An Improved Approximation Ratio for the Minimum Latency Problem

Michel Goemans* Jon Kleinberg†

Abstract

Given a tour visiting n points in a metric space, the *latency* of one of these points p is the distance traveled in the tour before reaching p. The *minimum latency problem* asks for a tour passing through n given points for which the total latency of the n points is minimum; in effect, we are seeking the tour with minimum average "arrival time." This problem has been studied in the operations research literature, where it has also been termed the "delivery-man problem" and the "traveling repairman problem."

The approximability of the minimum latency problem was first considered by Sahni and Gonzalez in 1976; however, unlike the classical traveling salesman problem, it is not easy to give any constant-factor approximation algorithm for the minimum latency problem. Recently, Blum, Chalasani, Coppersmith, Pulleyblank, Raghavan, and Sudan gave the first such algorithm, obtaining an approximation ratio of 144. In this work, we present an algorithm which improves this ratio to 21.55. The development of our algorithm involves a number of techniques that seem to be of interest from the perspective of the traveling salesman problem and its variants more generally.

1 Introduction

We consider the *minimum latency problem* (MLP): we are given a metric space M on n points $\{v_1, \cdots, v_n\}$, and we wish to find a tour in M rooted at v_1 which minimizes the sum of the arrival times at the n points. More concretely, if T is a tour rooted at v_1, we say that the *latency* of v_i with respect to T is the distance traveled in T before reaching v_i; and the latency of T is the sum of the latencies of the n points. The goal then is to find a tour of minimum latency. This problem is NP-complete by a simple reduction from, for example, the Hamiltonian cycle problem.

Despite the obvious similarities to the classical traveling salesman problem (TSP), the MLP appears to be much less well-behaved from a computational point of view. As discussed

*Department of Mathematics, MIT, Cambridge, MA 02139. Supported by NSF contract 9302476-CCR and an NEC research grant.

†Laboratory for Computer Science, MIT, Cambridge, MA 02139. Author supported by an ONR Graduate Fellowship. Present address: Department of Computer Science, Cornell University, Ithaca NY 14853. Email: kleinber@cs.cornell.edu. On leave at IBM Almaden Research Center, San Jose CA 95120.

in e.g. [5, 12, 15], various local changes in the input points can lead to highly non-local changes in the optimum solution. For example, even when the input points lie on a line, the optimum MLP tour may cross itself many times in a back-and-forth pattern — a phenomenon not encountered in the TSP. For weighted trees, on which the TSP is trivial, no polynomial-time algorithm is known for the MLP. Finally, although the approximability of the MLP was already considered by Sahni and Gonzalez in [13], there is no simple heuristic known which gives a constant-factor approximation; the first approximation algorithm for the MLP was only recently given by Blum, Chalasani, Coppersmith, Pulleyblank, Raghavan, and Sudan [5], who obtained a ratio of 144.

Our main result here is an improved approximation ratio of 21.55 for the MLP; we also obtain a 3.5912-approximation for the MLP on weighted trees, improving the bound of 8 from [5].

We are interested in approximation algorithms for this problem for a number of reasons. First of all, the MLP is a reasonably well-studied problem in the operations research literature (e.g. [1, 8, 12, 13, 15]), where it is also known as the "delivery-man problem" and the "traveling repairman problem." The problem is also of interest from the point of view of on-line search problems; for example, as noted by Rivest and Yin (reported in [5]), if one is a searching for a goal that is equally likely to be at one of n places in a metric space M, then the minimum latency tour is the one with the minimum expected time to find the goal. Indeed, the optimum solutions to the MLP in a number of special cases have structure similar to the on-line search patterns constructed by Baeza-Yates, Culberson, and Rawlins [3].

Another source of interest in this problem comes from the traveling salesman problem itself. Despite the significant differences between the TSP and MLP, the constituent parts of our approximation algorithm involve techniques that we feel may be of interest in understanding the structure of the TSP and its variants more generally. Our algorithm has the same global structure as that of [5], based on concatenating tours on larger and larger subsets of the vertices; our improvement on their bound comes both from the way in which we find these "partial tours" and the way in which we combine them to produce the final tour. This is discussed more fully in the following section.

1.1 Overview.

The general idea behind obtaining a constant-factor approximation is to visit vertices close to v_1 (the root of the tour) as early as possible. First let us consider how this would work in the case of a weighted tree. Here, one can solve the following k-TSP problem optimally: find a minimum-length tour, rooted at v_1, that visits at least k points. Thus, if one has an optimum k-TSP tour for each value of k, it would be natural to look for some concatenation of these as the final tour to output. This was the approach taken by [5], who showed that there is such a concatenation that gives a bound of 8.

