The Euclidean Orienteering Problem Revisited*

Ke Chen† Sariel Har-Peled‡

July 12, 2007

Abstract

We consider the *rooted orienteering problem*: Given a set P of n points in the plane, a starting point $r \in P$, and a length constraint $\mathcal{B}$, one needs to find a path starting from r that visits as many points of P as possible and of length not exceeding $\mathcal{B}$. We present a $(1-\varepsilon)$-approximation algorithm for this problem that runs in $n^{O(1/\varepsilon)}$ time; the computed path visits at least $(1-\varepsilon)k_{\text{opt}}$ points of P, where k_{opt} is the number of points visited by an optimal solution. This is the first polynomial time approximation scheme (PTAS) for this problem. The algorithm also works in higher dimensions.

1 Introduction

Consider a traveling salesperson who has a fixed amount of gasoline and wants to maximize the number of customers visited under this constraint. This is an instance of the *orienteering problem* that requires us to design a network that visits a maximum number of points, subject to an upper bound on the total length of the network. This problem is "dual" to the classical *k-TSP problem* [Aro98, Mit99, Gar05], which asks for a minimum length path visiting at least k points.

In this paper, we consider the *rooted path orienteering problem* in the plane, specified by P, r, and $\mathcal{B}$, where P is a set of n points in the plane, r is the starting point, and $\mathcal{B} > 0$ is the maximum length allowed. The solution to this problem is a path that starts at r and visits as many points as possible of P, such that the path length does not exceed $\mathcal{B}$. The effect of fixing the starting point is significant as far as approximation algorithms are concerned. Indeed, approximation algorithms for k-TSP extend easily to the *unrooted orienteering problem*, where there is no fixed starting point, while the approximation algorithm for the rooted orienteering problem is more challenging. The difficulty stems from the fact that an optimal path may visit a large number of points that lie in a small cluster at a distance nearly $\mathcal{B}$

*A preliminary version of this paper appeared in *Proc. 22nd Annu. ACM Sympos. Comput. Geom.*, pages 247–254, 2006.

†Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL 61801, USA; kechen@uiuc.edu; http://www.uiuc.edu/~kechen/.

‡Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL, 61801, USA; sariel@uiuc.edu; http://www.uiuc.edu/~sariel/. Work on this paper was partially supported by a NSF CAREER award CCR-0132901.

from r, thus making it difficult to visit at least a large fraction of these points unless the path is very efficient [AMN98].

Some related problems include the *prize-collecting traveling salesman problem* and the *vehicle routing problem*. They arise from real world applications such as delivering goods to locations or assigning technicians to maintenance service jobs. A substantial amount of work on heuristics for these problems can be found in the operations research literature [TV02].

Arkin *et al.* [AMN98] were the first to design approximation algorithms for the rooted orienteering problem. They considered the rooted orienteering problem for points in the plane when the underlying network is a path, a cycle, or a tree. Their algorithms provide a 2-approximation for the rooted path orienteering problem, and a 2 (resp. 3) approximation when the networks considered are cycles (resp. trees). Blum *et al.* [BCK$^+$03] proposed the first constant-factor approximation algorithm for the rooted path orienteering problem when the points lie in a general metric space. Bansal *et al.* [BBCM04] improved the approximation factor to 3. Arkin *et al.* [AMN98] asked whether a better approximation is possible in Euclidean spaces.

