Resource-Constrained Geometric Network Optimization

(Extended abstract)

Esther M. Arkin* Joseph S. B. Mitchell† Giri Narasimhan‡

December 18, 1997

Abstract

We study a variety of geometric network optimization problems on a set of points, in which we are given a resource bound, B, on the total length of the network, and our objective is to maximize the number of points visited (or the total "value" of points visited). In particular, we resolve the well-publicized open problem on the approximability of the *rooted* "orienteering problem" for the case in which the sites are given as points in the plane and the network required is a cycle. We obtain a 2-approximation for this problem. We also obtain approximation algorithms for variants of this problem in which the network required is a tree (3-approximation) or a path (2-approximation). No prior approximation bounds were known for any of these problems.

We also obtain improved approximation algorithms for geometric instances of the unrooted orienteering problem, where we obtain a 2-approximation for both the cycle and tree versions of the problem on points in the plane, as well as a 5-approximation for the tree version in edge-weighted graphs. Further, we study generalizations of the basic orienteering problem, to the case of multiple roots, sites that are polygonal regions, etc., where we again give the first known approximation results.

Our methods are based on some new tools which may be of interest in their own right: (1) some new results on m-guillotine subdivisions in the plane, strengthening the original approximation bound, and (2) some new combinatorial results on partitioning trees.

Our study helps to highlight where geometry "helps" in being able to approximate these hard combinatorial optimization problems.

*estie@ams.sunysb.edu; http://www.ams.sunysb.edu/~estie/. Department of Applied Mathematics and Statistics, State University of New York, Stony Brook, NY 11794-3600. Partially supported by NSF grant CCR-9504192.

†jsbm@ams.sunysb.edu; http://www.ams.sunysb.edu/~jsbm/. Department of Applied Mathematics and Statistics, State University of New York, Stony Brook, NY 11794-3600. Partially supported by NSF grant CCR-9504192, and by grants from Boeing Computer Services, Bridgeport Machines, Hughes Aircraft, and Sun Microsystems.

‡giri@msci.memphis.edu; http://www.msci.memphis.edu/~giri/. Department of Mathematical Sciences, University of Memphis, Memphis, TN 38152. Partially funded by NSF grant CCR-940-9752 and by a grant from Cadence Design Systems.

1 Introduction

Consider a traveling salesperson who has a given amount of gasoline (allowing travel up to distance B) and wants to maximize the number of sites visited before running out of gasoline. This problem is one of a class of *orienteering* problems that require us to design a network that visits a maximum number of nodes (sites), subject to an upper bound on the total length of the network.

In contrast with the classical traveling salesperson problem (TSP), which asks for a minimum-length cycle visiting *all* sites, in the orienteering problem there is a resource constraint on the length of the cycle and the problem is to do the "best possible" in the sense of maximizing the number of sites visited. In a sense, the orienteering problem is dual to the problem of minimizing the length of a cycle, subject to meeting a quota on the number of sites visited (in the "k-TSP" version) or the sum of the values of the sites visited (the "quota-driven salesperson" problem).

The orienteering problem has also been called the "bank robber" problem (a thief wishes to maximize his "haul", with a single tank of gas in the get-away car) and the "generalized traveling salesperson problem" [16, 28]. The name "orienteering" comes from the outdoor sport by the same name, in which each player is given a compass and a map of a given outdoor terrain, and has the goal to accumulate the largest score within a fixed amount of time. Players set out from a given root (the "home base"), and obtain a certain score (value) by visiting a subset of the n sites ("control points") before returning to the root.

In this paper, we distinguish between different versions of the orienteering problem according to the type of network that is to be constructed: In the *tree-orienteering* (*path-orienteering*, *cycle-orienteering*) problem, we must construct a tree (path, cycle, resp.) visiting a subset of a given set of n *sites*, subject to an upper bound B on the total length of the network. Additionally, we may have one or more *roots*, which are "essential" sites that we require the network to visit.

The *rooted* orienteering problem has, for several years, been a particularly challenging problem from the point of view of approximation algorithms — in fact, prior to our work, *no* approximation algorithms have been known for the problem, either in graphs or in geometry. Here, we provide *constant*-factor approximation algorithms for the geometric instances of the problem, in which the sites are given as points in the plane.

Summary of Main Results:

1. We give a 2-approximation algorithm for the *rooted* cycle- and path-orienteering problems on a set of points in the plane. This result is the first of its kind, as no approximation algorithms were previously known.
2. We give a 3-approximation algorithm for the rooted tree-orienteering problem, for points in the plane. Again, no prior approximation algorithms were known.
3. We give improved approximation algorithms for the unrooted instances of the orienteering problem. These results include a 2-approximation for the geometric tree- and cycle-orienteering, a 5-approximation for tree-orienteering in edge-weighted graphs.
4. We study the *multiply-rooted* versions of the orienteering problems in the plane, providing both an approximation algorithm when the number of roots is a constant, and a proof of hardness of approximability when the number of roots is part of the input.
5. We study geometric instances of the orienteering problems in which the sites are given as a set of *regions* (polygons) in the plane. We give $O(\log n)$-approximation algorithms for these versions.

Key to our methods are some new tools which may be of independent interest:

(1) We obtain new results on approximating an arbitrary connected planar subdivision with an m-guillotine subdivision, strengthening the original analysis given in Mitchell [23]; these new theorems seem to be crucial in addressing the types of approximation problems that arise in the rooted cases, and they may be more widely applicable.

(2) We obtain a variety of new combinatorial results on partitioning trees; these lemmas are critical in obtaining some of our improved bounds on the unrooted versions of the orienteering problems, as well as in analyzing the generalizations to non-point sites.

Related Work

There has been a wealth of work on the related traveling salesperson problem (TSP), both in networks and in geometric settings [9, 18, 27, 17]. TSP is NP-hard, even for points in the Euclidean plane. Until recently, the best approximation algorithm known for the Euclidean TSP was the Christofides heuristic (see, e.g., [18], which yields a 1.5-approximation. Polynomial-time approximation schemes (PTAS) were discovered recently by Arora [4, 5] and by Mitchell [21, 23]. The methods of [21, 23] employ "m-guillotine subdivisions" in the plane; this tool is also essential for some of our results, which are based on a strengthening of some of the analysis in [21, 23].

A related problem is the k-MST problem, in which we are given a graph on n vertices with nonnegative distances on the edges, and an integer $k \leq n$, and our goal is to find a tree of least total weight that spans some subset of k vertices. The problem is known to be NP-hard, both in general graphs and in the Euclidean plane [12, 29, 26]. The current best approximation algorithm for general edge-weighted graphs is a 3-approximation by Garg [14], which applies also to the "rooted" case (the tree is required to include a given node); this has been improved to a 2.5-approximation, by Arya and Ramesh [6], if the tree is not "rooted." The Euclidean k-MST now has a PTAS [4, 5, 21, 23], as does the k-TSP, which asks for a shortest cycle visiting some subset of k points. For the graph version of the k-TSP, with edge weights obeying triangle inequality, Garg's method yields a 3-approximation for both the rooted and unrooted version; the result of Arya and Ramesh [6] does not apply.