In fact, we show in Section 2 that with respect to a certain lower bound on the value of the optimum, one can find the best possible way to concatenate the tours returned by a k-TSP subroutine. This is done by solving a certain shortest-path problem on a graph whose vertex set is the set of k-TSP tours for each value of k. We give (Theorem 3) a tight analysis of the largest possible gap between this shortest-path formulation and the length of the optimum tour; it increases (as n tends to infinity) to a supremum of 3.5912, which we therefore obtain as our approximation ratio. The value of 3.5912 stands for the unique root of $\gamma \ln(\gamma) = \gamma + 1$.

In a general metric space, however, the k-TSP is NP-complete, and so the above approach must be modified. In particular, we must identify a different set of tours to concatenate. In [5], as well as in the conference version of the present paper, no constant-factor approximation was known for the k-TSP, and so [5] introduced the notion of an (α, β)-TSP-approximator, which is defined as follows.

Definition 1 *Assume that the maximum number of vertices that can be visited by a subtour of length at most L starting from a fixed root vertex is $n - a(L)$. Then an (α, β)-TSP-approximator is a polynomial-time algorithm that given a bound L finds a subtour of length at most βL visiting at least $n - \alpha a(L)$ vertices.*

Blum et al. showed how such an approximator can essentially be used as a subroutine in place of a constant-factor approximation for the k-TSP; given an (α, β)-TSP-approximator, they obtain an $8\lceil\alpha\rceil\beta$–approximation for the MLP. Using the techniques of Section 2, we strengthen this general bound to $\gamma\lceil\alpha\rceil\beta$, where again $\gamma \sim 3.5912$.

Thus the question is to determine the smallest vales of α and β for which one can obtain an (α, β)-TSP-approximator. Blum et al. show how to obtain a $(3, 6)$-TSP-approximator from a 2-approximation algorithm for the “prize-collecting TSP” due to Goemans and Williamson [11]. In Section 3, when we treat the case of a general metric space, we show how a stronger analysis of the the prize-collecting TSP algorithm of [11] allows one to obtain a $(2, 4)$-TSP-approximator. In addition, working more directly from a linear programming relaxation, we obtain a $(2, 3)$-TSP-approximator. By the discussion of the previous paragraph, this provides us with a 21.55-approximation algorithm.

Since the publication of the conference version of this paper, two constant–factor approximations for the k-TSP have been obtained: one due to Blum, Ravi, and Vempala [7], and a second due to Garg [9]. Garg's algorithm provides a 3-approximation to the k-TSP, and hence constitutes a $(1, 3)$-TSP-approximator. Plugging these values of α and β into the general bound of this paper gives an approximation ratio of 10.78; this is currently the best approximation ratio known for the MLP.

2 The Minimum Latency Problem on Trees

Recall that we are given points $V = \{v_1, \ldots, v_n\} \subset M$, and we wish to find a tour rooted at v_1 of minimum total latency. Throughout this section, we assume that M is the shortest-path metric of a weighted tree. That is, we are given a tree T with positive weights on its edges, and the distance between any two vertices is the weight of the path between them. For this problem, the best previously known approximation ratio was 8 [5].

Our algorithm will make use of solutions to the k-traveling salesman problem (k-TSP) in which one wants to find the shortest tour, starting and ending at v_1, which meets at least k points of V. For tree metrics Blum et al. show how to solve the k-TSP exactly in polynomial time [5].

As a first step, our algorithm will solve the k-TSP on V for $k = 2, 3, \ldots, n$. Denote the tours obtained by $T_2, T_3, \ldots, T_n$, with lengths $d_2 \leq d_3 \leq \cdots \leq d_n$. We skip the value $k = 1$ since this corresponds to simply visiting v_1 at a cost of 0. Given an increasing set of indices

$$1 < j_1 < j_2 < \cdots < j_m = n,$$

we define the *concatenated tour* $T = T_{j_1} T_{j_2} \cdots T_{j_m}$ as follows. Starting from v_1, we traverse T_{j_1}, then T_{j_2}, and so on up to T_{j_m}, while (i) introducing short-cuts to eliminate redundant visits to points, and (ii) traversing the subtour T_{j_i} in the direction that minimizes the total latency of the previously unvisited points in this subtour.

Consider the following upper bound on the total latency of the tour $T = T_{j_1} T_{j_2} \cdots T_{j_m}$. Suppose that p_i points are first visited in the subtour T_{j_i}, and write

$$q_i = \sum_{\ell \leq i} p_\ell.$$

Note that $p_i \leq j_i \leq q_i$. We should emphasize that p_i is incomparable to $j_i - j_{i-1}$ since the tours T_{j_i} are not necessarily nested. However, we have the following fact. (For notational reasons we write $j_0 = 0$ and $q_0 = 0$.)

