A Faster, Better Approximation Algorithm for the Minimum Latency Problem

Aaron Archer[*] Asaf Levin[†] David P. Williamson[‡]

Abstract

We give a 7.18-approximation algorithm for the minimum latency problem that uses only $O(n \log n)$ calls to the prize-collecting Steiner tree (PCST) subroutine of Goemans and Williamson. This improves the previous best algorithms in both performance guarantee and running time. A previous algorithm of Goemans and Kleinberg for the minimum latency problem requires an approximation algorithm for the k-MST problem which is called as a black box for each value of k. Their algorithm can achieve a performance guarantee of 10.77 while making $O(n^2 \log n)$ PCST calls (via a k-MST algorithm of Garg), or a performance guarantee of $7.18+\epsilon$ while using $n^{O(1/\epsilon)}$ PCST calls (via a k-MST algorithm of Arora and Karakostas). In all cases, the running time is dominated by the PCST calls. Since the PCST subroutine can be implemented to run in $O(n^2)$ time, the overall running time of our algorithm is $O(n^3 \log n)$.

The basic idea for our improvement is that we do not treat the k-MST algorithm as a black box. This allows us to take advantage of some special situations in which the PCST subroutine delivers a k-MST with a performance guarantee of 2. We are able to obtain the same approximation ratio that would be given by Goemans and Kleinberg if we had access to 2-approximate k-MST's for all values of k, even though we have them only for some values of k that we are not able to specify in advance.

1 Introduction

Given a metric space with n nodes and a tour starting at some node and visiting all of the others, the latency of node v is defined to be the total distance traveled before reaching v. The *minimum latency problem* (MLP) asks for a tour starting at a specified root r and visiting all nodes, such that the total latency is minimized. This problem is sometimes called the *traveling repairman problem* or the *delivery man problem*, and has been well studied in both the computer science and operations research literature. The MLP models routing problems in which one wants to minimize the average time each customer (node) has to wait before being served (visited), rather than the total time to visit all nodes, as in the case of the famous *traveling salesman problem* (TSP). In this sense, the MLP takes a customer-oriented view, whereas the TSP is server-oriented.

Koutsoupias et al. [21] and Ausiello et al. [7] motivate the MLP in terms of searching a graph (such as the web) to find a hidden treasure. If the treasure is equally likely to reside at any node of the graph, then the optimal MLP tour minimizes the expected time to find it.

[*]Operations Research Department, Cornell University, Ithaca, NY 14853. Email: aarcher@orie.cornell.edu. Supported by the Fannie and John Hertz Foundation. Most of this research was performed while visiting the IBM Almaden Research Center.

[†]Department of Statistics and Operations Research, Tel-Aviv University, Tel-Aviv 69978, Israel. Email: levinas@post.tau.ac.il

[‡]IBM Almaden Research Center, 650 Harry Rd. K53/B1, San Jose, CA 95120. Email: dpw@almaden.ibm.com.

The MLP was shown to be NP-hard for general metric spaces by Sahni and Gonzalez [27]. It is also Max-SNP hard, by a reduction from TSP(1,2) (the traveling salesman problem with all distances 1 or 2) [25, 10]. Therefore, there is no polynomial-time approximation scheme for the MLP on general metric spaces unless $P = NP$. Recently, Sitters [29] showed the problem is NP-hard even for weighted trees. On the positive side, Arora and Karakostas gave quasipolynomial-time approximation schemes for the MLP on weighted trees and constant dimensional Euclidean spaces [4]. Blum et al. [10] gave the first constant factor approximation algorithm for general metric spaces, which was later improved by Goemans and Kleinberg [19]. We elaborate on these below. Recently, Fakcharoenphol et al. [12] gave a constant factor approximation algorithm for the variant where there are k repairmen who must collectively visit all the nodes.

Much work has focused on exact (exponential time) solution approaches to the MLP [28, 14, 9, 23, 35], and on the more general time-dependent traveling salesman problem (TDTSP) [32, 26]. In the TDTSP, the distance between the i^{th} and $(i+1)^{st}$ nodes in the traveling salesman tour is multiplied by some weight $w(i)$ in the objective function. The ordinary TSP is the case where all $w(i) = 1$; the MLP is the case where $w(i) = n - i$. The time-dependent orienteering problem (considered in [15]) is dual to the TDTSP – the salesman aims to maximize the number of nodes visited before a given deadline, given that travel times vary as in the TDTSP. Various heuristics for the MLP are evaluated in [31, 33], while [2] analyzes a stochastic version of the problem. In the online variant [13, 22], new nodes appear in the graph as the repairman is traveling. Many authors have considered special cases of the MLP, where the metric is given by an underlying network with some special structure [1, 24, 8, 30, 34].

Because the MLP is NP-hard, we shall consider approximation algorithms for the problem. An α-approximation algorithm produces solutions with total latency no more than α times the total latency of a minimum latency tour. The value of α is called the *performance guarantee* of the algorithm.

The first approximation algorithm for the problem was given by Blum et al. [10], who achieve a performance guarantee of 72. They also showed how to use a β-approximation algorithm for the rooted k-minimum spanning tree (k-MST) problem as a black box, and convert it into an 8β-approximation algorithm for the MLP. In the k-MST problem, we are given a graph with costs on the edges, and must find the minimum-cost tree spanning at least k nodes. In the *rooted* version, the tree must contain some specified root r. The connection between the k-MST problem and the MLP is that the cost of the optimal k-MST rooted at r is a lower bound on the latency of the k^{th} node visited by the optimal MLP tour. Goemans and Kleinberg (GK) [19] subsequently improved the performance guarantee of the algorithm of Blum et al. to 3.59β. The best approximation algorithms now known for the rooted k-MST problem are a $(2+\epsilon)$-approximation by Arora and Karakostas (AK) [5], and a 3-approximation by Garg [17], yielding MLP guarantees of $7.18+\epsilon$ and 10.77, respectively.[1]

In this paper we further explore the connection between the MLP and rooted k-MST problem. We obtain a performance guarantee of 7.18, slightly improving the previous best of $7.18 + \epsilon$.[2] Moreover, our algorithm also has a running time that is better than the running time of the GK algorithm using either AK or Garg. In each of these algorithms, the running time is dominated by subroutine calls to an algorithm of Goemans and Williamson for the prize-collecting Steiner tree (PCST) problem [20]. The GK algorithm using Garg as a subroutine requires $O(n^2 \log n)$ PCST

[1]Arya and Ramesh [6] gave a 2.5-approximation algorithm and Garg [18] announced a 2-approximation algorithm for the *unrooted* version of the k-MST problem, but these cannot be used in the MLP algorithms.