Another related problem is the *quota-driven salesperson* problem, in which each site has an associated integral value, w_i, and a salesperson has a given integer quota, R. The objective is to find a shortest possible cycle such that the sum of the values for the sites visited is at least R. By replicating each site (w_i times), a k-MST approximation algorithm gives also an approximation for this problem. In the *prize-collecting salesman* problem, as studied by Balas [8] (see also [10]), the setup is the same as in the quota-driven salesman problem, except that, in addition to "values" w_i, there are non-negative penalties associated with each site, and the objective function is now to minimize the sum of the distances traveled *plus* the sum of the penalties on the points *not* visited, subject to satisfying the quota R. As discussed in [7], an approximation algorithm follows from concatenating a cycle obtained for the quota-driven salesman, with the 2-approximation cycle given by the algorithm of Goemans and Williamson [15] (which considers the effect of penalties, but does not use the quota constraint).

The class of problems we address in this paper is based on the *orienteering problem* (or *bank robber* problem). The problem is essentially the dual of the quota-driven salesman problem: Given an upper bound on distance, B, maximize the total value of all points visited, for a salesperson that departs from a given root site. Heuristics were provided (without approximation guarantees) by [16, 28]. Awerbuch et al. [7] give an approximation algorithm for the *unrooted* cycle case, based on approximations to the k-TSP. These previous bounds are discussed in more detail later, as we compare them to our own results. For the original (rooted) version of the problem, it has been an intriguing open problem to obtain *any* nontrivial approximation algorithm.

Our work is also related to problems in bicriteria path optimization, in which we are to minimize one measure of "length", subject to an upper bound on another notion of "length"; see [2, 3, 11, 24], as well as the surveys by Mitchell [20, 22].

Preliminaries

We assume that the input to our problems is given by a set of n points (*sites*) in the Euclidean plane. For the path and cycle cases, we also consider a discrete metric space (an edge-weighted graph whose weights obey the triangle inequality), while for the tree case, we consider graphs with general (nonnegative) edge weights. We use the terms "length" and "weight" interchangeably. We let $d(p, q)$ denote the distance between p and q.

We let $n(T)$ and $w(T)$ denote the number of sites visited by network T and the total weight of T, respectively.

We let $B > 0$ be the given resource bound; B is an upper bound on the total length of the allowed network (tree, path, or cycle).

A *root* is a site that is required to be covered by the network.

Sites may have associated weights ("values"). Assuming these weights are integral, we can replicate point sites, and thereby reduce the problem to the unweighted case, in which our objective is to maximize the *number* of sites visited by the network. (This approach results in time bounds that are pseudopolynomial, in n and the sum of the weights.) Thus, we focus on the unweighted case in the remainder of this paper.

In this paper, all algorithms are *polynomial-time* approximation algorithms, whose worst-case complexity is polynomial in n. We omit the term "polynomial-time," as well as the actual time bounds, in our statements of results. A c-approximation algorithm for a maximization (resp., minimization) problem is one which guarantees that the solution obtained has an objective value that is at least OPT/c (resp., at most $c \cdot OPT$), where OPT is the optimal value of the objective function.

2 Rooted Orienteering Problems for Points in the Plane

We begin by considering the *rooted* case of the orienteering problems, in which one particular site, r, is given, and the tree/path/cycle is constrained to contain r.

The effect of having a root is considerable, as far as approximation algorithms are concerned. The difficulty stems from the fact that an optimal network may visit a huge number of sites that lie in a very small cluster at a distance from r that is nearly B, making it difficult to visit at least a constant fraction of these sites, unless the network is very efficient in its use of length.

In fact, no previous approximation algorithms, even with a logarithmic approximation factor, were known for the rooted orienteering problems. Here, by exploiting geometric structure, and tightening some analysis of m-guillotine subdivision approximations, we are able to obtain constant-factor approximations.

2.1 Rooted Tree-Orienteering Problem

We obtain a 3-approximation for points in the plane as follows.

(1) For each value of k, $k = 1, \ldots, n$, we find a shortest possible connected m-guillotine subdivision, rooted at r, spanning k or more points. We tabulate the lengths of the resulting graphs, for each k.

(2) For each value of k, $k = 1, \ldots, n$, and for each vertex v, we find a shortest possible connected m-guillotine subdivision, rooted at v, spanning k or more points; if the resulting graph does not contain r, then we add the vertex r and the edge rv to the graph. We tabulate the lengths of the resulting graphs, for each k.

(3) Among the graphs tabulated, the algorithm selects the graph with the maximum number of points and whose total length is less than the specified bound B.

In the above algorithm, it will suffice to set $m = 2$. Let T^* denote an optimal rooted tree, spanning the maximum number, $n(T^*)$, of points, while having weight at most B. Now, we claim that:

Lemma 1 *Among the graphs tabulated of types (1) and (2), there must be one whose weight is less than B, while the cardinality, k, of the point set spanned is at least $n(T^*)/3$.*

Proof. By standard results on separators in trees (e.g., see [25], Theorem 9.1, page 150), we know that there exists a partitioning of the optimal tree T^* into two subtrees, T_1 and T_2, sharing a common vertex v of T^*, so that $(2/3)n(T^*) \geq n(T_i) \geq (1/3)n(T^*)$. Without loss of generality, we assume that $r \in T_1$. We now consider two cases:

(i) $(w(T_1) - d(r, v) > w(T_2))$

Let T' be the union of $T_{2,G}$ and the segment rv, where $T_{2,G}$ is an m-guillotine subdivision containing T_2. By Theorem 3.3 of Mitchell [23], we know that there exists an m-guillotine subdivision, $T_{2,G}$ so that

$$w(T_{2,G}) \leq (1 + \epsilon)w(T_2),$$

where $\epsilon = \frac{\sqrt{2}}{m}$, which implies that

$$w(T') \quad \leq \quad (1 + \epsilon)w(T_2) + d(r, v)$$

$$
\begin{aligned}
&< \epsilon w(T_2) + w(T_1) \\
&\leq w(T_2) + w(T_1) \\
&\leq w(T^*) \leq B
\end{aligned}
$$

where we assume that $m \geq 2$, so that $\epsilon < 1$. Thus, T' will be among the trees of type (2) enumerated by the algorithm.