Claim 1

$$\sum_{i=1}^{m} p_i d_{j_i} \leq \sum_{i=1}^{m} (j_i - j_{i-1}) d_{j_i}.$$

Proof. Since $\sum_{i=1}^m p_i = n = \sum_{i=1}^m (j_i - j_{i-1})$, both sides of the inequality can be written as a sum of n terms, each of which is one of $\{d_{j_1}, \ldots, d_{j_m}\}$. The ℓ^{th} smallest term of the left-hand side is equal to d_{j_i}, where $q_{i-1} < \ell \leq q_i$; the ℓ^{th} smallest term of the right-hand side is equal to $d_{j_{i'}}$, where $j_{i'-1} < \ell \leq j_{i'}$. But since $\ell > q_{i-1} \geq j_{i-1}$, we know $i' \geq i$, and hence $d_{j_{i'}} \geq d_{j_i}$, from which the result follows. ∎

Now, the subtour T_{j_i} adds d_{j_i} to the latency of each of the $n - q_i$ points remaining to be visited in later subtours, and adds at most a total of $\frac{1}{2} p_i d_{j_i}$ to the latencies of points first visited in T_{j_i} (since we traverse T_{j_i} in the order minimizing this total). Thus, the total latency of T is at most

$$\begin{aligned} &\sum_i (n - q_i) d_{j_i} + \frac{1}{2} \sum_i p_i d_{j_i} \\ &\quad \leq \quad \sum_i (n - j_i) d_{j_i} + \frac{1}{2} \sum_i (j_i - j_{i-1}) d_{j_i} \\ &\quad = \quad \sum_i \left(n - \frac{j_{i-1} + j_i}{2} \right) d_{j_i} \qquad (1) \end{aligned}$$

where the inequality follows from Claim 1 and the fact that $q_i \geq j_i$.

There is another very useful way to rewrite the upper bound $\sum_i (n - j_i) d_{j_i} + \frac{1}{2} \sum_i (j_i - j_{i-1}) d_{j_i}$. List the points in the order in which they are first reached by the tour. Let the "modified latency" π_k of the k^{th} point be

$$\pi_k = \frac{1}{2} d_{j_i} + \sum_{t=1}^{i-1} d_{j_t}$$

for $j_{i-1} < k \leq j_i$. Then

$$\sum_i (n - j_i) d_{j_i} + \frac{1}{2} \sum_i (j_i - j_{i-1}) d_{j_i} = \sum_{k=1}^{n} \pi_k.$$

In effect, the worst case for our algorithm occurs when all tours are nested by inclusion and all points are reached halfway through the tour that first visits them — the modified latency π_k is simply the value of the latency of the k^{th} point in this worst case.

The benefit of the right-hand-side of inequality (1) is that we now have an upper bound for the total latency of $T_{j_1} T_{j_2} \cdots T_{j_m}$ solely in terms of the indices $j_1, \ldots, j_m$ (and the d_i's). Given this, our algorithm to approximate the minimum latency tour for V is as follows.

(i) For $k = 2, 3, \ldots, n$, compute T_k, the minimum-length k-TSP tour on V rooted at v_1. Let d_k denote the length of T_k.

(ii) Let G_n denote the complete graph on the vertex set $\{1, 2, \ldots, n\}$; turn G_n into a directed graph by orienting the edge (i, j) from $\min(i, j)$ to $\max(i, j)$.

(iii) Assign a length function to each directed arc of G_n; the length of arc (i, j) will be

$$\left(n - \frac{i+j}{2}\right) d_j.$$

(iv) Compute the shortest 1-n path in G_n; suppose that it goes through vertices

$$1 = j_0 < j_1 < \cdots < j_m = n.$$

(v) Output the concatenated tour

$$T = T_{j_1}, T_{j_2}, \ldots, T_{j_m}.$$

By the discussion leading up to inequality (1), we have the following

Lemma 1 *The total latency of the concatenated tour $T_{j_1} T_{j_2} \cdots T_{j_m}$ is at most the length of the path $j_0, j_1, j_2, \ldots, j_m$ in the graph G_n.*

We can also give a lower bound on the total latency of the optimum tour in terms of the d_i's.

Lemma 2 *Let l_k^* denote the latency of the k^{th} vertex of the optimum tour. Then $l_k^* \geq d_k/2$, and the optimum total latency is lower bounded by $\frac{1}{2} \sum_{k=2}^{n} d_k$.*

Proof. By doubling the path from the root to the k^{th} vertex of the optimum tour, we obtain a tour of length at most $2l_k^*$ visiting at least k vertices. This implies that $d_k \leq 2l_k^*$ and the bound on the total latency follows. ∎

Lemmas 1 and 2 together imply that our algorithm is a ρ-approximation algorithm if we can prove that, for any n and any non-decreasing sequence of d_i's, the ratio of the length of the shortest path in G_n to $\frac{1}{2} \sum_{k=2}^{n} d_k$ is at most ρ.