[2]An earlier version of this paper [3] had the same algorithm, but an analysis with a performance guarantee of 9.28.

calls, and using AK it requires $n^{O(\frac{1}{\epsilon})}$ calls. Our algorithm requires only $O(n \log n)$ PCST calls.

Goemans and Williamson showed how to implement their PCST algorithm in $O(n^2 \log n)$ time. A recent result of Gabow and Pettie [16] improves this to $O(n^2)$. Thus, our algorithm runs in time $O(n^3 \log n)$ overall.

The main idea in achieving our result is that we do not treat the k-MST algorithm as a black box. It is easy to show that the PCST algorithm returns k-MSTs of cost no more than twice optimal for some values of k that cannot be specified in advance [11]. We call a k-MST with cost no more than α times the optimal k-MST an *α-approximate* k-MST. If we had 2-approximate k-MST's for all $k = 2, \ldots, n$, then we could run the GK algorithm, which uses the costs of the trees to select some subset of them to concatenate into an MLP tour. Our trick is to successfully bluff the GK algorithm. We *pretend* to have trees of all sizes by interpolating the costs of the trees we do have to fill in the tree costs for the missing values of k. We refer to these as *phantom* trees. We then prove that if the GK algorithm were to be run with both the real and phantom trees, it would never choose any of the phantom trees to concatenate, so it never calls our bluff. For the analysis to go through, we must also carefully extend our k-MST lower bounds to the phantom values of k. To do this, we utilize the fact that the PCST problem is a Lagrangean relaxation of the k-MST problem, as observed in [11].

The paper is structured as follows. In Section 2, we review the main ideas of previous approximation algorithms for the MLP. In Section 3, we give our algorithm, assuming a certain kind of k-MST algorithm. Section 4 analyzes the performance guarantee of the algorithm, and in Section 5 we give the k-MST algorithm that we need. We conclude in Section 6 with some thoughts about approaches for further improvements.

2 Intuition and overview

We now describe the basic ideas behind the Blum et al. [10] and GK [19] algorithms, and how our approach departs from them. Both analyses use the cost of the optimal k-MST as a lower bound for the latency of the k^{th} node visited in the optimal MLP tour, and both algorithms start with β-approximate solutions to the k-MST problem rooted at r, for $k = 2, 3, \ldots, n$. They then select a subsequence of these trees with geometrically increasing costs and concatenate them to get a solution for the MLP. For the sake of intuition, let us assume throughout this section that the sets of nodes spanned by these trees are nested, which turns out to be the worst case for the analysis.

Without loss of generality, the cost of the k-MST's increases with k. The Blum et al. algorithm buckets the trees according to their cost – for each integer ℓ, it selects the most expensive tree with cost in $(2^\ell, 2^{\ell+1}]$. It doubles each of the selected trees and shortcuts it to make a cycle rooted at r, then traverses all of these cycles in order, shortcutting nodes it has already visited. Since the last tree selected spans all the nodes, so does the resulting MLP tour. They compare the latency of the k^{th} node visited in the tour to the cost of the optimal k-MST. They upper bound the latency of the k^{th} node visited by the total cost of all the concatenated cycles up to and including the first one that visits this node. They lose a factor of β because the trees are β-approximate k-MST's, a factor of 2 from the bucketing ratio, a factor of 2 from doubling the trees to get cycles, and a factor of 2 from the geometric sum. This yields the approximation factor of 8β.

The GK improvement derives from two sources. First, it orients each of the concatenated cycles in the direction that minimizes the total latency of the new nodes visited by that cycle. Second, it applies a random shift to the bucket breakpoints. Using buckets of ratio $\gamma \approx 3.59$ instead of ratio 2, it achieves a performance guarantee of $\gamma\beta$.

Our algorithm departs from these previous ones in that we do not start off with approximate k-MST's for every value of k. Instead, we obtain $(2-\frac{1}{n})$-approximate n_i-MST's for some subsequence $1 = n_1 < \cdots < n_\ell = n$ that is not under our control. Let d_i denote the cost of the tree spanning i nodes and b_i denote our lower bound on the optimal i-MST, for $i = n_1, \ldots, n_\ell$. We derive these trees using a Lagrangean relaxation technique, which allows us to guarantee that linearly interpolating the b_{n_i} to the missing values of k yields valid lower bounds on the cost of the optimal k-MST. We will obtain our MLP solution by concatenating some subset of these trees, as in Blum et al. and GK.

The GK analysis uses the idea of *modified latency*. Roughly, one can think of the modified latency of node v as the average latency of all the nodes first visited by the cycle in the concatenation that first visits node v. The total modified latency is an upper bound on the latency of the MLP tour we construct. The GK randomized bucketing procedure yields a solution whose expected total modified latency is at most $\gamma(d_2 + \cdots + d_n) \leq \gamma\beta(OPT_2 + \cdots + OPT_n) \leq \gamma\beta OPT$ (where OPT denotes the value of the optimal MLP tour, and OPT_i denotes optimal i-MST value). Goemans and Kleinberg also observe that one can use a shortest path computation to determine which concatenation of trees minimizes the total modified latency. Since the minimum is no more than the expectation, this yields a deterministic $\gamma\beta$-approximation algorithm. Whereas Goemans and Kleinberg introduce the shortest path calculation merely to derandomize their algorithm, for us the use of the shortest path computation is central to the analysis of our performance guarantee.