(ii) $(w(T_1) - d(r, v) \leq w(T_2))$

Let $T'' = T_{1,G}$ be an m-guillotine subdivision containing T_1, and let $\epsilon = \frac{\sqrt{2}}{m}$. Again, by Theorem 3.3 of Mitchell [23], we know that there exists an m-guillotine subdivision, $T_{1,G}$ so that

$$w(T_{1,G}) \leq (1 + \epsilon) w(T_1).$$

However, it turns out that the above inequality is not strong enough for our purposes here. Theorem 4 (see Section 2.4 below) is a strengthening of Theorem 3.3 of Mitchell [23]. Corollary 1 of Theorem 4 proves the following inequality:

$$w(T_{1,G}) \leq d(r, v) + (1 + \epsilon)(w(T_1) - d(r, v)).$$

With this result, we then obtain

$$
\begin{aligned}
w(T'') = w(T_{1,G}) &\leq d(r, v) + (1 + \epsilon)(w(T_1) - d(r, v)) \\
&\leq d(r, v) + 2(w(T_1) - d(r, v)) \\
&= w(T_1) + w(T_1) - d(r, v) \\
&\leq w(T_1) + w(T_2) = w(T^*) \leq B,
\end{aligned}
$$

where we have again assumed that $m \geq 2$, so that $\epsilon < 1$. Thus, T'' will be among the trees of type (1) enumerated by the algorithm.

Thus, the algorithm will enumerate both T' or T'' (among others), one of which must contain at least a third of the vertices in the optimal tree T^*. □

Theorem 1 *There is a 3-approximation polynomial-time algorithm for the rooted tree-orienteering problem on a set of points in the plane.*

2.2 Rooted Path-Orienteering Problem

The algorithm for the path-orienteering problem is very similar to the one for the tree-orienteering problem. The main difference is that instead of looking for connected m-guillotine subdivisions in the first two steps, the algorithm looks for connected m-guillotine subdivisions from which a path induced on the same set of vertices can be extracted. The proof of the approximation guarantee is also similar, except for one essential difference: there exists a partitioning of the optimal path T^* into two subpaths, sharing a common vertex, each having at least $n(T^*)/2$ vertices. This enables us to prove a performance guarantee of 2 for rooted path-orienteering. We omit details here.

Theorem 2 *There is a 2-approximation polynomial-time algorithm for the rooted path-orienteering problem on a set of points in the plane.*

2.3 Rooted Cycle-Orienteering Problem

We obtain a 2-approximation for points in the plane as follows. The algorithm computes for each value of k, $k = 1, \ldots, n$, a shortest possible connected "bridge-doubled" m-guillotine subdivision that includes the root vertex r and at least $k - 1$ other points, and where every vertex that is not on a bridge has degree two. It then extracts a tour from each such subdivision, using the property that bridge-doubling allows one to obtain a subset of the edges which is Eulerian (as outlined in Section 4.2 of Mitchell [23]). The algorithm tabulates the lengths of the resulting tours for each k, and among the tours with length bounded by B, the algorithm selects one having the maximum number of vertices.

In the algorithm, it will suffice to set $m \geq 3$. Let T^* denote an optimal tour visiting the maximum number, $n(T^*)$, points, while having weight at most B. Now, we claim that:

Lemma 2 *Among the tours tabulated by the algorithm described above, there must be one whose weight is less than B, while the cardinality, k, of the point set visited is at least $n(T^*)/2$.*

Proof. The length, $w(T^*)$, of an optimal tour, T^*, is bounded by B. Let v be a point on the tour (not necessarily a vertex) such that the points r and v partition the tour into two paths of weight $w(T^*)/2$. Let the two paths be called T_1 and T_2. Without loss of generality, assume that the path T_1 has at least $n(T^*)/2$ vertices on it.

Let T' be the union of T_1 and the segment rv. Let T'_G be a (bridge-doubled) m-guillotine subdivision containing T'. Once again, we need a stronger version of Theorem 3.3 from Mitchell [23]. Here, since T' contains a tour on its induced vertices, we apply Corollary 1 from Section 2.4 below, which provides the appropriate strengthening. Thus, we have:

$$\begin{aligned} w(T'_G) &\leq w(T') + \epsilon(w(T') - 2d(r,v)) \\ &= w(T_1) + d(r,v) + \epsilon(w(T_1) - d(r,v)) \\ &= (1+\epsilon)w(T_1) + (1-\epsilon)d(r,v) \\ &\leq (1+\epsilon)w(T_1) + (1-\epsilon)w(T_1) = 2w(T_1) = B \end{aligned}$$

Although v may not be a vertex, it is clear that there exists a m-guillotine subdivision T''_G that contains the same set of vertices as T'_G and with length no more than that of T'_G. Thus, T''_G will be among the tours enumerated by the algorithm. □

Theorem 3 *There is a 2-approximation polynomial-time algorithm for the rooted cycle-orienteering problem on a set of points in the plane.*

2.4 A Strengthened m-Guillotine Subdivision Theorem

We now supply a theorem on m-guillotine subdivisions, which strengthens the original analysis in [23]. The crucial new aspect to the theorem is that, when converting an arbitrary subdivision into an m-guillotine subdivision, there is some portion of the length of the original subdivision that does not need to be "charged." This allows us, in the corollary below, to bound the increase in length (between an original subdivision and its guillotine version) by a small fraction of the *difference* between the length of the original subdivision and the length of an appropriate subgraph that is known to exist in the subdivision.

First, we need some notation. For a set E of line segments, we say that a segment pq on an edge of E is *i-exposed above* if there are exactly i connected components in the intersection of E and the upward-pointing ray from any point on pq. (The ray includes its endpoint, so that the intersection with pq is counted among the i components.) See Figure 2.4 for an example of a portion of E that is 2-exposed above. We similarly define i-exposed *below*, i-exposed *right*, and i-exposed *left*. Let $\xi_{i,1}(E)$ ($\xi_{i,2}(E)$) be the sum of the x-projections of all subsegments of E that are i-exposed above (resp., below). Similarly, let $\xi_{i,3}(E)$ ($\xi_{i,4}(E)$) be the sum of the x-projections of all subsegments of E that are i-exposed right (resp., left). Further, let $\xi_j^{(m)}(E) = \sum_{i=1}^{m-1} \xi_{i,j}(E)$ be the total length of the respective projections for the portion of the edges E that are within the first m levels from the boundary of the bounding box of E. Finally, let $\xi^{(m)}(E) = \sum_{j=1}^{4} \xi_j^{(m)}(E)$ be the total of these numbers summed over the four sides of the bounding box. Note that $\xi^{(1)}(E)$ is simply the perimeter of the bounding box of the polygonal subdivision and that $\xi^{(m-1)}(E) \leq \xi^{(m)}(E)$.