It turns out that we can precisely determine the worst-case value of this ratio ρ; the bound for the algorithm then follows as a corollary. For a given edge-weighted graph G_n, let $\sigma(G_n)$ denote the length of the shortest 1-n path in G_n; and let τ_n denote the worst-case ratio of $\sigma(G_n)$ to $\frac{1}{2}\sum_{k=2}^{n} d_k$ over all non-decreasing sequences of d_i. The following theorem gives an upper bound on τ_n that holds for all n, and additionally shows that this is the tightest such bound possible. This latter argument is not necessary for the analysis of the algorithm, only to show that our bound for τ_n is tight.

Theorem 3 *For all n,*

$$\tau_n = \sup_{d_2 \leq \cdots \leq d_n} \frac{2\sigma(G_n)}{\sum_{k=2}^{n} d_k} \leq \gamma < 3.5912,$$

where γ is the unique root of $\gamma \ln(\gamma) = \gamma + 1$. Moreover, $\sup_n \tau_n = \gamma$.

Corollary 4 *The above algorithm is a γ-approximation algorithm for the MLP on trees, where $\gamma < 3.5912$ is the unique root of $\gamma \ln(\gamma) = \gamma + 1$.*

Proof of Theorem 3. Consider any non-decreasing sequence of d_i's. Assume that $d_2 = 1$. We shall construct a path in G_n and compare its length to $\frac{1}{2}\sum_{k=2}^{n} d_k$.

Fix $c > 1$ and $1 \leq L_0 < c$; both c and L_0 will be determined later. For $i = 1, \ldots$, let j_i denote the maximum element of $\{k : d_k \leq L_0 c^{i-1}\}$. For some large enough value of m, $j_m = n$. Consider the path $j_0, j_1, \ldots, j_m$ in G_n. Its length is equal to

$$\sum_{i=1}^{m} \left(n - \frac{j_{i-1} + j_i}{2}\right) d_{j_i}.$$

This can also be expressed as $\sum_{k=1}^{n} \pi_k$, where the "modified latency" π_k of the k^{th} point for $j_{i-1} < k \leq j_i$ was defined as

$$\pi_k = \frac{1}{2} d_{j_i} + \sum_{t=1}^{i-1} d_{j_t}.$$

By definition, $d_{j_i} \leq L = L_0 c^s$ for $s = \lceil \log_c(d_k/L_0) \rceil$. Observe that $s \geq 0$ since $d_k \geq 1$ and $L_0 < c$. We can upper bound the modified latency π_k by

$$\begin{aligned} \pi_k &\leq \frac{1}{2}L + \frac{L}{c} + \frac{L}{c^2} + \ldots \\ &= \left(\frac{c}{c-1} - \frac{1}{2}\right) L \\ &= \left(\frac{c+1}{2(c-1)}\right) L. \end{aligned}$$

We could already relate π_k to d_k by observing that $L \leq c d_k$ by definition. This shows that $\pi_k \leq \frac{c(c+1)}{c-1} \frac{d_k}{2}$, which implies that the length of the path we have constructed is at most $c(c+1)/(c-1)$ times the value $\sum_k d_k/2$.

However, we can obtain a better bound by carefully selecting L_0. We use a probabilistic argument. Let $L_0 = c^U$, where U is a random variable uniformly distributed between 0 and 1.

This therefore defines a random path in G_n, and its expected length is equal to $\sum_{k=1}^{n} E[\pi_k]$. The expected value of π_k is at most

$$\frac{c+1}{2(c-1)}E[L].$$

We now compute $E[L]$. Observe that L/d_k is a random variable of the form c^Y where Y is uniform between 0 and 1. Hence,

$$E[L] = d_k E[c^Y] = d_k \int_0^1 c^x \, dx = d_k \frac{c-1}{\ln c}.$$

Thus

$$E[\pi_k] \leq \frac{c+1}{2\ln c} d_k,$$

and the expected length of our random path is at most $\frac{c+1}{\ln c}$ times $\sum_k d_k/2$. This value is optimized by setting c to be equal to the root of $c\ln(c) - c - 1 = 0$, which turns out to be $\gamma = 3.59112142\cdots$. This proves the upper bound.

We prove the second claim of the theorem as follows. First, it is not difficult to show that γ has the property that for all $x > 0$,

$$\gamma \ln x \leq 1 + x. \tag{2}$$

Now fix $n \geq 1$, and define $d_2, \ldots, d_n$ by setting $d_n = 1$ and $d_i = 1/(n-i)$ for $2 \leq i \leq n-1$. This gives us a weighted graph G_n.