3 The algorithm

Here we precisely specify our MLP algorithm. We start by using our tree-generating algorithm of Section 5 to produce some set of ℓ trees $T_{n_1}, \ldots, T_{n_\ell}$ rooted at r and spanning $n_1 < \cdots < n_\ell$ nodes respectively, where $n_1 = 1$ and $n_\ell = n$. For $i = n_1, n_2, \ldots, n_\ell$, let d_i denote the cost of tree T_i. Without loss of generality we assume d_i is increasing with i. Our tree-generating algorithm also establishes lower bounds on OPT_k, the cost of the optimal k-MST rooted at r, and hence on the latency of the k^{th} node visited in the optimal MLP tour, for $k = 2, \ldots, n$. These lower bounds b_k satisfy the properties

1. $b_k \leq OPT_k$ for $1 \leq k \leq n$;
2. $d_{n_i} \leq 2b_{n_i}$ for $1 \leq i \leq \ell$;
3. $b_k = b_{n_{i-1}} + \frac{b_{n_i} - b_{n_{i-1}}}{n_i - n_{i-1}}(k - n_{i-1})$ for $n_{i-1} \leq k \leq n_i$ and $i = 2, \ldots, n$.

That is, b_k is the linear interpolation of $b_{n_{i-1}}$ and b_{n_i}, and each T_{n_i} is a 2-approximate n_i-MST.

Now we use a shortest path calculation described below to select some subcollection of these trees to concatenate. Denote the selected trees by $T_{j_1}, \ldots, T_{j_m}$, so $j_1, \ldots, j_m$ is the increasing sequence of nodes they span. For each selected tree T_i, double all of the tree edges and traverse an Eulerian tour starting at r, shortcutting nodes already visited, to obtain a cycle $\hat{C}_i$. Now obtain cycle C_i from $\hat{C}_i$ by shortcutting all nodes (except for r) that are visited by some $\hat{C}_k$ with $k < i$. Let $S_i = C_i - r$. Orient C_i in the direction that minimizes the total latency of the nodes in S_i.

To obtain our MLP solution, simply traverse each rooted, oriented cycle $C_{j_1}, \ldots, C_{j_m}$ in order, shortcutting the intermediate visits to the root between cycles. Let $C = C_{j_1}, \ldots, C_{j_m}$ denote this concatenated tour. Following Goemans and Kleinberg, we define the *modified latency* of the k^{th} node of C to be

$$\pi_k = d_{j_{p(k)}} + 2(d_{j_{p(k)-1}} + \cdots + d_{j_1}), \tag{1}$$

where $p(k)$ is the smallest index such that $k \leq j_{p(k)}$. The motivation for this definition is that if the sets of nodes spanned by $T_{j_1}, \ldots, T_{j_m}$ are nested, then π_k is an upper bound on the average latency of the nodes first visited by cycle $C_{j_{p(k)}}$. Indeed, in Section 4, we repeat an argument of Goemans and Kleinberg that in all cases $\pi_2 + \cdots + \pi_n$ is an upper bound on the total latency of C.

It is now easy to describe the shortest path computation. We construct a graph G with nodes $n_1, \ldots, n_\ell$, and arcs $i \to k$ for each $i < k$. A path $j_1 \to \cdots \to j_m$ corresponds to selecting trees $T_{j_1}, \ldots, T_{j_m}$. Thus, the cost on arc $i \to k$ is

$$(k-i)d_k + 2(n-k)d_k = 2d_k \left(n - \frac{i+k}{2} \right), \tag{2}$$

which corresponds to the contribution made to the total modified latency by traversing tree T_k immediately after traversing T_i. This is because tree T_k contributes d_k to the modified latencies of the $(k-i)$ new nodes it visits, and $2d_k$ to each of the remaining $(n-k)$ unvisited nodes. If the shortest path from n_1 to n_ℓ in this graph goes $j_1 \to \cdots \to j_m$, then we select trees $T_{j_1}, \ldots, T_{j_m}$ to concatenate.

4 Analyzing the approximation ratio

The analysis of our algorithm proceeds in three steps. First we demonstrate that $\pi_2 + \cdots + \pi_n$ really is an upper bound on the latency of the tour we construct. Next, we appeal to a result of Goemans and Kleinberg that upper bounds the total modified latency of the tour given by the shortest path computation, in the event that we have trees of all sizes, $2, \ldots, n$. Finally, we show that if we were to run this shortest path computation with our real trees and some "phantom" interpolated trees, the computation would never select any of the phantom trees. Therefore, we achieve the same performance guarantee that GK would achieve if we actually did have access to the phantom trees.

We begin by repeating an argument of Goemans and Kleinberg about $\pi_2 + \cdots + \pi_n$.

Lemma 4.1 ([19]) *The total latency of the MLP tour obtained by concatenating trees $T_{j_1}, \ldots, T_{j_m}$ (where $j_1 = 1$, $j_m = n$) is at most $\pi_2 + \cdots + \pi_n$, where the π_k are given by* (1)*.*

Proof: The following argument essentially says that the worst case for our analysis is when the sets of nodes spanned by the selected trees $T_{j_1}, \ldots, T_{j_m}$ are nested. Let us consider the latency of the k^{th} node we visit in C, where r is considered to be the first node, whose latency is zero. If the k^{th} node in C was encountered as part of cycle C_{j_p}, then we can upper bound its latency by the sum of the costs of cycles $C_{j_1}, \ldots, C_{j_{p-1}}$ plus the portion of cycle C_{j_p} that is traversed prior to reaching this node. Since we traverse cycle C_{j_p} in the direction that minimizes the total latency of the new nodes S_{j_p}, the average contribution of this cycle to the latencies of the nodes in S_{j_p} is at most half the cost of the cycle. To see this, notice that for any node $i \in S_{j_p}$, if we traverse C_{j_p} in one direction, it contributes some amount x to the latency of i, and if we traverse it in the other direction, it contributes $\text{cost}(C_{j_p}) - x$, so on average it contributes $\text{cost}(C_{j_p})/2$. For each $i = j_1, \ldots, j_m$, the cost of C_i is clearly at most $2d_i$. Therefore, the average latencies of the nodes in S_{j_p} is at most $2(d_{j_1} + \cdots + d_{j_{p-1}}) + d_{j_p}$, so the total latency of C is at most

$$\sum_{p=1}^{m} |S_{j_p}|(2(d_{j_1} + \cdots + d_{j_{p-1}}) + d_{j_p}). \tag{3}$$