Theorem 4 *Let S be a polygonal subdivision, with edge set E, of length L. Then, for any positive integer m, there exists an m-guillotine subdivision, S_G, of length at most $L + \frac{\sqrt{2}}{m}(L - \frac{\xi^{(m)}(E)}{2})$ whose edge set, E_G, contains E.*

Proof. (Sketch) We follow much of the proof of Mitchell [23], while making use of the fact that the portion of E that is within m levels of the top (resp., bottom, left, right) of the boundary box of E is never "charged" from below (resp., from the top, right, left). The total projected length of what is left uncharged is thus $\xi^{(m)}(E)$. □

The following corollary is required in proofs from Section 2.1, Section 2.2, and Section 2.3 above.

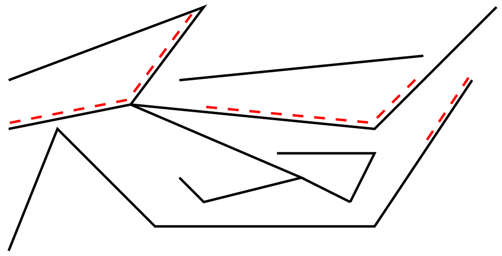

Figure 1: The portion of the edge set E that is 2-exposed above is highlighted.

Corollary 1 *Let S be a polygonal subdivision, with edge set E, of length L. If S connects two vertices r and v in the subdivision, then, for any positive integer m, there exists an m-guillotine subdivision, S_G, whose edge set contains E, and is of length at most $L+\frac{\sqrt{2}}{m}(L-d(r,v))$. Furthermore, if S contains a cycle involving the vertices r and v, then, for any positive integer $m \geq 2$, there exists an m-guillotine subdivision, S_G, whose edge set contains E, and is of length at most $L+\frac{\sqrt{2}}{m}(L-2d(r,v))$ (or, $L+\frac{2\sqrt{2}}{m}(L-2d(r,v))$, in the case of bridge-doubled m-guillotine subdivisions)*

Proof. (Sketch) By Theorem 4, we know that

$$w(S_G) \leq L + \frac{\sqrt{2}}{m}(L - \frac{\xi^{(m)}(E)}{2})$$

The two statements in the corollary are proved by observing the following: if S connects vertices r and v, then $\xi^{(m)}(E) \geq \xi^{(1)}(E) \geq 2d(r,v)$; if S contains a cycle involving the vertices r and v, then $\xi^{(m)}(E) \geq \xi^{(2)}(E) \geq 4d(r,v)$. □

3 Unrooted Orienteering Problems

The following observation was made in [7], for the cycle-orienteering problem. A similar proof holds also for the path and tree versions:

Theorem 5 *Given a c-approximation for the k-TSP we can obtain a $2c$-approximation for the unrooted orienteering problem (tree-, path-, or cycle-orienteering).*

Proof. We first prove the cycle-orienteering bound: For all $k = 1,2,\ldots,n$ find T_k an approximate k-TSP. Let k be the number of points such that $T_k \leq cB$ and $T_{k+1} > cB$. Take the cycle T_k and break it into $2c$ pieces of length at most $B/2$ each. Take the piece that contains the most points, double it to create a cycle of length at most B containing at least $k/(2c)$ points. Note that since $T_{k+1} > cB$ we must have that the optimal cycle of $k+1$ points $T^*_{k+1} > B$, therefore the optimal solution to the orienteering problem with bound B is k.

The proof for the path- and tree-orienteering problems is similar: For all $k = 1,2,\ldots,n$ find T_k an approximate k-TSP. Let k be the number of points such that $T_k \leq 2cB$ and $T_{k+1} > 2cB$. Take the cycle T_k and break it into $2c$ pieces of length at most B each. Take the piece that contains the most points, it is a path-therefore tree of length at most B containing at least $k/(2c)$ points. Note that since $T_{k+1} > 2cB$ we must have that the optimal cycle of $k+1$ points $T^*_{k+1} > 2B$, and hence the best tree on $k+1$ points is of length $> B$, and therefore the optimal solution to the tree-orienteering problem with bound B is k. □

Remark. We note that the method in the proof of the above theorem does not apply to the *rooted* cases.

Garg's [14] $c = 3$ bound for metric spaces for the k-TSP thus yields an approximation factor 6 for the unrooted tree-, path-, and cycle-orienteering problems in metric spaces. The PTAS of Mitchell [23] or Arora [4] imply $c = 1+\epsilon$, and therefore an approximation factor $2+\epsilon$ for the unrooted tree-, path-, and cycle-orienteering problems for points in the plane. Next, we improve these results.

3.1 Unrooted Tree-Orienteering

We begin with a result on the geometric version of the problem:

Theorem 6 *There is a 2-approximation polynomial-time algorithm for the unrooted tree-orienteering problem on a set of points in the plane.*

Proof. Our algorithm begins by computing a $(1+\epsilon)$-approximate k-MST, for each of the n possible values of k ($k = 1, 2, \ldots, n$); this can be done in polynomial time, using the algorithms of Mitchell [23] or Arora [4]. Then, among these n trees, we simply output the maximum cardinality (maximum k) tree whose weight is at most B.

We now argue that this algorithm yields a tree having at least $k^*/2$ vertices, where k^* is the optimal value of k (the maximum number of points that can be spanned with a tree of weight at most B).

For each value of $k = 1, 2, \ldots, n$, let B_k be the length of an *optimal* k-MST, and let A_k be the length of the approximately optimal tree produced by the algorithm. We know that $A_k \leq B_k(1+\epsilon)$, for each value of k. By Theorem 10, we know that $B_{k/2} \leq (2/3)B_k$. If we choose ϵ such that $\epsilon < 1/2$, then it is clear that $A_{k/2} \leq B_{k/2}(1+\epsilon) < (2/3)B_k(1+\epsilon) < B_k$, for all k. Thus, since k^* is the optimal value of k, meaning that $B_{k^*} \leq B < B_{k^*+1}$, we know that $A_{k^*/2} < B_{k^*} \leq B$. This implies that our algorithm will select a tree having a value of k that is *at least* as large as $k^*/2$. □

Next we provide a 5-approximation algorithm for graphs, improving the previous factor of 6.

Theorem 7 *There exists a 5-approximation algorithm for the (unrooted) tree-orienteering problem in an edge-weighted graph.*

Proof. The algorithm remains the same as in the proof of the previous theorem. However, no PTAS is known for this problem. There exists a 2.5-approximate algorithm for the k-MST problem [6]. For each value of $k = 1, 2, \ldots, n$, let B_k be the length of an *optimal* k-MST, and let A_k be the length of the approximately optimal tree produced by the algorithm. We know that $A_k \leq (5/2)B_k$, for each value of k. By Theorem 11 below, with $K = 5/2$, we know that $B_{k/5} \leq (2/5)B_k$. Hence, $A_{k/5} \leq (5/2)B_{k/5} < (2/5)(5/2)B_k = B_k$, for all k. Thus, since k^* is the optimal value of k, meaning that $B_{k^*} \leq B < B_{k^*+1}$, we know that $A_{k^*/5} < B_{k^*} \leq B$. This implies that our algorithm will select a tree having a value of k that is *at least* as large as $k^*/5$. □

3.2 Unrooted Path-Orienteering Problem

As in the next section, we cannot get improvements based on our combinatorial tools. Our improvement is made in the geometric case, where, using the results on the *rooted* version, we achieve a factor 2, improving the previous bound of $2+\epsilon$.