We claim that $\sigma(G_n) \geq \frac{1}{2}(\gamma \ln(n-1) - 1)$. To prove this, it is sufficient to define a number ψ_i for each $i = 1, \ldots, n$ such that $\psi_n = 0$, $\psi_1 = \frac{1}{2}(\gamma \ln(n-1) - 1)$, and for $i < j$, we have

$$\begin{aligned} \psi_i - \psi_j &\leq \text{length of arc } (i,j) \qquad (3) \\ &= \left(n - \frac{i+j}{2}\right) d_j. \end{aligned}$$

These conditions indeed imply that ψ_i is a lower bound on the length of the shortest path from i to n. We define $\psi_n = 0$ and $\psi_i = \frac{1}{2}(\gamma \ln(n-i) - 1)$ for $i = 1, \ldots, n-1$. We now verify Equation (4) in the following two cases.

(i) For $j = n$, we have

$$\begin{aligned} \psi_i - \psi_n &= \frac{1}{2}(\gamma \ln(n-i) - 1) \\ &\leq \frac{1}{2}(n-i) \\ &= \text{length of arc } (i,n), \end{aligned}$$

with the inequality following from (2).

(ii) For $j < n$, we have

$$\psi_i - \psi_j = \frac{1}{2}\left[\gamma \ln\left(\frac{n-i}{n-j}\right)\right]$$

$$
\begin{aligned}
&\leq \frac{1}{2}\left(1+\frac{n-i}{n-j}\right)\\
&= \left(n-\frac{i+j}{2}\right)\cdot\frac{1}{n-j}\\
&= \text{length of arc } (i,j),
\end{aligned}
$$

with the inequality again following from (2).

Now we have

$$\sup_n \tau_n \geq \sup_n \frac{2\sigma(G_n)}{\sum_{i=2}^n d_i} \geq \sup_n \frac{\gamma\ln(n-1)-1}{1+H(n-2)} \geq \gamma,$$

where $H(\cdot)$ denotes the harmonic function. ∎

Observe that we don't really need to compute the *shortest* path in G_n in order to obtain a 3.5912-approximation algorithm; the (random) path constructed in the proof of the theorem is sufficient. This leads to a simple randomized 3.5912-approximation algorithm for the MLP on trees. Furthermore, this algorithm can be derandomized by discretizing the probability distribution. More precisely, by choosing the best path among p paths corresponding to $L_0 = c^{k/p}$ for $k = 0,\ldots,p-1$, the resulting bound can be seen to be equal to

$$\frac{(c+1)c^{1/p}}{p(c^{1/p}-1)},$$

which can be made arbitrarily close to $\gamma = (\gamma+1)/\ln(\gamma)$ for p arbitrarily large and c arbitrarily close to γ.

A final observation is as follows. One can show that for each n, the computation of τ_n can be formulated as a linear program with $O(n)$ variables and $O(n^2)$ constraints. We have computed a few values of τ_n and found that it converges to γ relatively slowly; for example, $\tau_{20} = 2.63362\cdots$, $\tau_{160} = 3.07745\cdots$ and $\tau_{300} = 3.15522\cdots$. Of course, this means that for small trees, our approximation algorithm is provably achieving a performance ratio noticeably better than γ.

3 General Metric Spaces

Now we consider the case in which the set V of n points is a subset of an arbitrary metric space M. The approach of the previous section must be modified, since the k-TSP problem is NP-complete in general metric spaces. However, the techniques we have developed provide much of what we need. Blum et al. observed that it is enough to simply bound the latencies of the final $\frac{1}{\alpha}$ fraction of the points, for some constant α, in order to obtain an approximation algorithm to the MLP [5]. This motivated their definition of an (α,β)-TSP-approximator given in the introduction. They showed that any (α,β)-TSP-approximator gives a $8\lceil\alpha\rceil\beta$-approximation algorithm for the MLP in general metric spaces. (In [5], a method was also claimed to obtain a $4\lceil\alpha\rceil\beta$ approximation, but this proved to be incorrect.)

In [5] it was shown that there exists a $(3,6)$-TSP-approximator, leading to a 144-approximation algorithm for the MLP; the authors of [5] have subsequently observed that their technique also provides a $(4,4)$-TSP-approximator, which lowers this bound to 128. In what follows, we provide improved (α,β)-TSP-approximators. This by itself improves on the ratio of [5], and combined with the techniques of the previous section brings the ratio down to 21.55.

