentWeight $< k$, visit the next vertex $v$.
   (a) If segmentWeight $+w(v) < k$, set segmentWeight $+= w(v)$.
   (b) Else if $w(v) \geq k/2$, remove $v$ from $t$, and use shortcutting to connect $p$ to $s$. (comment:
       $v$ is a segment by itself)
   (c) Else remove the edge connecting $v$ to $p$ from $t$, and set segmentWeight $= w(v)$.
6. Return the segments.
```

Algorithm CMST-MAIN (V, r, k)

```
1. Construct an MST $T$ spanning $V$.
2. Root $T$ at the root vertex $r$.
3. Let $t_1, t_2, \ldots, t_m$ be the subtrees rooted at the children of $r$.
4. For every subtree $t_i$ connected to $r$ in $T$,
   (a) $S \leftarrow$ PARTITION-TOUR $(\{v|v \in t_i\}, r, k)$.
   (b) For every segment in $S$, if it is not already connected to $r$, connect $r$ to the closest vertex
       in the segment.
```

It can be verified that the above algorithm outputs a feasible solution for a given CMST instance. In what follows, we show that the cost of the solution output by the above algorithm is at most 4 times the cost of an optimal solution.

Theorem 4.1 *Our CMST algorithm guarantees an approximation ratio of 4.*

Proof. The cost of the final solution comprises of two components: (i) the cost of the TSP tours for each subtree hanging off r, and (ii) the cost of connecting the individual segments, of weight at most k, to r. It can be easily seen that the cost of the TSP tours is at most twice the cost of the MST we started with. This is due to the reason that one can easily construct a tour spanning r and the vertices in a subtree rooted at r by just doubling the necessary MST edges and shortcutting the resulting Euler tour.

We now show that the cost of connecting the individual segments to r is at most twice the cost of OPT. As per the partitioning procedure, observe that all, but the last, segments in a tour are guaranteed to be of weight at least $k/2$. Moreover, notice that there is no need to connect the first and the last segment of a tour to r as they are already connected to r by the tour edges, whose cost has already been accounted. Let $s_1, s_2, \ldots, s_m$ be the segments for which the algorithm added edges to connect them to r. Recall that each of these edges connects r to the closest vertex in a segment. Let V' represents the set of all vertices in segments $s_1, s_2, \ldots, s_m$, and let v_i be the vertex in s_i that is connected to r. Let $W(s_i) = \sum_{j \in s_i} w(j)$. Then,

$$\begin{aligned}
|rv_i| &\leq \sum_{j \in s_i} \frac{w(j)|rj|}{W(s_i)} \\
&\leq \sum_{j \in s_i} \frac{w(j)|rj|}{k/2} \\
\sum_{i=1}^{m} |rv_i| &\leq \sum_{i=1}^{m} \sum_{j \in s_i} \frac{w(j)|rj|}{k/2} \\
&\leq \sum_{j \in V} \frac{w(j)|rj|}{k/2} \\
&\leq 2 \times C^* \qquad \text{(by Lemma 2.1)}
\end{aligned}$$

Thus, we can conclude that the overall cost of the tree output by the algorithm is at most 4 times than that of an optimal solution. ∎

In practice, our algorithm may actually perform better than AG's algorithm. Let us consider an instance with unit weight vertices, with r placed at distance M from all the other vertices. Let the non-root vertices be clustered into $k+1$ groups of $k-1$ vertices, with inter-distance between vertices in the same group being ϵ, and inter-distance between vertices in different groups being $2M$. Observe that the edges of the instance under consideration obeys triangle inequality. Figures 1 and 2 illustrate the execution of AG's algorithm and our algorithm, respectively, for the instance with $k = 8$. While the cost of the solution obtained by AG's algorithm is $3(k-1)M + (k-1)(k-2)\epsilon$, the cost of the solution obtained by our algorithm would only be $kM + (k-1)\epsilon$. For a large k and a small ϵ, our algorithm would return a solution of cost about one third of that returned by AG's algorithm. Note that the solution obtained by our algorithm is in fact optimal.

References

[1] A. Amberg, W. Domschke and S. Voß, "Capacitated minimum spanning trees: Algorithms using intelligent search," *Combinatorial Optimization: Theory and Practice* **1**, 9-39, 1996.

[2] K. Altinkemer and B. Gavish, Heuristics with constant error guarantees for the design of tree networks, Management Science (1998) 331-341.

[3] B. Gavish and K. Altinkemer, Parallel savings heuristics for the topological design of local access tree networks, Proc. IEEE INFOCOM (1986) 130-139.

[4] R. Jothi and B. Raghavachari, Revisiting Esau-Williams' algorithm: On the design of local access networks, Proc. 7th INFORMS Telecommunications Conference (2004) 104-107.

[5] C.H. Papadimitriou, The complexity of the capacitated tree problem, Networks 8 (1978) 217-230.

[6] S. Voß, Capacitated minimum spanning trees, in: C.A. Floudas and P.M. Pardalos (Eds), Encyclopedia of Optimization, Kluwer, Boston, 1 (2001) 225–235.