Since $\sum |S_{j_p}| = n$, we can view this as a weighted sum. Clearly, the worst case for this analysis is when the sets of nodes spanned by the trees $T_{j_1}, \ldots, T_{j_m}$ are nested, since this puts the greatest weight on the larger terms in (3). In this worst case, our upper bound on the average latency of the $(j_{p-1}+1)^{th}$ through j_p^{th} nodes in C becomes $2(d_{j_1} + \cdots + d_{j_{p-1}}) + d_{j_p}$. Thus since $\pi_k = d_{j_{p(k)}} + 2(d_{j_{p(k)-1}} + \cdots + d_{j_1})$, where $p(k)$ is the smallest index such that $k \leq j_{p(k)}$, our tour has latency at most $\pi_2 + \cdots + \pi_n$. ■

The advantage of the upper bound $\pi_2 + \cdots + \pi_n$ is that it depends only on the costs of the selected trees and the number of nodes they span, not on their structure. We now state the main theorem of the Goemans and Kleinberg paper.

Theorem 4.2 ([19]) *Given $d_2, \ldots, d_n \geq 0$ and a graph G on nodes $1, \ldots, n$ including all arcs $i \to k$ for $i < k$, with arc lengths given by (2), then the shortest path in G from node 1 to node n has length at most $\gamma(d_2 + \cdots + d_n)$, where $\gamma \approx 3.59$ is the unique root of $\gamma \ln \gamma = \gamma + 1$.*

Recall that our tree-generating procedure of Section 5 returns trees of sizes $n_1, \ldots, n_\ell$ and costs $d_{n_1}, \ldots, d_{n_\ell}$. It also establishes lower bounds b_k on the cost OPT_k of the optimal k-MST, for every k, and these bounds satisfy properties (1)-(3) from Section 3. Let us linearly interpolate the tree costs d_{n_i} to the missing values of k. That is, set

$$d_k = d_{n_{i-1}} + \frac{d_{n_i} - d_{n_{i-1}}}{n_i - n_{i-1}}(k - n_{i-1}) \tag{4}$$

for $n_{i-1} \leq k \leq n_i$ and $i = 2, \ldots, n$. Then clearly $d_k \leq 2b_k$ for all k. That is, if we actually had a tree of cost d_k spanning k nodes, it would be a 2-approximate k-MST. Since we don't actually have these trees, we will call them *phantom trees*. Now suppose we were to run the GK shortest path computation using the full set of costs $d_2, \ldots, d_n$, i.e. using both the real trees and the phantom trees. Then by Theorem 4.2, the modified latency of the resulting solution would be at most

$$\begin{aligned} \gamma(d_2 + \cdots + d_n) &\leq 2\gamma(b_2 + \cdots + b_n) \\ &\leq 2\gamma(OPT_2 + \cdots + OPT_n) \\ &\leq 2\gamma OPT, \end{aligned}$$

where OPT denotes the optimal MLP value. The difficulty is that the shortest path computation might select one of the phantom trees, in which case we cannot actually construct the MLP tour. Fortunately, this never occurs.

Theorem 4.3 *In the shortest path computation described above, a shortest path never visits any of the nodes corresponding to phantom trees.*

Proof: Suppose on the contrary that a shortest path visits $i \to j \to k$, where $n_{lo} < j < n_{hi}$ and n_{lo} and n_{hi} are two consecutive sizes of actual trees in our collection. Treating j as a variable now, we show that we can obtain a strictly shorter path by setting j to either $\max(i, n_{lo})$ or $\min(k, n_{hi})$, arriving at a contradiction. Set $\lambda = (d_{n_{hi}} - d_{n_{lo}})/(n_{hi} - n_{lo}) > 0$. By definition, $d_j = d_{n_{lo}} + \lambda(j - n_{lo})$. By definition of the arc lengths, the subpath from $i \to j \to k$ costs

$$\left(n - \frac{i+j}{2}\right) 2d_j + \left(n - \frac{j+k}{2}\right) 2d_k = \left(n - \frac{i+j}{2}\right) 2(d_{n_{lo}} + \lambda(j - n_{lo})) + \left(n - \frac{j+k}{2}\right) 2d_k. \quad (5)$$

This cost is valid for $\max(i, n_{lo}) \leq j \leq \min(k, n_{hi})$ and is a quadratic function of j, where the coefficient on the j^2 term is $-\lambda$. Thus, the cost is strictly concave in j, so it attains a strict minimum at one of the endpoints $\max(i, n_{lo})$ or $\min(k, n_{hi})$. In the case that the minimum is attained at i or k, we can eliminate the self-loop $i \to i$ or $k \to k$ to further reduce the path length. This is a contradiction, because we already started with a shortest path. ■

Since a shortest path in the graph with the phantom trees included never actually uses any of the phantom trees, we might as well run the shortest path computation using just the actual trees, as in the algorithm description of Section 3. Putting Theorem 4.3 together with the discussion preceeding it yields our main result.

Theorem 4.4 *The algorithm described in Section 3 yields an MLP tour of cost at most $2\gamma OPT$, where $\gamma \approx 3.59$ is the unique root of $\gamma \ln \gamma = \gamma + 1$.*

5 The tree-finding algorithm and analysis

In this section, we give the algorithm for finding the set of trees $T_{n_1}, T_{n_2}, \ldots, T_{n_\ell}$ including a root node r and spanning $1 = n_1 < n_2 < \cdots < n_\ell = n$ nodes. Denote the cost of the trees by $d_{n_1}, \ldots, d_{n_\ell}$. We also compute a set of n lower bounds $b_1, \ldots, b_n$. For simplicity, we denote the cost of an optimal k-MST as OPT_k. We need to find a set of trees and lower bounds such that the following three properties hold:

1. $b_k \leq OPT_k$ for all k, $1 \leq k \leq n$;

2. $d_{n_i} \leq 2b_{n_i}$;

3. $b_k = b_{n_{i-1}} + \frac{b_{n_i} - b_{n_{i-1}}}{n_i - n_{i-1}}(k - n_{i-1})$ for $n_{i-1} \leq k \leq n_i$ and $i = 2, \ldots, n$,

as promised in Section 3.