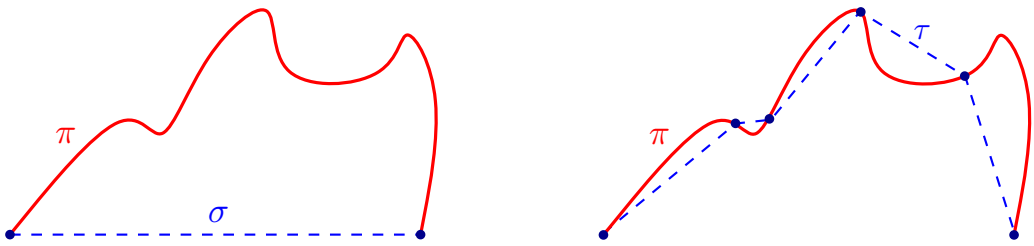
One difficulty in assailing this problem is the relative lack of algorithmic tools to handle rigid budget constraints. Since the development of $(1+\varepsilon)$-approximation algorithms for TSP [Aro98, Mit99], a large class of the problems that aim to minimize the tour length, subject to certain constraints on the points visited by the tour have been resolved. See the surveys by Mitchell [Mit00] and Arora [Aro03] for further information. In contrast, the behavior of the optimization problems that seek to maximize some function on the points visited, subject to constraints on the length of the path used or the timespans the points are being visited [BES05, BBCM04, CP05] are not as well understood.

The main idea in the previous approximation algorithms for the orienteering problem [AMN98, BCK$^+$03, BBCM04] was to transform an approximation algorithm for the *rooted k-TSP problem* into an approximation algorithm for the orienteering problem. In particular, Blum *et al.* [BCK$^+$03] formulated the notion of the *excess* for a path (which is defined to be the difference between the length of a path and the distance between the endpoints of the path), and then combined dynamic programming with the use of k-TSP to obtain an algorithm for orienteering in a metric space. These techniques were also implicitly used in the work of Arkin *et al.* [AMN98].

Our results. To obtain a PTAS for the orienteering problem, we extend the concept of the excess of a path into the $\mathfrak{u}$-*excess* of a path, which is (loosely) the difference in lengths between π and the best approximation to π by a polygonal line having $\mathfrak{u}$ vertices, see Figure 1. (Therefore, the previous notion of excess is 2-excess in our notation.)

To this end, we revisit the rooted k-TSP problem in the plane, and show that Mitchell's algorithm [Mit99] computes an $(\varepsilon, \mathfrak{u})$-approximation for rooted k-TSP; that is, the algorithm outputs a rooted path of length $\leq \|\pi\| + \varepsilon \cdot \mathcal{E}_{\pi,\mathfrak{u}}$, where π is any path that starts from the root and visits k points, $\|\pi\|$ denotes the length of π, and $\mathcal{E}_{\pi,\mathfrak{u}}$ is the $\mathfrak{u}$-excess of π. Note that the quantity $\mathcal{E}_{\pi,\mathfrak{u}}$ might be smaller than $\|\pi\|$ by several orders of magnitude. Therefore, we show that Mitchell's algorithm provides a much tighter approximation for k-TSP than what was previously known. See Section 3.

Armed with the new approximation algorithm for k-TSP, it is now possible to reduce



(i) (ii)

Figure 1: (i) The segment σ is the 2-skeleton of the path π. The 2-excess of π, namely $\mathcal{E}_{\pi,2}$, is the difference between the length of π and the length of σ. (ii) The polygonal line τ forms a 6-skeleton of π. The 6-excess of π, namely $\mathcal{E}_{\pi,6}$, is at most $\|\pi\| - \|\tau\|$.

the orienteering problem to an instance of k-TSP. The PTAS for orienteering is presented in Section 5.

The rest of the paper is organized as follows. Section 2 defines the problem. Section 3 presents the new analysis of Mitchell's algorithm. Section 4 extends the new k-TSP algorithm into higher dimensions by combining Mitchell's technique and Arora's k-TSP algorithm. Section 5 gives the PTAS for the orienteering problem. We conclude in Section 6.