3.3 Unrooted Cycle-Orienteering Problem

By applying the approximations for the tree-orienteering problem, we obtain approximation algorithms also for the cycle-orienteering problem, based on a simple "doubling" of a minimum spanning tree:

Corollary 2 *If G is a complete graph with edge weights from a metric space, then there exists a cycle on at least $n/3$ vertices of G with total weight bounded by the weight of a minimum spanning tree of G.*

Note that the method of Christofides (see, e.g., [18]) does not help here, since there exist graphs for which the optimal TSP has length twice that of the minimum spanning tree.

Theorem 8 *There exists a $3c$-approximation algorithm for the orienteering problem, where c is the approximation factor for the weight-bounded tree cover problem. (Theorem 6 shows that $c = 2$ is attainable for points in the plane, while Theorem 7 shows that $c = 5$ is attainable for vertices in a general metric space.)*

Proof. The algorithm finds a tree with length bound of $B/2$ and then to doubles it to find a cycle of length at most B. We use the fact that a tree of length bounded by B can cover at least as many vertices as a cycle of length bounded by B. Thus a tree of length bound $B/2$ can cover at least as many as a third of those covered by a cycle of length B. □

The approximation factors obtained by this approach are 6 for points in the plane, and 15 for graphs with weights from a metric space, which are both inferior to previous results ($2+\epsilon$, and 6 resp.)!

However, for the case of points in the plane, we can obtain an improved result simply by applying the rooted result, for every choice of root:

Theorem 9 *There is a 2-approximation polynomial-time algorithm for the unrooted path- and cycle-orienteering problems on a set of points in the plane.*

3.4 Combinatorial Tools

Lemma 3 (Number-Partition Lemma) *If S is a set of n real numbers $x_1, x_2, \ldots, x_n$ such that their sum is B and $x_i < B/3, \forall i = 1, 2, \ldots, n$, then there exists a partition of S into two subsets, S_1 and S_2, such that $B/3 \leq \sum_{x_j \in S_i} x_j \leq 2B/3, i = 1, 2$.*

Lemma 4 (Tree-Partition Lemma) *Given an edge-weighted tree T, the following is true:*

1. *There exists a partition of the edges of T into an edge e and two subtrees T_1 and T_2 such that e is incident on both the subtrees, and $w(T_i) < w(T)/3, i = 1, 2$ (so $w(e) > w(T)/3$); OR*
2. *There exists a partition of the edges of T into two subtrees T_1 and T_2 such that $w(T)/3 \leq w(T_i) \leq 2w(T)/3, i = 1, 2$.*

Proof. Assume that the first case does not happen, i.e., there does not exist a large weight edge e such that the edges of T can be partitioned into edge e and two subtrees T_1 and T_2 with $w(T_i) < w(T)/3, i = 1, 2$. We will then show that there must exist a partition of the edges of T into two subtrees T_1 and T_2 such that $w(T)/3 \leq w(T_i) \leq 2w(T)/3, i = 1, 2$. We display an algorithm that will, in fact, find such a partition.

The algorithm is as follows. Start at any vertex v with incident edges $e_1, e_2, \ldots, e_t$. Consider a partition of T into subtrees $T_i, i = 1, 2, \ldots, t$, such that subtree T_i includes edge e_i and all the subtrees share only one vertex in common, namely vertex v.

Case 1: If there exists a subtree T_i such that $w(T)/3 \leq w(T_i) \leq 2w(T)/3$, then the lemma is proved by the following argument. Let T_1 be the subtree T_i. Let T_2 be the union of the other subtrees after identifying vertex v. The subtrees T_1 and T_2 satisfy the inequality $w(T)/3 \leq w(T_i) \leq 2w(T)/3, i = 1, 2$.

Case 2: If there exists no subtree T_i such that $w(T)/3 \leq w(T_i)$, then by Lemma 3, there exists a partition of the subtrees into two sets such that the sum of the weights of the subtrees in each of the sets lies between $w(T)/3$ and $2w(T)/3$.

Case 3: If there exists a subtree T_i such that $w(T_i) > 2w(T)/3$, and if the edge e_i is incident on vertices v and v', then we restart the algorithm from vertex v'.

It is clear that if Cases 1 or 2 occur at any time, the algorithm terminates after correctly finding the subtrees T_1 and T_2 needed for the proof of the lemma. We are left to show that Case 3 can occur only a finite number of times. As long as the algorithm does not return to a vertex v, it is bound to hit a leaf and terminate in at most n steps. If the algorithm ever returns to vertex v from vertex v', it will oscillate forever between vertices v and v'. We will now show that this cannot happen. For the sake of contradiction, assume that the algorithm oscillates between adjacent vertices v and v'. Let e be the edge connecting them. Let the union of all subtrees incident on v (v') that do not include edge e be called T_{i1} and T_{i2}. Define the subtrees $T_i = T_{i1} \cup \{e\}$ and $T_i' = T_{i2} \cup \{e\}$. Since the algorithm moves from vertex v to v', we know that $w(T_i) > 2w(T)/3$. Similarly, since the algorithm moves from vertex v' to v, we know that $w(T_i') > 2w(T)/3$. Since $w(T_{i1}) + w(T_i') = w(T) = w(T_{i2}) + w(T_i')$, we get that $w(T_{i1}) < w(T)/3$ and $w(T_{i2}) < w(T)/3$, contradicting our assumption that Case 1 does not occur. □

Theorem 10 *Given an edge-weighted tree T spanning n vertices, the following statements are true:*

1. *There exists a subtree T' of T with $w(T') \leq 2w(T)/3$ and spanning at least $n/2$ vertices, AND*
2. *There exists a subtree T' of T with $w(T') \leq w(T)/2$ and spanning at least $n/3$ vertices.*

Proof. By the tree-partition lemma (Lemma 4), the tree T has one of two possible decompositions. If the first case occurs, then T can be partitioned into two subtrees (T_1 and T_2) and an edge with both subtrees of weight at most $w(T)/3$. Since T_1 and T_2 also partition the vertex set of T, one of them must contain at least $n/2$ vertices. Without loss of generality we assume that T_1 contains at least $n/2$ vertices. Then T_1 satisfies both the statements of the theorem.