Theorem 5 *There exists a $(2,4)$-TSP-approximator.*

Proof. As in [5], we obtain the algorithm from a 2-approximation algorithm to the prize-collecting TSP due to Goemans and Williamson [11]. In the prize-collecting TSP, each vertex v has a penalty π_v, and the goal is to find a subtour from the root minimizing the length of the subtour plus the sum of the penalties of the vertices not visited.

By looking at the proof of Goemans and Williamson, one verifies that the GW algorithm produces a subtour such that the length of the subtour plus *twice* the sum of the penalties of the vertices not visited is at most twice the cost of the optimum subtour (i.e. length plus penalties of unvisited vertices). We note in passing that this observation does not improve the performance guarantee for the prize-collecting TSP.

Recall that $a(L)$ is the minimum number such that there is a tour of length L which visits $n - a(L)$ vertices of V. For $i = 1, \ldots, n$, call the algorithm of [11] with all penalties set to L/i; let us the denote the length of the resulting subtour by $D(i)$ and the number of unvisited vertices by $b(i)$. By the observation of the previous paragraph, we must have that

$$D(i) + 2b(i)L/i \leq 2(L + a(L)L/i).$$

That is,

$$\frac{D(i)}{L} + 2\frac{b(i)}{i} \leq 2 + 2\frac{a(L)}{i}. \tag{4}$$

Let k be the smallest value of i such that the left-hand-side of the above inequality is at most 4. Since $a(L)$ satisfies this condition (because of the right-hand-side of (4)), we have that $k \leq a(L)$. Therefore, (4) implies:

$$\frac{D(k)}{L} + 2\frac{b(k)}{a(L)} \leq \frac{D(k)}{L} + 2\frac{b(k)}{k} \leq 4. \tag{5}$$

This means that $b(k) \leq 2a(L)$ and that $D(k) \leq 4L$, showing that the algorithm is a $(2,4)$-TSP-approximator. ∎

We note that this proof (and, in particular, inequality (5)) shows that this algorithm is an (α,β)-TSP-approximator for some unknown values of α and β satisfying $2\alpha + \beta \leq 4$. However, we don't know how to exploit this fact.

We can give an even better TSP-approximator by using linear programming techniques in the spirit of Bienstock et al. [4] for the prize-collecting TSP.

Theorem 6 *There exists a $(2,3)$-TSP-approximator.*

Proof. We view V as a complete graph on n vertices, with the weight c_e of the edge $e = (v_i, v_j)$ denoting the distance between v_i and v_j. We adapt a standard linear programming relaxation of the traveling salesman problem to the setting of the k-TSP, and from this derive a $(2,3)$-TSP-approximator. For each edge e, we define a variable $x_e \in [0,1]$, and for each vertex v_i we define a variable $y_i \in [0,1]$. Intuitively, $x_e = 1$ indicates that edge e will be used by the tour, and $y_i = 1$ indicates that vertex v_i will be visited by the tour.

Now, for a set $S \subset V$, let $\delta(S)$ denote the set of edges with exactly one end in S, and $x(\delta(S))$ denote the sum $\sum_{e\in\delta(S)} x_e$. Consider the following family of linear programs (LP_k), for $k = 1, \ldots, n$.

$$\begin{array}{lll} (LP_k) & \text{Min} & \sum_e c_e x_e \\ & \text{s.t.} & x(\delta(S)) \geq 2y_i \qquad (v_1 \in S,\ v_i \notin S) \\ & & \sum_i y_i = k \\ & & 0 \leq x_e \leq 1 \\ & & 0 \leq y_i \leq 1 \\ & & y_1 = 1. \end{array}$$

Observe that any k-TSP tour gives a feasible solution to (LP_k). Indeed, the first set of constraints says that if vertex i is visited then there are at least 2 edges of the tour in $\delta(S)$. The second constraint says that k vertices are visited. Since x and y may take fractional values, the optimum value of (LP_k) has a value $z_k \leq d_k$.

We use the optimum solution to (LP_k) to produce a tour that passes through at least $m < k$ vertices, as follows. First, we choose the m vertices whose y-values are the largest. Let W denote this set of vertices. The m^{th} largest y-value is at least t, where

$$(m-1) + t(n-m+1) \geq k$$

i.e.

$$t \geq \frac{1+k-m}{1+n-m} \geq \frac{k-m}{n-m}.$$

We multiply our solution to (LP_k) by $\frac{1}{t}$, obtaining x-values which satisfy

$$x(\delta(S)) \geq 2 \qquad (v_1 \in S,\ W - S \neq \emptyset).$$

The value of the objective function is now $\frac{z_k}{t}$. By a result of Goemans and Bertsimas [10] (see also [4]), the value of the objective function is unchanged if we include the constraints

$$x(\delta(\{v_i\})) = \begin{cases} 2 & v_i \in W \\ 0 & v_i \notin W. \end{cases}$$