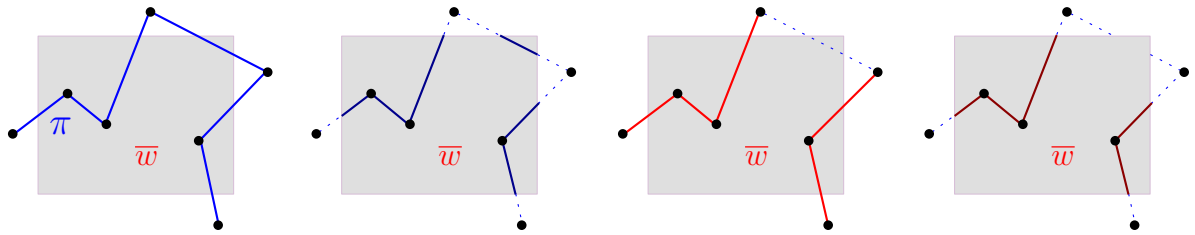
2 Problem statement and definitions

Let $\pi = \langle p_1, p_2, \ldots, p_k \rangle$ be a path that visits k points of P, starting at p_1 and ending at p_k. The *length* of π is denoted by $\|\pi\| = \sum_{i=1}^{k-1} \|p_{i+1} - p_i\|$. More generally, for a collection S of segments, $\|S\|$ denotes the total length of segments in S. Let $1 = i_1 < \ldots < i_{\mathfrak{u}} = k$ be a sequence of $\mathfrak{u} \leq k$ integers. The path $\langle p_{i_1}, p_{i_2}, \ldots, p_{i_{\mathfrak{u}}} \rangle$ is a $\mathfrak{u}$-*skeleton* of π. The *optimal* $\mathfrak{u}$-*skeleton* of π is the $\mathfrak{u}$-skeleton of π with maximum total length, denoted by $\mathcal{S}^{\mathfrak{u}}_{\text{opt}}(\pi)$. See Figure 1.

The $\mathfrak{u}$-*excess* of a path π is the difference between the length of π and its optimal $\mathfrak{u}$-skeleton, that is, $\mathcal{E}_{\pi,\mathfrak{u}} = \|\pi\| - \left\|\mathcal{S}^{\mathfrak{u}}_{\text{opt}}(\pi)\right\|$. Note that the $\mathfrak{u}$-excess of π may be considerably smaller than the length of π.

Given a set P of n points and a starting point $r \in P$, the *rooted k-TSP problem* is to find a shortest path that visits k points of P starting at r. An $(\varepsilon, \mathfrak{u})$-*approximation* to the rooted k-TSP is a path ϕ that visit k points of P starting at r, such that the length of ϕ is no more than $\|\mathcal{T}\| + \varepsilon \cdot \mathcal{E}_{\mathcal{T},\mathfrak{u}}$, for any path $\mathcal{T}$ that visits k points of P starting at r.

Definition 2.1 (The rooted orienteering problem.) Given a set P of n points, a budget $\mathcal{B}$, and a starting point $r \in P$, the *rooted orienteering problem* is to find a path ω_{opt} that visits as many points of P as possible, under the constraint that the length of ω_{opt} is at most $\mathcal{B}$. Let k_{opt} denote the number of points visited by ω_{opt}. A $(1-\varepsilon)$-*approximation* to the rooted orienteering problem is a path ω (starting at r) that visits at least $(1-\varepsilon)k_{\text{opt}}$ points of P, such that the length of ω is at most $\mathcal{B}$.



(i) π (ii) $\pi \cap \overline{w}$ (iii) $\pi(\overline{w})$ (iv) $\pi(\overline{w}) \cap \overline{w}$

Figure 2: The different ways of clipping a path π to a window.

3 An $(\varepsilon, \mathfrak{u})$-approximation algorithm for k-TSP

In this section, we present an $(\varepsilon, \mathfrak{u})$-approximation algorithm for k-TSP. The algorithm is the k-TSP algorithm of Mitchell [Mit99], and our contribution is the new tighter analysis of its performance (see Section 3.3). In the following, we first review Mitchell's algorithm and then present our new improved analysis.

3.1 Preliminaries

In the following, m is a fixed constant. We assume, without loss of generality, that the points of P all have distinct x and y coordinates, and P is contained in an axis-parallel square $\mathcal{Q}$. Let π be a given path visiting k points of P.

Definition 3.1 A closed, axis-parallel rectangle $\overline{w}$ is a *window* if $\overline{w} \subseteq \mathcal{Q}$. The *extent* of a window $\overline{w}$ is the larger of the width and height of $\overline{w}$, and is denoted by $\Delta_{\overline{w}}$. Let $\pi(\overline{w})$ denote the subset of π consisting of the union of segments of π having at least one endpoint inside (or on the boundary of) $\overline{w}$. Given a collection S of segments, we slightly abuse notations and denote the set of segments of S clipped to $\overline{w}$ by $S \cap \overline{w}$. See Figure 2.