The basic idea is as follows. To obtain trees, we apply a primal-dual algorithm that is parameterized by λ; more precisely, we apply the 2-approximation algorithm of Goemans and Williamson [20] for the prize-collecting Steiner tree problem, where all penalties are set to λ. The algorithm returns a tree T and a lower bound b such that $\text{cost}(T) \leq (2 - \frac{1}{n-1})b$, and $b \leq OPT_{|T|}$. For $k = 1, \ldots, n$, we perform a binary search on the value of λ, seeking a tree returned by the algorithm with exactly k nodes. If we succeed, we add the tree and the lower bound to our collection. If we do not succeed, in the end we have two trees T_{lo} and T_{hi} and two bounds b_{lo} and b_{hi} with $|T_{lo}| < k$ and $|T_{hi}| > k$ for two values of λ sufficiently close. We add both trees and bounds to our collection. We show that for close enough values of λ, an interpolation of b_{lo} and b_{hi} gives a lower bound b_k on the value of a tree containing k nodes. At the end of the procedure, we pick a subset of trees found so that all three desired properties hold.

We begin by explaining the pieces of the algorithm that we will need. We model the k-MST problem as the following integer program:

$$\text{Min} \qquad \sum_{e \in E} c_e x_e$$

subject to:

$$\begin{aligned}
\sum_{e \in \delta(S)} x_e + \sum_{T: T \supseteq S} z_T &\geq 1 \quad && \forall S \subseteq V \setminus \{r\} \\
\sum_{S: S \subseteq V \setminus \{r\}} |S| z_S &\leq n - k && \\
x_e &\in \{0, 1\} && \forall e \in E \\
z_S &\in \{0, 1\} && \forall S \subseteq V \setminus \{r\},
\end{aligned}$$

where $\delta(S)$ is the set of edges with exactly one endpoint in S. The variable $x_e = 1$ indicates that the edge e is in the tree, while $z_S = 1$ indicates that the set of nodes S is not spanned. The first set of constraints says that for any set S of nodes not containing the root, either they are contained in the unspanned set, or there is a selected edge in $\delta(S)$. The second constraint says that at most $n - k$ nodes are unspanned.

Following [11], we can convert this to something close to a prize-collecting Steiner tree problem by applying Lagrangean relaxation to the second constraint:

$$\text{Min} \qquad \sum_{e \in E} c_e x_e + \lambda \left(\sum_{S: S \subseteq V \setminus \{r\}} |S| z_S - (n - k) \right)$$

subject to:

$$\begin{aligned}
\sum_{e \in \delta(S)} x_e + \sum_{T: T \supseteq S} z_T &\geq 1 \quad && \forall S \subseteq V \setminus \{r\} \\
x_e &\in \{0, 1\} && \forall e \in E \\
z_S &\in \{0, 1\} && \forall S \subseteq V \setminus \{r\}.
\end{aligned}$$

Note that any solution feasible for the previous integer program will be feasible for this one at no greater cost for $\lambda \geq 0$. Recall the definition of the prize-collecting Steiner tree problem: we are given an undirected graph $G = (V, E)$, a root node $r \in V$, non-negative costs on the edges $c_e \geq 0$ for all $e \in E$, and non-negative penalties p_i for $i \in V, i \neq r$. The goal is to find a tree spanning the root node so as to minimize the cost of the edges in the tree plus the penalties of the nodes not in the tree. Here we set all penalties $p_i = \lambda$. Observe that the integer program above exactly models this problem for $p_i = \lambda$, except that the objective function has an additional constant term of $-(n - k)\lambda$.

Goemans and Williamson [20] give a primal-dual 2-approximation algorithm for the prize-collecting Steiner tree problem. Their algorithm returns a tree spanning the root node, and a solution to the dual of a linear programming relaxation of the prize-collecting Steiner tree problem. The dual solution is feasible for the dual of the linear programming relaxation of the integer program above; in particular, this dual is:

$$\text{Max} \quad \sum_{S \subseteq V \setminus \{r\}} y_S - (n-k)\lambda$$

subject to:

$$(D) \qquad \begin{array}{lr} \displaystyle\sum_{S: e \in \delta(S)} y_S \leq c_e & \forall e \in E \\ \displaystyle\sum_{T: T \subseteq S} y_T \leq |S|\lambda & \forall S \subseteq V \setminus \{r\} \\ y_S \geq 0 & \forall S \subseteq V \setminus \{r\}. \end{array}$$

We will abbreviate their algorithm as PCST. In particular, they show the following.

Theorem 5.1 ([20]) *PCST returns a tree T and a dual solution y feasible for (D) such that if X is the set of nodes not spanned by T, then*

$$\sum_{e \in T} c_e + \left(2 - \frac{1}{n-1}\right) \lambda |X| \leq \left(2 - \frac{1}{n-1}\right) \sum_{S \subseteq V \setminus \{r\}} y_S.$$

Lemma 5.2 *If the tree T returned by PCST contains k nodes, then*

$$\tilde{b}_k := \sum_{S \subseteq V \setminus \{r\}} y_S - (n-k)\lambda \leq OPT_k,$$

and the cost of T is no more than $\left(2 - \frac{1}{n-1}\right) \tilde{b}_k$.

Proof: Note that if y is a feasible dual solution to (D), then since $\sum_{S \subseteq V \setminus \{r\}} y_S - (n-k)\lambda$ is the dual objective function of (D), it is a lower bound on the cost of an optimal k-MST. By Theorem 5.1, if PCST returns tree T and X is the set of $n-k$ nodes not spanned by T, then

$$\sum_{e \in T} c_e + \left(2 - \frac{1}{n-1}\right) \lambda(n-k) \leq \left(2 - \frac{1}{n-1}\right) \sum_{S \subseteq V \setminus \{r\}} y_S,$$

which implies that

$$\sum_{e \in T} c_e \leq \left(2 - \frac{1}{n-1}\right) \left(\sum_{S \subseteq V \setminus \{r\}} y_S - (n-k)\lambda \right) \leq \left(2 - \frac{1}{n-1}\right) OPT_k.$$

■

We further need the following observation, which relies on the workings of the PCST algorithm.