If, on the other hand, the first case does not occur, then T can be partitioned into two subtrees (T_1 and T_2) with both subtrees of weight between $w(T)/3$ and $2w(T)/3$. Since one of them must contain at least $n/2$ vertices, this subtree must satisfy the second statement of the theorem. We are left only with proving the first statement of the theorem for this case. Without loss of generality assume that $w(T_1) \geq w(T_2)$. Thus $w(T_2) \leq w(T)/2$. If $n(T_2) \geq n/3$ vertices, then T_2 satisfies the requirements of the theorem. However, if $n(T_2) < n/3$, then $n(T_1) > 2n/3$. Now we apply the partition lemma on tree T_1.

If the first case occurs, then T_1 can be partitioned into two subtrees (T_{11} and T_{12}) and an edge, with both subtrees of weight at most $w(T_1)/3$. Let $n(T_{11}) \geq n(T_{12})$. Thus, $n(T_{11}) \geq n(T_1)/2 \geq (1/2)(2n/3) = n/3$. Also, $w(T_{11}) < w(T_1)/3 < w(T)/3$. Thus, T_{11} satisfies the requirements of the theorem.

Finally, if the first case does not hold for T_1, then T_1 can be partitioned into two subtrees, T_{11} and T_{12}, each of weight at most $2w(T_1)/3$. Without loss of generality, assume that $n(T_{11}) \geq n(T_{12})$. Thus $n(T_{11}) \geq n(T_1)/2 \geq (1/2)(2n/3) = n/3$. Also, $w(T_{11}) \leq 2w(T_1)/3 \leq (2/3)(2w(T)/3) < w(T)/2$. Thus, T_{11} satisfies the requirements of the theorem. □

We generalize Theorem 10 as follows.

Theorem 11 *Given an edge-weighted tree T spanning n vertices, and any $K > 1$, there exists a subtree T' of T with $w(T') \leq w(T)/K$ and spanning at least $n/2K$ vertices.*

Proof. Start with the tree T spanning n vertices. Now apply the partition lemma on T. If the first case holds, then among the two subtrees T_1 and T_2 "pick" the subtree with more number of vertices. If the second case holds, then "pick" the subtree with the larger ratio of number of vertices to the total weight. If the subtree "picked" has weight at most $w(T)/k$, then return it. If the subtree "picked" has weight larger than $2w(T)/K$, then iterate the process with the "picked" subtree. Let the subtree at the end of the j-th iteration be denoted by $T^{(j)}$. Let the above process terminate after $i-1$ iterations and let $T^{(i-1)}$ be the resulting tree with weight at most $2w(T)/K$.

The idea here is that whenever the partition lemma is applied, it produces a subtree whose ratio of number of vertices to the total weight is no less than that at the start of the iteration. Let $T_1^{(j)}$, $T_2^{(j)}$ be the two subtrees of $T^{(j)}$, and assume without loss of generality that $n(T_1^{(j)})/w(T_1^{(j)}) \geq n(T_2^{(j)})/w(T_2^{(j)})$, then by cross multiplying we get $n(T_1^{(j)})/w(T_1^{(j)}) \geq (n(T_1^{(j)}) + n(T_2^{(j)}))/(w(T_1^{(j)}) + w(T_2^{(j)})) = (n(T^{(j)})+1)/w(T^{(j)})$. Therefore, since the ratio of the the number of vertices to the weight never decreases, $n(T^{(i-1)}) \geq n \cdot w(T^{(i-1)})/w(T) \geq n/K$.

The last step of the algorithm has two cases. If $w(T^{(i-1)}) \leq 3w(T)/(2K)$ then by the first part of Theorem 10, there exists a subtree T^i such that $w(T^i) \leq 2w(T^{(i-1)})/3 \leq w(T)/K$ with $n(T^i) \geq n(T^{(i-1)})/2 \geq n/2K$. On the other hand, if $3w(T)/(2K) < w(T^{(i-1)}) \leq 2w(T)/K$, then by the second part of Theorem 10, there exists a subtree T^i such that $w(T^i) \leq w(T^{(i-1)})/2 \leq w(T)/K$ with $n(T^i) \geq n(T^{(i-1)})/3 \geq n/2K$. □

We now show that the bounds above are tight for the metric case (for $K > 2$, in Theorem 11).

Lemma 5 *For $K > 2$, there exists a metric space on a set S of $Kt+1$ vertices such that any tree spanning at least $t+2$ of the vertices has total weight at least $2L/K$, where L is the weight of a minimum spanning tree of S. For $K = 2$, there exists a metric space on a set S of $2t$ vertices such that any tree spanning at least $t+1$ of the vertices has total weight L, where L is the weight of a minimum spanning tree connecting S.*

Proof. For $K = 2$, consider a configuration of n points that consists of two groups of vertices (each with t vertices) where any two vertices in the same group are at distance 0 from each other and the distance between a pair of vertices in two different groups is L, which is also the length of a minimum spanning tree on the given vertices. Clearly, any tree that spans $t+1$ vertices or more, must have length at least L.

For $K > 2$ we modify the construction slightly. Let s be a special vertex. Consider K groups of vertices (each with t vertices) where any two vertices in the same group are at distance 0 from each other and each of the vertices is at distance L/K from vertex s.. The distance between a pair of vertices in two different groups is defined by the metric space and is thus $2L/K$. Any tree that spans $t+2$ vertices or more, must span vertices from at least two different groups and consequently must have length at least $2L/K$. □

Lemma 6 *There exists a set S of $3t+1$ points in the plane such that any tree spanning at least $t+2$ of the vertices has total weight at least $2L/\sqrt{3}$, where L is the weight of a minimum spanning tree connecting S.*

Proof. Let s be a special vertex. Consider 3 groups of vertices (each with t vertices) with each group located at the 3 corners of an equilateral triangle with s at its center. The distance from a corner of the triangle to its center is $L/3$. Any two points in the same group is at distance 0 from each other. The distance between a pair of vertices in two different groups is $2L/\sqrt{3}$. □

The above lemma also shows that there exist configurations for which one needs a tree of weight at least $2L/\sqrt{3}$ in order to span at least $n/2$ points.

4 Generalizations and Extensions

4.1 Multiply-Rooted Orienteering

In the multiply-rooted orienteering problems, we are given, in addition to the length bound B, a set R of $\rho \geq 1$ required sites *roots* (they must be visited by the network) in addition to the n original sites, S. The goal is to maximize the number of non-root sites visited by a network, having length at most B, that is required to visit all ρ of the roots.

Since our results for rooted versions of the orienteering problem are strongly based on geometric methods, we again restrict ourselves to the Euclidean plane.