A line ℓ is a *cut* for π, with respect to $\overline{w}$, if ℓ is a horizontal or vertical line and ℓ intersects $\overline{w}$; ℓ is an *m-perfect cut* for π, with respect to $\overline{w}$, if ℓ intersects the segments of $\pi(\overline{w}) \cap \overline{w}$ at most m times.

Definition 3.2 The combinatorial type of a window $\overline{w}$ with respect to π is the subset of P inside (or on the boundary of) $\overline{w}$ and a listing, for each of the four sides of $\overline{w}$, of the identities of the line segments of $\pi(\overline{w})$ that intersect it. (In particular, if a segment of π intersects $\overline{w}$ but both its endpoints are outside $\overline{w}$, then the segment is not considered in the combinatorial type of $\overline{w}$.) We say that $\overline{w}$ is a *minimal window* if there is no window $\overline{w}'$ that is strictly contained in $\overline{w}$ with the same combinatorial type as $\overline{w}$.

For a minimal window $\overline{w}$, if there is no m-perfect cut for π, with respect to $\overline{w}$, then it is *m-dense*. Namely, any horizontal or vertical line that intersects $\overline{w}$ has more than m intersection points with $\pi(\overline{w}) \cap \overline{w}$.

Given a window $\overline{w}$, Mitchell [Mit99] described how to "shrink" $\overline{w}$ into a minimal window by "pinning" all four sides of $\overline{w}$. It is not hard to see that the number of all possible minimal windows is $O(n^4)$, since (intuitively) it has four degrees of freedom.

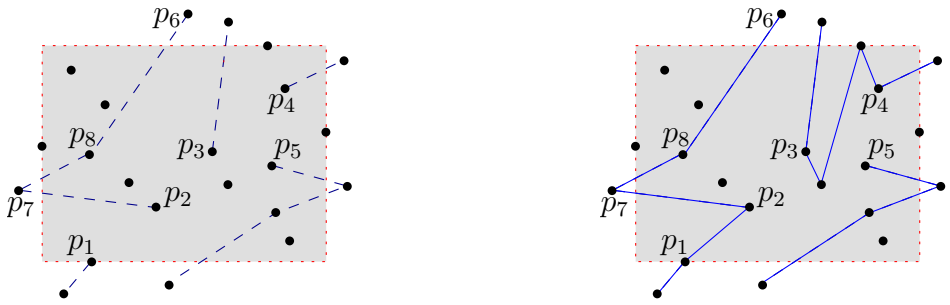


Figure 3: (i) An instance of the **WindowTSP** problem. The segments specify how the solution crosses the boundary of $\overline{w}$. The connectivity constraints are as follows: p_1 is required to connect to p_2 (possibly via other points within $\overline{w}$), p_3 is required to connect to p_4 (possibly via other points within $\overline{w}$), and p_5 is the starting point r of the path (namely, the degree of p_5 is 1). The multi-paths are required to visit 9 points in the window. (ii) A possible solution.

Claim 3.3 *If a window $\overline{w}$ is m-dense then $\|\pi(\overline{w}) \cap \overline{w}\| \geq m \cdot \Delta_{\overline{w}}$.*

Proof: Assume, without loss of the generality, that the width of $\overline{w}$ is greater than the height of $\overline{w}$, and the interval $[x_1, x_2]$ is the projection of $\overline{w}$ onto the x-axis. Let $f(\alpha)$ denote the number of segments of $\pi(\overline{w})$ within $\overline{w}$ that intersects the vertical line $x = \alpha$. By the density of $\overline{w}$, $f(x) > m$ for $x \in [x_1, x_2]$. The total length of the segments of $\pi(\overline{w}) \cap \overline{w}$ is lower bounded by the integral of $f(x)$ over $[x_1, x_2]$, which in turn is lower bounded by $m(x_2 - x_1) = m \cdot \Delta_{\overline{w}}$. ■

3.2 Review of the k-TSP algorithm

The "new" $(\varepsilon, \mathfrak{u})$-approximation algorithm for k-TSP is the algorithm of Mitchell [Mit99] for k-TSP, and we review it here only for the sake of completeness. We remind that the reader that m is a fixed (constant) number.