Observation 5.3 *If we call PCST with $\lambda = 0$, it will return a tree containing only the root node. If we call PCST with $\lambda = c_{\max}$, where $c_{\max}$ is the maximum edge cost, it will return a tree with n nodes.*

As suggested above, we now perform the following algorithm. For each value of k, we call a k-MST subroutine. This subroutine performs binary search on the value of λ, looking for a value of λ

such that PCST returns exactly k nodes. If we find such a tree, we return it and the corresponding dual lower bound $\tilde{b}_k$ equal to the value of the objective function of (D) for the dual solution returned by PCST. If we do not find such a tree, we continue the binary search until $\lambda_{hi} - \lambda_{lo}$ is no more than $c_{\min}/n(4n-5)$, where $c_{\min}$ is the cost of the cheapest edge adjacent to the root. We then return the two trees T_{lo} and T_{hi}, and the two lower bounds $\tilde{b}_{lo}$ and $\tilde{b}_{hi}$. Note that we can assume that $c_{\min} > 0$ since otherwise we can take the connected component of zero cost edges containing the root node, visit all the nodes in the component first as part of our latency tour, and return to the root. We can then discard these nodes from consideration in building the rest of the tour, since they add no latency to the rest of the tour. Observe that for each value of k, we call PCST $O(\log \frac{nc_{\max}}{c_{\min}})$ times, for a total of $O(n \log \frac{nc_{\max}}{c_{\min}})$ invocations. At the end of the section we discuss how this can be reduced to $O(n \log n)$ PCST invocations.

For technical reasons, we will actually scale the values of the lower bounds down by a factor of $1 - \frac{1}{4(n-1)}$ before returning them. Let b denote the scaled value of $\tilde{b}$; that is, $b = \left(1 - \frac{1}{4(n-1)}\right) \tilde{b}$. We state below that in the case that the k-MST routine returns two trees, then interpolating the scaled bounds b_{lo} and b_{hi} appropriately gives a lower bound b_k on the value of an optimal k-MST. The proof is deferred for a moment. For two trees T_{lo} and T_{hi} returned by the algorithm such that $|T_{lo}| < k$ and $|T_{hi}| > k$, let $\alpha_{lo}, \alpha_{hi} \geq 0$ be such that $\alpha_{lo} + \alpha_{hi} = 1$ and $\alpha_{lo}|T_{lo}| + \alpha_{hi}|T_{hi}| = k$.

Lemma 5.4 *Let T_{lo}, T_{hi} be trees returned by PCST when all penalties are set to λ_{lo} and λ_{hi} respectively, with $|T_{lo}| < k$ and $|T_{hi}| > k$, and $\lambda_{hi} - \lambda_{lo} \leq c_{\min}/n(4n-5)$. Express k as a convex combination $k = \alpha_{lo}|T_{lo}| + \alpha_{hi}|T_{hi}|$, where $\alpha_{lo} + \alpha_{hi} = 1$. If we set $b_k = \alpha_{lo} b_{lo} + \alpha_{hi} b_{hi}$, then $b_k \leq OPT_k$.*

At the end of the process we will have a set of trees T and their associated lower bounds b. For a tree T_k of cost d_k, and its associated lower bound b_k, Lemma 5.2 gives

$$d_k \leq \left(2 - \frac{1}{n-1}\right) \tilde{b}_k = \frac{2 - \frac{1}{n-1}}{1 - \frac{1}{4(n-1)}} b_k \leq 2b_k, \tag{6}$$

so we satisfy Property (2). Whatever collection of trees we select, we will linearly interpolate the bounds b_{n_i} so that Property (3) holds by definition. Thus, to enforce Property (1) we must figure out which trees to keep so that the interpolated bounds b_k are valid lower bounds on OPT_k.

To enforce Property (1), we keep only a subset of the trees, removing every T_k such that its corresponding bound b_k is greater than the linear interpolation of any pair of other bounds in the set. This is equivalent to keeping only the trees corresponding to the lower envelope of bounds plotted with the size of the tree on the x axis and the value of the bound on the y axis. We denote our final set of trees by $T_{n_1}, \ldots, T_{n_\ell}$, where $n_1 = 1 < n_2 < \cdots < n_\ell = n$, and tree T_{n_i} spans n_i nodes, costs d_{n_i}, and has an associated bound b_{n_i}. For k such that $n_{i-1} < k < n_i$, we set b_k to be the appropriate interpolation of $b_{n_{i-1}}$ and b_{n_i}. We now show that this satisfies Property (1), which should be clear from Figure 1. Note that the running time for computing this lower envelope is dominated by the PCST calls.

Lemma 5.5 *For all k, $1 \leq k \leq n$, $b_k \leq OPT_k$.*

Proof: For the set of trees prior to pruning it is the case that for every k, $1 \leq k \leq n$, either there is a tree of size k in the set or, by Lemma 5.4, there are two trees T' and T'', with $|T'| < k$ and $|T''| > k$, such that there exist $\alpha', \alpha'' \geq 0$ with $\alpha' + \alpha'' = 1$, $\alpha'|T'| + \alpha''|T''| = k$, and $\alpha' b' + \alpha'' b'' \leq OPT_k$. After taking the lower envelope, we claim it is clear that the appropriate interpolation of $b_{n_{i-1}}$ and

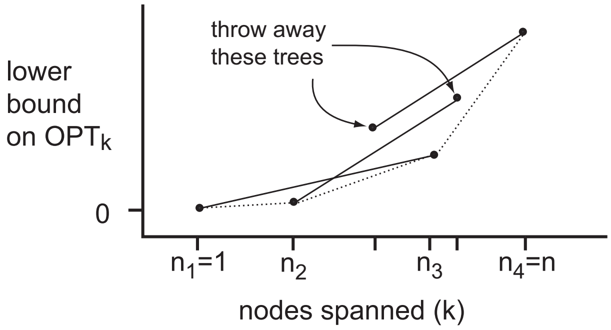

Figure 1: Each dot represents a (size of tree,lower bound) pair returned by PCST, and the solid lines are the interpolated lower bounds from Lemma 5.4. The lower envelope (dotted line) is still clearly a valid lower bound at each point, since it is below the solid lines. We keep only the trees whose dots are on the lower envelope.

b_{n_i} is a lower bound on OPT_k, for k such that $n_{i-1} < k < n_i$. This follows since the lower bound on OPT_k is a convex combination of one or two bounds in the initial set; once we take the lower envelope, the appropriate interpolation of $b_{n_{i-1}}$ and b_{n_i} can be no greater. ■

We now complete the proof of Lemma 5.4.