The multiply-rooted tree-orienteering problem is closely related to the (planar) Euclidean Steiner tree problem, which is known to be NP-hard [13]. In the Steiner tree problem, one is to determine a tree of minimum total length whose vertices are a *superset* of a given set of points. The hardness of the Steiner tree problem implies that it is NP-hard to decide if there exists a tree spanning all of the given set, R, of roots, whose length is at most B, since this amounts to asking if the length, L^*, of a minimum Steiner tree satisfies $L^* \leq B$.

One might ask whether the problem becomes "easier" if we are told that $B > L^*$. We show that even for such instances, if we can approximate the multiply-rooted tree-orienteering problem within a constant factor, then we can solve the Euclidean Steiner tree problem exactly, within the same time bound. Hardness also applies to approximating the path and cycle versions of the problem.

Theorem 12 *Unless P=NP, there is no polynomial-time algorithm achieving a constant-factor approximation of the number of additional (non-root) sites visited by a multiply-rooted tree (or cycle or path) of length at most B, where B is known to exceed the length of a Steiner tree on the set of roots.*

Proof. We give here the proof for the tree version of the problem; proofs for the cycle and path versions are similar, from Euclidean TSP. Given an instance of the Euclidean Steiner tree problem on a set P of points, we create an instance of the tree-orienteering problem in which the set of roots is $R = P$. For the Steiner tree problem, we must decide if $L^* \leq L$, where L^* is the length of the minimum Steiner tree on R. We add one additional non-root site v, at distance $B - L$ from a point $p \in R$ on the convex hull of R (so that the distance of v to R is minimized at p). Clearly, a tree of length at most B that visits v, as well as all roots, exists if and only if a Steiner tree of length at most L exists for the points R. Thus, if $L^* \leq L$, then a c-approximation algorithm for tree-orienteering must return a tree that includes v; otherwise, it may return a tree with zero additional (non-root) points. □

Remark. We note that the hardness proof applies also to the problem of approximating the maximum *total* number of sites visited (including the roots), by replacing v with many copies of itself.

We obtain a 3-approximation for the multiply-rooted tree-orienteering problem, provided the number, ρ, of roots is *constant*. (The running time is $O(n^{O(\rho)})$.) Additionally, we obtain 2-approximations for the path and cycle versions of the problem.

Theorem 13 *For any constant ρ, there is a polynomial-time 2-approximation (resp., 3-approximation) algorithm for the multiply-rooted path- and cycle-orienteering problems (resp., tree-orienteering problem) on a set of points in the plane.*

The proof of this theorem is based on some of the same ideas as in Theorems 1, 2, and 3, but with the application of a critical new theorem on approximations with m-guillotine subdivisions.

The algorithm does the following: For each choice of

(a). integer k, $k = 1, \ldots, n$,

(b). site v,

(c). partition of R into two sets, R_1 and R_2,

we find a shortest possible connected $m\rho$-guillotine subdivision, visiting site v, spanning k or more non-root sites, and spanning all of the roots R_1. We then append this tree to the minimum (Steiner) tree spanning $\{v\} \cup R_2$, and tabulate the lengths of all such trees. The tree whose length is at most B that maximizes the number of sites visited is a tree that we claim approximates the multiply-rooted tree-orienteering problem.

Lemma 7 *Among the graphs tabulated by the algorithm, there must be one whose weight is less than B, while the cardinality of the point set spanned is at least $n(T^*)/3$.*

Proof. As in the proof of Lemma 1, we know that there exists a partitioning of an optimal tree T^* into two subtrees, T_1 and T_2, sharing a common vertex v of T^*, so that $(2/3)n(T^*) \geq n(T_i) \geq (1/3)n(T^*)$. Let S_i denote the Steiner tree spanning the roots that are contained in subtree T_i.

Without loss of generality, we may assume that $w(T_1) - w(S_1) \leq w(T_2) - w(S_2)$. Now, consider the tree $T' = T_{1,G} \cup S_2$, where $T_{1,G}$ is an $m\rho$-guillotine subdivision containing the edge set of $T_{1,G}$. Then, T' spans all of R, and it is among the class of trees enumerated by the algorithm Thus, we have only to show that its weight is bounded by B:

$$\begin{aligned} w(T') &= w(T_{1,G}) + w(S_2) \\ &\leq w(T_1) + \epsilon(w(T_1) - w(S_1)) + w(S_2) \\ &\leq w(T_1) + \epsilon(w(T_2) - w(S_2)) + w(S_2) \\ &\leq w(T_1) + (w(T_2) - w(S_2)) + w(S_2) \\ &= w(T_1) + w(T_2) \leq B \end{aligned}$$

where we assume that $m\rho \geq 2$, so that $\epsilon < 1$, and we have used Theorem 14 and its corollary, below, to obtain the first inequality. $\square$

In order to state our new result on m-guillotine subdivisions, we need to generalize some of the notions from [23]. Our goal is to define the concept of an m-guillotine subdivision *with respect to* another subdivision, S', having edge set E'. We assume some familiarity with the terminology in [23].

First, we generalize the notion of an "m-span" to that of an m-span, $\sigma_{m,S'}(\ell)$, *with respect to another subdivision* S': $\sigma_{m,S'}(\ell)$ is defined as the union of the (usual) m-spans of ℓ, taken *between* each pair of crossing points between ℓ and E'. A horizontal or vertical line ℓ is an *m-perfect cut of S with respect to subdivision S', and the rectangular window W* if $\sigma_{m,S'}(\ell) \subseteq E$. We then say that subdivision S is *m-guillotine with respect to* another subdivision, S', if there exists an m-perfect cut *with respect to* S', such that the restriction of S to each side of the cut is (recursively) m-guillotine with respect to S'.

For a positive integer m, we say that a subdivision S, having edge set E, is *m-perfect* if there are at most m connected components in the intersection of E with any axis-parallel line.

Theorem 14 *Let S be a polygonal subdivision, with connected edge set E, of length L. Let S' be an m'-perfect polygonal subdivision, with edge set F, of length $L' \leq L$. Let m be any positive integer. Then, there exists a subdivision, S_G, that is mm'-guillotine with respect to S', with length at most $L + \frac{\sqrt{2}}{m}(L - L')$ whose edge set, E_G, contains E.*

Corollary 3 *Let S be a polygonal subdivision, with edge set E, of length L, If S spans the set, R, of ρ roots, then, for any positive integer m, there exists an $m\rho$-guillotine subdivision, S_G, whose edge set contains E, and is of length at most $L + \frac{\sqrt{2}}{m}(L - L^*(R))$, where $L^*(R)$ is the length of a Steiner minimum spanning tree of R.*

4.2 Polygonal Sites

In the case that the n sites are given to us as a collection of *regions* rather than points, we get a problem closely related to the "TSP with Neighborhoods" (TSPN) problem. We assume that the regions are given as a collection of n simple polygons, having a total of N vertices.