A problem instance of **WindowTSP** consists of: (i) a (minimal) window $\overline{w}$ that contains at least one point of P, with its boundaries determined by (up to) four points of P, (ii) an integer $h \geq 0$, indicating how many points interior to $\overline{w}$ should be visited, (iii) boundary information specifying at most m crossing segments (each determined by a pair of points of P, one interior to or on the boundary of $\overline{w}$, another outside $\overline{w}$) for each side of the boundary of $\overline{w}$, and (iv) connectivity constraints, indicating which pairs of crossing segments are required to be connected within $\overline{w}$. The solution to the **WindowTSP** problem is a set of (hopefully short) paths inside window $\overline{w}$ such that: (i) all of the boundary constraints are satisfied, (ii) all of the connectivity constraints within $\overline{w}$ are satisfied, and (iii) h points of P are visited by the paths inside $\overline{w}$. See Figure 3.

Clearly, the rooted k-TSP problem can be formulated as a **WindowTSP** instance consisting of a bounding box $\mathcal{Q}$ of P, a parameter k, empty boundary information, and connectivity constraints requiring that k points of P inside $\mathcal{Q}$ must be connected by a single path, with r as an endpoint of the path.

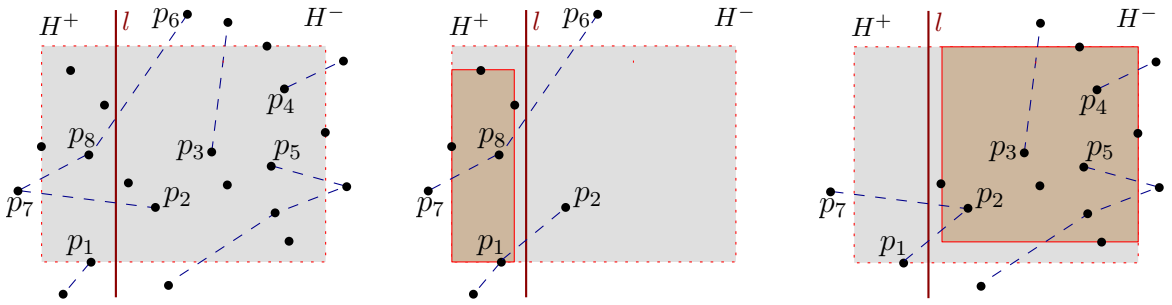


Figure 4: Performing a cut on the **WindowTSP** instance of Figure 3. (i) A cut l divides the window $\overline{w}$ into smaller windows $\overline{w} \cap H^+$ and $\overline{w} \cap H^-$. (ii) The minimal window for $\overline{w} \cap H^+$. The segments represent the crossing boundary information for the new window. In particular, the segment p_1p_2 is introduced when "guessing" the boundary information along the cut l before the recursive call. The other segments intersecting the boundary are inherited from the original instance. (iii) The minimal window for $\overline{w} \cap H^-$.

The recursive algorithm for **WindowTSP** works as follows. If the window $\overline{w}$ has at most m points of P in its interior, then the problem is solved by enumeration of all possible solutions. Otherwise, the algorithm tries all possible cuts for the current window recursively, enumerating over all the possible choices of valid boundary information along this cut and computing the cheapest option. If there is no m-perfect cut then the algorithm performs a cut and reduces the intersection by introducing bridges; see Remark 3.4 below. For our analysis, we only care whether a cut used by the algorithm is an m-perfect cut, or is it a more complicated cut.

When a cut divides a window into two smaller windows, we need to "shrink" those two windows into minimal windows. In particular, segments that just pass through a window are ignored during the shrinking. This is a small but important technicality. See Figure 4.