We now have a standard linear programming relaxation of the traveling salesman problem, written for the k vertices in W. Results of Wolsey [16] and Shmoys and Williamson [14] show that if we apply Christofides's heuristic to produce a tour on the vertices in W, the length of this tour will be at most $\frac{3}{2}$ times the optimum value $\frac{z_k}{t}$ of this linear program. Thus, we obtain a tour on the vertices in W of length at most

$$\frac{3}{2} \cdot \frac{z_k}{t} \leq \frac{3(n-m)}{2(k-m)} \cdot d_k \tag{6}$$

Using the technique above, we now obtain an $(a, \frac{3a}{2a-2})$-TSP-approximator, for each $a > 1$. We are given a bound L, and suppose that the greatest number of points that can be reached by a tour of length L is $(1-\varepsilon)n$. Thus when $k = (1-\varepsilon)n$, $d_k \leq L < d_{k+1}$. We set $m = (1-a\varepsilon)n$ and run the above algorithm. By Equation (6), we obtain a tour of length at most

$$\frac{3n[1-(1-a\varepsilon)]}{2n[(1-\varepsilon)-(1-a\varepsilon)]} \cdot d_k \leq \frac{3a}{2a-2} \cdot L.$$

Of course we don't know that value of ε, but we can try each possible value of k and take the tour of length at most $\frac{3aL}{2a-2}$ that reaches the greatest number of vertices.

Setting $a = 2$ gives us a $(2, 3)$-TSP-approximator, and this choice of a minimizes the product $\alpha\beta = \frac{3a^2}{2a-2}$. ∎

We can now obtain a $\lceil\alpha\rceil\beta\gamma$-approximation algorithm for the MLP, where $\gamma < 3.5912$ was defined in the previous section. In fact, there are two ways to do this — one can give a direct randomized construction of such an approximately optimal tour, along the lines of the proof of Theorem 3; or one can exploit the way in which our $(2, 3)$-TSP-approximator works to set up a shortest-path problem as was done in the previous section. For the sake of brevity, we choose the former; however, we briefly discuss the latter approach below.

The randomized algorithm is as follows. Let $L_0 = c^U$, where U is uniformly distributed between 0 and 1. Let T_i denote the tour obtained by calling the $(2, 3)$-TSP-approximator with input $L_0 c^{i-1}$; suppose that T_m is a tour spanning V. Our algorithm returns the concatenated tour $T = T_1 \cdots T_m$.

Theorem 7 *For any (α, β)-TSP-approximator, the above algorithm achieves an (expected) approximation ratio of at most $\lceil\alpha\rceil\beta\frac{c+1}{\ln c}$. For $(\alpha, \beta) = (2, 3)$ and $c = 3.5912$, the resulting bound is less than 21.55.*

Proof. The argument is essentially the same as that used in the proof of Theorem 3, except that we are now using a TSP-approximator instead of solving the k-TSP exactly. Let l^*_{n-i} denote the latency of the $(n-i)^{\text{th}}$ vertex of the optimum tour. For $s = \lceil\log_c(2l^*_{n-i}/L_0)\rceil$, we know the existence of a tour visiting $n-i$ vertices and of length at most $2l^*_{n-i} \leq L = L_0 c^s$, so the TSP-approximator with input L returns a subtour of length at most βL which visits at least $n - \alpha i$ vertices. The notion of "modified latency" can still be used in this setting; as in Theorem 3, the modified latency π_{n-i} is at most $\frac{c+1}{2(c-1)}L$, and its expected value is at most

$$\beta\frac{c+1}{\ln c}l^*_{n-i}.$$

If the approximator was guaranteed to return a tour on at least $n-i$ vertices, the performance guarantee would be $\beta\frac{c+1}{\ln c}$. However, we only know that the subtour visits at least $n - \alpha i$

vertices. We can thus charge the latency in our tour of the vertices indexed $n - \lceil\alpha\rceil(i+1) + 1, \ldots, n - \lceil\alpha\rceil i$ to the modified latency π_{n-i}. In the process, $\lceil\alpha\rceil$ vertices are charged to π_{n-i}, proving the desired bound. ∎

Using the remark following the proof of Theorem 3, the algorithm does not need to be randomized. Again, we could select L_0 to be of the form $c^{k/p}$ for $k = 0, \cdots, p-1$, and this leads to a performance guarantee of

$$\lceil\alpha\rceil\beta\frac{(c+1)c^{1/p}}{p(c^{1/p}-1)}.$$

Also, as mentioned above, our $(2,3)$-TSP-approximator has a special feature which allows us to give another deterministic 6γ-approximation algorithm, following more closely our algorithm for the MLP on trees. Instead of being given L, our $(2,3)$-TSP-approximator can instead receive a value $k > n/2$ and output a tour on $m = n - 2(n-k) = 2k - n$ vertices and of length at most three times that of the optimum k-TSP (see inequality (6)). We can compute all these tours and set up a directed graph G_n with the length of the arc (i,j) being $n - \frac{i+j}{2}$ times the length of the tour on j or $j-1$ vertices output by our algorithm. A shortest path in this graph will then lead to a tour of total latency at most 6γ by arguments similar to those in the proof of Theorems 3 and 7.