For the TSPN, Arkin and Hassin [1] have obtained $O(1)$-approximation algorithms for "well behaved" (possibly overlapping) regions (e.g., regions are disks or have roughly equal-length and parallel diameter segments), while Mata and Mitchell [19] have obtained an $O(\log n)$-approximation algorithm for n general (possibly overlapping) regions, based on "guillotine rectangular subdivisions".

We prove the following theorem, giving the first approximation results for the orienteering version of the problem:

Theorem 15 *There is an $O(\log n)$-approximation algorithm for the tree-orienteering (and path- and cycle-orienteering) problems on a set of polygonal sites in the plane. The running time is polynomial in n and N.*

Proof. (Sketch) We give here the proof only for the tree version of the problem. Let T^* denote an optimal rooted tree, spanning the maximum number, $n(T^*)$, points, while having weight at most B.

Our approximation algorithm is to tabulate, for each value of k, $k = 1, \ldots, n$, a shortest possible connected guillotine rectangular subdivision spanning k or more regions. (This can be done in time polynomial in n and N, using dynamic programming, as in [19].) We then select, from among these, the subdivision that visits the maximum number k of regions, subject to its weight being less than the bound B.

We now have to prove that, among the subdivisions tabulated, there must be one, T, whose weight is less than B, while the cardinality, k, of the set of regions spanned is at least $n(T^*)/O(\log n)$. Our proof is based on our earlier combinatorial theorem, Theorem 11, which guarantees that there exists a subtree, T', of T^*, with $w(T') \leq w(T^*)/C \log n$ and spanning at least $n(T^*)/2C \log n$ regions, for any constant C. Then, there must exist a guillotine rectangular subdivision, G, spanning the same set of regions as does T' (and thus, spanning at least $n(T^*)/2C \log n$ regions), such that the length of G is at most $w(T') \cdot C \log n \leq w(T^*) \leq B$. This proves that our algorithm will indeed discover a tree spanning at least $n(T^*)/O(\log n)$ regions. □

5 Conclusion

An outstanding open problem that remains is to obtain an approximation algorithm (or prove that none exists) for the rooted orienteering problems in graphs.

Also, it is interesting to note that, in the geometric case, our current best bound (2) for the rooted cycle- and path-orienteering problems is better than the bound (3) for the rooted tree-orienteering problem. Can the tree version be improved? (In contrast, in the unrooted graph version, our methods give a factor of 5 for trees and 6 for cycles and paths.) Is the factor 2 best possible for the cycle and path versions?

References

[1] E. M. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Appl. Math.*, 55:197–218, 1994.

[2] E. M. Arkin, S. Khuller, and J. S. B. Mitchell. Geometric knapsack problems. *Algorithmica*, 10:399–427, 1993.

[3] E. M. Arkin, J. S. B. Mitchell, and C. D. Piatko. Bicriteria shortest path problems in the plane. In *Proc. 3rd Canad. Conf. Comput. Geom.*, pages 153–156, 1991.

[4] S. Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In *Proc. 37th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS 96)*, pages 2–12, 1996.

[5] S. Arora. Nearly linear time approximation schemes for Euclidean TSP and other geometric problems. In *Proc. 38th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS 97)*, 1997.

[6] S. Arya and H. Ramesh. A 2.5 factor approximation algorithm for the k-MST problem. Manuscript, Hong Kong University of Science and Technology, 1997.

[7] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala. Improved approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. In *Proc. 27th Annu. ACM Sympos. Theory Comput. (STOC 95)*, pages 277–283, 1995.

[8] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19:621–636, 1989.

[9] J. L. Bentley. Fast algorithms for geometric traveling salesman problems. *ORSA J. Comput.*, 4(4):387–411, 1992.

[10] D. Bienstock, M. X. Goemans, D. Simchi-Levi, and D. Williamson. A note on the prize collecting traveling salesman problem. *Math. Prog.*, 59:413–420, 1993.

[11] D. Eppstein, M. Overmars, G. Rote, and G. Woeginger. Finding minimum area k-gons. *Discrete Comput. Geom.*, 7:45–58, 1992.

[12] M. Fischetti, H. W. Hamacher, K. Jørnsten, and F. Maffioli. Weighted k-cardinality trees: Complexity and polyhedral structure. *Networks*, 24:11–21, 1994.

[13] M. R. Garey, R. L. Graham, and D. S. Johnson. The complexity of computing Steiner minimal trees. *SIAM J. Appl. Math.*, 32:835–859, 1977.

[14] N. Garg. A 3-approximation for the minimum tree spanning k vertices. In *37th Annual Symposium on Foundations of Computer Science*, pages 302–309, Burlington, Vermont, October 14-16 1996.

[15] M. Goemans and D. Williamson. General approximation technique for constrained forest problems. In *Proc. 3rd ACM-SIAM Sympos. Discrete Algorithms (SODA '92)*, pages 307–315, 1992.

[16] B. L. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Res. Logistics*, 34:307–318, 1991.

[17] M. Jünger, G. Reinelt, and G. Rinaldi. The traveling salesman problem. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Network Models*, Handbook of Operations Research/Management Science, pages 225–330. Elsevier Science, Amsterdam, 1995.

[18] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors. *The Traveling Salesman Problem*. Wiley, New York, NY, 1985.

[19] C. Mata and J. S. Mitchell. Approximation algorithms for geometric tour and network design problems. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 360–369, 1995.

[20] J. Mitchell. Shortest paths and networks. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 24, pages 445–466. CRC Press LLC, 1997.

[21] J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple new method for the geometric k-MST problem. In *Proc. 7th ACM-SIAM Sympos. Discrete Algorithms*, pages 402–408, 1996.

[22] J. S. B. Mitchell. Geometric shortest paths and network optimization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*. Elsevier Science, Amsterdam, 1998, to appear.

[23] J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM J. Comput.*, to appear.

[24] J. S. B. Mitchell, C. Piatko, and E. M. Arkin. Computing a shortest k-link path in a polygon. In *Proc. 33rd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 573–582, 1992.

[25] T. Nishizeki and N. Chiba. Planar graphs: Theory and algorithms. *Ann. Discrete Math.*, 32, 1988.

[26] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi. Spanning trees short and small. In *Proc. 5th ACM-SIAM Sympos. Discrete Algorithms*, pages 546–555, 1994.

[27] G. Reinelt. Fast heuristics for large geometric traveling salesman problems. *ORSA J. Comput.*, 4:206–217, 1992.

[28] T. Tsiligirides. Heuristic methods applied to orienteering. *J. of the Operational Research Society*, 35(9):797–809, 1984.

[29] A. Zelikovsky and D. Lozevanu. Minimal and bounded trees. In *Tezele Cong. XVIII Acad. Romano-Americane, Kishinev*, pages 25–26, 1993.