Proof of Lemma 5.4: Let $k_{lo} = |T_{lo}|$ and $k_{hi} = |T_{hi}|$. Let y^{lo} and y^{hi} be the dual solutions returned by PCST for penalty value λ_{lo} and λ_{hi} respectively. Letting $y = \alpha_{lo} y^{lo} + \alpha_{hi} y^{hi}$ and $\delta = (1 - \frac{1}{4(n-1)})$, observe that

$$\begin{aligned}
b_k &= \alpha_{lo} b_{lo} + \alpha_{hi} b_{hi} \\
&= \delta \left(\alpha_{lo} \tilde{b}_{lo} + \alpha_{hi} \tilde{b}_{hi} \right) \\
&= \delta \left(\alpha_{lo} \left(\sum_{S \subseteq V \setminus \{r\}} y_S^{lo} - (n - k_{lo}) \lambda_{lo} \right) + \alpha_{hi} \left(\sum_{S \subseteq V \setminus \{r\}} y_S^{hi} - (n - k_{hi}) \lambda_{hi} \right) \right) \\
&= \delta \left(\sum_{S \subseteq V \setminus \{r\}} y_S - \alpha_{lo}(n - k_{lo})(\lambda_{hi} + \lambda_{lo} - \lambda_{hi}) - \alpha_{hi}(n - k_{hi}) \lambda_{hi} \right) \\
&\leq \delta \left(\sum_{S \subseteq V \setminus \{r\}} y_S - (n - k)\lambda_{hi} + \alpha_{lo}(n - k_{lo}) \frac{c_{\min}}{n(4n-5)} \right) \\
&\leq \left(1 - \frac{1}{4(n-1)} \right) \left(\sum_{S \subseteq V \setminus \{r\}} y_S - (n - k)\lambda_{hi} \right) + \frac{1}{4(n-1)} c_{\min} \\
&\leq OPT_k.
\end{aligned}$$

The last inequality holds because (y, λ_{hi}) is feasible for (D) by the convexity of the feasible region, and thus both $\sum_{S \subseteq V \setminus \{r\}} y_S - (n - k)\lambda_{hi}$ and $c_{\min}$ are lower bounds on the cost of an optimal k-MST. ■

We now use a little bit of slack built into our calculations to reduce the number of PCST calls from $O(n \log \frac{nc_{\max}}{c_{\min}})$ to $O(n \log n)$ and maintain a performance guarantee of $2\gamma \approx 7.18$. We observe that the n-MST will always be selected in the lower envelope calculation. By (6) we in fact know that

$$d_n \leq \left(2 - \frac{1}{n-1}\right) \tilde{b}_n = \frac{2 - \frac{1}{n-1}}{1 - \frac{1}{4(n-1)}} b_n = \left(2 - \frac{2}{4n-5}\right) b_n \leq \left(2 - \frac{2}{4n-5}\right) OPT_n.$$

Therefore, the latency of the tour constructed is in fact no more than $2\gamma OPT - \frac{2\gamma}{4n-5} OPT_n$. We use this bit of slack to reduce the number of PCST calls as follows. We look at the connected component of edges of cost at most $c_{\max}/4n^3$ that includes the root, and we visit all of the nodes of this component first at the beginning of our tour and return to the root. We then disregard these nodes as we construct the rest of our tour. For the remaining instance $c_{\min} \geq c_{\max}/4n^3$, and this adds cost at most $c_{\max}/2n$ to the total latency of the tour, since edge costs obey the triangle inequality. But $c_{\max} \leq OPT_n$, so this adds an additional $\frac{1}{2n} OPT_n$ to the tour cost, which can be absorbed by the $-\frac{2\gamma}{4n-5} OPT_n$ term.

6 Concluding remarks

We showed how to use the tree concatenation technique of Blum et al. as refined by Goemans and Kleinberg to construct a 7.18-approximation algorithm for the MLP, while having access to 2-approximate k-MST's for only a few values of k that we cannot specify in advance. The $7.18 = 2\gamma$ guarantee comes from two sources. The 2 comes from our k-MST's being 2-approximate, while the γ comes from the tree concatenation procedure. Both of these pieces represent significant barriers to further improvement.

All known constant factor approximation algorithms for the k-MST problem rely explicitly or implicitly on the LP relaxation we used for the PCST problem. Since this relaxation has an integrality gap of essentially 2, it seems that achieving a β-approximation algorithm for k-MST for a constant $\beta < 2$ will require a significantly different approach.

The factor of $\gamma \approx 3.59$ from the tree concatenation is also inherent in any analysis that blindly concatenates trees and upper bounds the latency by the sum of modified latencies. This is because Goemans and Kleinberg prove that Theorem 4.2 is tight; that is, the costs $d_2, \ldots, d_n$ can be selected such that the ratio of shortest path length in the graph G to $d_2 + \cdots + d_n$ is arbitrarily close to γ. Thus, in order to attain a provably better latency, one would need to either have some knowledge of the costs d_i, or pay attention to the actual structure of the trees being concatenated.

One direction for future work would be to consider LP relaxations that address the MLP objective function directly, rather than using k-MST for our lower bounds. Perhaps the most attractive special case to look at is the case where the underlying metric is given by a tree. Since k-MST can be solved optimally on trees, the GK algorithm can be used to obtain a 3.59-approximation for this special case. At present, this is the best result known.