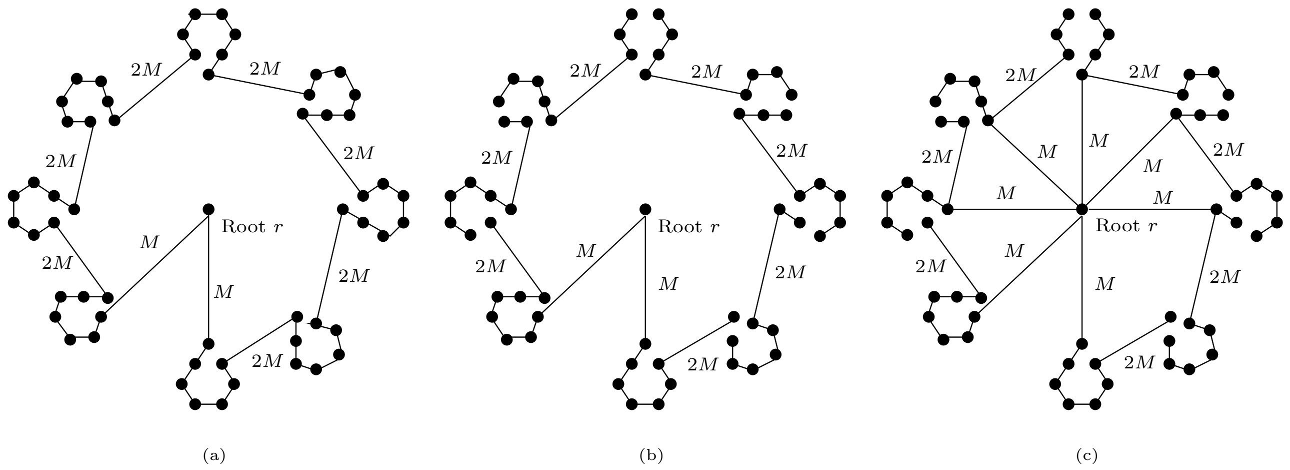

Figure 1: AG's algorithm on the sample instance with $k = 8$. (a) TSP tour spanning all the vertices. (b) Partitioning of the TSP tour. (c) Partitioned subtrees are connected to root r, with the cost of the final tree being $3(k-1)M + (k-1)(k-2)\epsilon$.

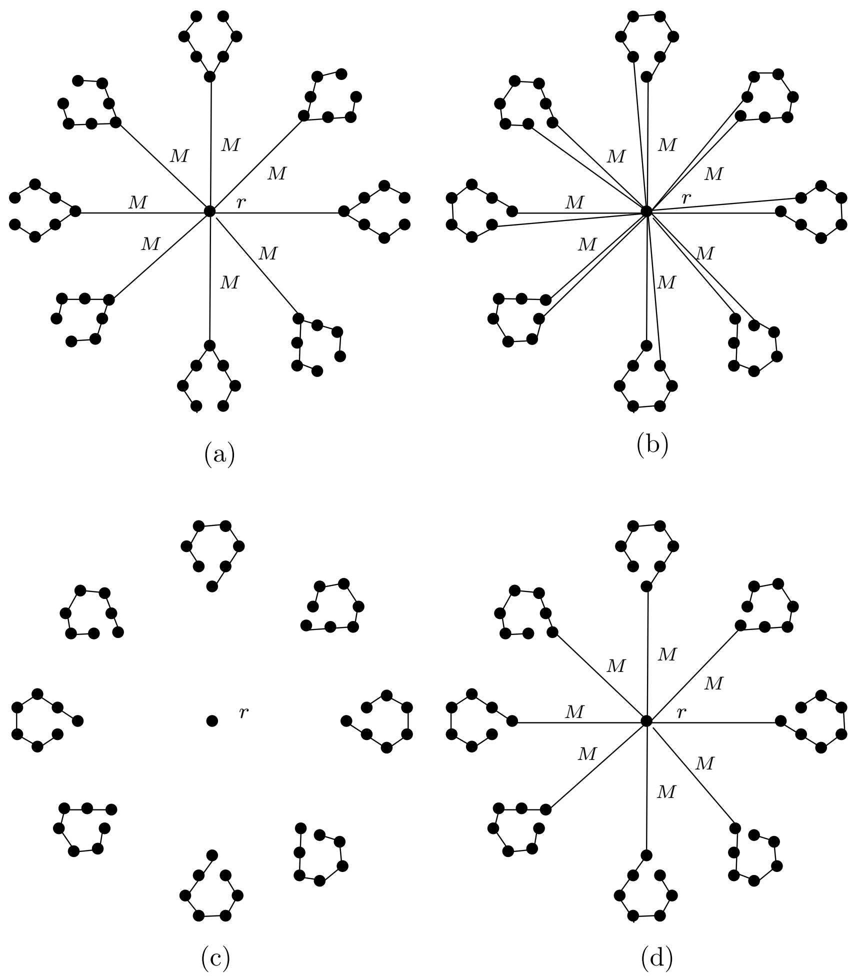

Figure 2: Our algorithm on the sample instance with $k = 8$. (a) MST spanning all the vertices. (b) TSP tours, each spanning r and the vertices in a subtree rooted hanging off r. (c) Partitioning of the TSP tours into segments. (d) Partitioned subtrees are connected to root r, with the cost of the final tree being $kM + (k-1)\epsilon$.