4 Conclusion

We have improved the best known approximation ratio for the minimum latency problem from 144 to 21.55 in general metric spaces, and from 8 to 3.5912 in weighted trees. For general metric spaces, as mentioned in the introduction, the 3-approximation for the k-TSP due to Garg [9] implies a yet better approximation ratio of $3\gamma < 10.78$.

A number of further directions for inquiry are raised by the techniques used here. Directly related to the MLP is the problem of improving on the approximation ratios of γ and 3γ for the case of weighted trees and general metric spaces. It is also worth noting that the MLP is not known to be NP-hard in weighted trees, so it is worth considering whether it could be solved optimally. Another line of questions is related to (α,β)-TSP-approximators, which we believe to be of interest in their own right. What is the range of (α,β) for which a polynomial-time (α,β)-TSP-approximator exists?

The notion of concatenating tours in lengths that increase geometrically is a method that is well-known in a number of settings — for example, in the design of on-line algorithms (see e.g. [3]). It would be interesting to consider other applications of the idea of choosing a random initial starting point for the geometric sequence.

Finally, there remain a number of natural variants of the traveling salesman problem for which it is unknown whether there exists a constant-factor approximation algorithm. It is possible that some of the techniques developed here could be useful in attacking such problems.

Acknowledgements.

We have benefited from several stimulating discussions with Yuval Rabani. Thanks also to the referees for their comments.

References

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

[2] B. Awerbuch, Y. Azar, A. Blum and S. Vempala, "Improved approximation guarantees for minimum-weight k-trees and prize-collecting salesmen," *Proc. 27th ACM Symposium on the Theory of Computing*, 1995, pp. 277-283.

[3] R. Baeza-Yates, J. Culberson and G. Rawlins, "Searching in the plane," *Information and Computation*, 106(1993), pp. 234–252.

[4] D. Bienstock, M. Goemans, D. Simchi-Levi and D. Williamson, "A note on the prize-collecting traveling salesman problem," *Mathematical Programming*, 59(1993), pp. 413–420.

[5] A. Blum, P. Chalasani, D. Coppersmith, W. Pulleyblank, P. Raghavan and M. Sudan, "The minimum latency problem," *Proc. 26th ACM Symposium on the Theory of Computing*, 1994, pp. 163–171.

[6] A. Blum, P. Chalasani and S. Vempala, "A constant-factor approximation for the k-MST problem in the plane", *Proc. 27th ACM Symposium on the Theory of Computing*, 1995, pp. 294–302.

[7] A. Blum, R. Ravi, S. Vempala, "A constant–factor approximation for the k-MST problem," *Proc. 28th ACM Symposium on the Theory of Computing*, 1996.

[8] M. Fischetti, G. Laporte and S. Martello, "The delivery man problem and cumulative matroids", *Operations Research*, 41(1993), pp. 1055–1064.

[9] N. Garg, "A 3-approximation for the minimum tree spanning k vertices," *Proc. 37th IEEE Symposium on Foundations of Computer Science*, 1996.

[10] M. Goemans and D. Bertsimas, "Survivable networks, linear programming relaxations and the parsimonious property", *Mathematical Programming*, **60**, 145–166, 1993.

[11] M. Goemans and D. Williamson, "A general approximation technique for constrained forest problems," *SIAM J. Computing*, **24**, 296–317, 1995.

[12] E. Minieka, "The delivery man problem on a tree network," *Annals of Operations Research*, 18(1989), pp. 261–266.

[13] S. Sahni and T. Gonzalez, "P-complete approximation problems," *Journal of the ACM*, 23(1976), pp. 555–565.

[14] D. Shmoys and D. Williamson, "Analyzing the Held-Karp TSP bound: a monotonicity property with application," *Information Proc. Letters*, 35(1990), pp. 281–285.

[15] J. Tsitsiklis, "Special cases of the traveling salesman and repairman problems with time windows," *Networks*, 22(1992), pp. 263–282.

[16] L. Wolsey, "Heuristic analysis, linear programming, and branch and bound," *Math. Programming Study*, 13(1980), pp. 121–134.