Acknowledgments

A preliminary version of this paper by the first and third authors [3] had a performance guarantee of 9.28. The second author contributed the core idea for the improvement of the performance guarantee to 7.18.

We thank Tim Roughgarden for many enlightening discussions. The first author is supported by the Fannie and John Hertz Foundation. His research was carried out primarily while visiting IBM Almaden.

References

[1] F. Afrati, S. Cosmadakis, C. H. Papadimitriou, G. Papageorgiou, and N. Papakostantinou. The complexity of the traveling repairman problem. *Informatique Theorique et Applications*, 20(1):79–87, 1986.

[2] S. R. Agnihothri. A mean value analysis of the travelling repairman problem. *IIE Transactions*, 20(2):223–229, June 1988.

[3] A. Archer and D. P. Williamson. Faster approximation algorithms for the minimum latency problem. In *Proceedings of the* 14^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003.

[4] S. Arora and G. Karakostas. Approximation schemes for minimum latency problems. In *Proceedings of the* 31^{st} *Annual ACM Symposium on Theory of Computing*, pages 688–693, 1999.

[5] S. Arora and G. Karakostas. A $2 + \epsilon$ approximation algorithm for the k-MST problem. In *Proceedings of the* 11^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 754–759, 2000.

[6] S. Arya and H. Ramesh. A 2.5 factor approximation algorithm for the k-MST problem. *Information Processing Letters*, 65(3):117–118, 1998.

[7] G. Ausiello, S. Leonardi, and A. Marchetti-Spaccamela. On salesmen, repairmen, spiders and other traveling agents. In *Proceedings of the Italian Conference on Algorithms and Complexity*, pages 1–16, 2000.

[8] I. Averbakh and O. Berman. Sales-delivery man problems on treelike networks. *Networks*, 25:45–58, 1995.

[9] L. Bianco, A. Mingozzi, and S. Ricciardelli. The traveling salesman problem with cumulative costs. *Networks*, 23(2):81–91, 1993.

[10] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan. The minimum latency problem. In *Proceedings of the* 26^{th} *Annual ACM Symposium on Theory of Computing*, pages 163–171, 1994.

[11] F. Chudak, T. Roughgarden, and D. P. Williamson. Approximate k-MSTs and k-Steiner trees via the primal-dual method and Lagrangean relaxation. In *Proceedings of the* 8^{th} *Conference on Integer Programming and Combinatorial Optimization*, pages 60–70, 2001.

[12] J. Fakcharoenphol, C. Harrelson, and S. Rao. The k-traveling repairman problem. In *Proceedings of the* 14^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003.

[13] E. Feuerstein and L. Stougie. On-line single-server dial-a-ride problems. *Theoretical Computer Science*, 268(1):91–105, 2001.

[14] M. Fischetti, G. Laporte, and S. Martello. The delivery man problem and cumulative matroids. *Operations Research*, 41:1065–1064, 1993.

[15] F. V. Fomin and A. Lingas. Approximation algorithms for time-dependent orienteering. *Information Processing Letters*, 83:57–62, 2002.

[16] H. N. Gabow and S. Pettie. The dynamic vertex minimum problem and its application to clustering-type approximation algorithms. In 8^{th} *Scandinavian Workshop on Algorithm Theory*, pages 190–199, 2002.

[17] N. Garg. A 3-approximation for the minimum tree spanning k vertices. In *Proceedings of the* 37^{th} *Annual Symposium on Foundations of Computer Science*, pages 302–309, 1996.

[18] N. Garg. Personal communication, 1999.

[19] M. Goemans and J. Kleinberg. An improved approximation ratio for the minimum latency problem. *Mathematical Programming*, 82:111–124, 1998.

[20] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24:296–317, 1995.

[21] E. Koutsoupias, C. H. Papadimitriou, and M. Yannakakis. Searching a fixed graph. In *Proceedings of the* 23^{rd}*International Colloquium on Automata, Languages, and Programming*, pages 280–289, 1996.

[22] S. O. Krumke, W. E. de Paepe, D. Poensgen, and L. Stougie. News from the online traveling repairman. In *Proceedings of the* 26^{th} *International Symposium on Mathematical Foundations of Computer Science*, pages 487–499, 2001.

[23] A. Lucena. Time-dependent traveling salesman problem - the deliveryman case. *Networks*, 20(6):753–763, 1990.

[24] E. Minieka. The delivery man problem on a tree network. *Annals of Operations Research*, 18:261–266, 1989.

[25] C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 18:1–11, 1993.

[26] J.-C. Picard and M. Queyranne. The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *Operations Research*, 26:86–110, 1978.

[27] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23:555–565, 1976.

[28] D. Simchi-Levi and O. Berman. Minimizing the total flow time of n jobs on a network. *IIE Transactions*, 23(3):236–244, September 1991.

[29] R. Sitters. The minimum latency problem is NP-hard for weighted trees. In *Proceedings of the* 9^{th} *Conference on Integer Programming and Combinatorial Optimization*, pages 230–239, 2002.

[30] J. N. Tsitsiklis. Special cases of traveling salesman and repairman problems with time windows. *Networks*, 22:263–282, 1992.

[31] R. J. Vander Wiel and N. V. Sahinidis. Heuristic bounds and test problem generation for the time-dependent traveling salesman problem. *Transportation Science*, 29(2):167–183, May 1995.

[32] R. J. Vander Wiel and N. V. Sahinidis. An exact solution approach for the time-dependent traveling-salesman problem. *Naval Research Logistics*, 43:797–820, 1996.

[33] I. R. Webb. Depth-first solutions for the deliveryman problem on tree-like networks: an evaluation using a permutation model. *Transportation Science*, 30(2):134–147, May 1996.

[34] B. Wu. Polynomial time algorithms for some minimum latency problems. *Information Processing Letters*, 75:225–229, 2000.

[35] C. Yang. A dynamic programming algorithm for the travelling repairman problem. *Asia-Pacific Journal of Operations Research*, 6:192–206, 1989.