Remark 3.4 The algorithm of Mitchell [Mit99] also introduces bridges (close to, or) on the boundary of the window $\overline{w}$, where a bridge is a vertical or horizontal segment. To simplify our exposition, we ignored those bridges in describing the algorithm. Of course, for a correct working implementation those bridges are necessary. See [Mit99] for full details. See also Remark 3.10 below.

3.3 Analysis of the algorithm

The key observation in our analysis is that the approximation algorithm does not introduce any error when a cut is m-perfect. Thus, the error is introduced only when the algorithm works inside an m-dense window, but such windows have high "excess".

Definition 3.5 For a set S of segments, let $I_x(S)$ denote the projection of S to x-axis; namely, $I_x(S)$ is the set of all points α (on the x-axis), such that the vertical line $x = \alpha$ intersects the segments of S. Let $\mathrm{len}_x(S)$ denote the total length of $I_x(S)$. Note that $I_x(S)$ is a set of (disjoint) intervals on the real line, and $\mathrm{len}_x(S)$ is the total length of these intervals. We define $I_y(S)$ and $\mathrm{len}_y(S)$ in a similar fashion.

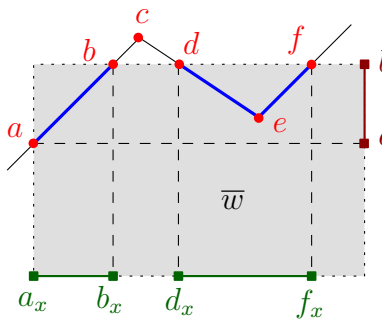


Figure 5: Illustrating the intersection of a polygonal line $\pi = \langle a, b, c, d, e, f \rangle$ with a window $\overline{w}$. The set $I_x(\pi \cap \overline{w})$ consists of the segments $a_x b_x$ and $d_x f_x$; the set $I_y(\pi \cap \overline{w})$ consists of segment $a_y b_y$. The surplus of π in $\overline{w}$ is $\|a-b\| + \|d-e\| + \|e-f\| - \sqrt{(\|a_x - b_x\| + \|d_x - f_x\|)^2 + \|a_y - b_y\|^2}$.

Definition 3.6 For a window $\overline{w}$ and a path π, the *surplus* of π in $\overline{w}$ is

$$\rho(\overline{w}, \pi) = \|\pi \cap \overline{w}\| - \sqrt{(\mathrm{len}_x(\pi \cap \overline{w}))^2 + (\mathrm{len}_y(\pi \cap \overline{w}))^2}.$$

It is easy to verify that the surplus is always non-negative. See Figure 5.

Lemma 3.7 *If $X, Y, X_1, \ldots, X_n, Y_1, \ldots, Y_n$ are non-negative real numbers such that*

$$\sum_{i=1}^{n} X_i \geq X \quad \text{and} \quad \sum_{i=1}^{n} Y_i \geq Y,$$

then $\sum_{i=1}^{n} \sqrt{X_i^2 + Y_i^2} \geq \sqrt{X^2 + Y^2}$.

Proof: This is an easy application of the triangle inequality. Indeed, let q_i be the point $(\sum_{j=1}^{i} X_j, \sum_{j=1}^{i} Y_j)$ in the plane, for $1 \leq i \leq n$; and let $q_0 = (0,0)$. Consider the path $\pi = \langle q_0, q_1, \ldots, q_n \rangle$. Clearly, we have that

$$\begin{aligned} \|\pi\| &= \sum_{i=1}^{n} \sqrt{X_i^2 + Y_i^2} = \sum_{i=1}^{n} \|q_i - q_{i-1}\| \geq \|q_n - q_0\| \\ &= \sqrt{\left(\sum_{i=1}^{n} X_i\right)^2 + \left(\sum_{i=1}^{n} Y_i\right)^2} \geq \sqrt{X^2 + Y^2}, \end{aligned}$$

as required. ∎

Lemma 3.8 *Let D be a set of interior disjoint windows (inside $\mathcal{Q}$), and let π be a polygonal path inside $\mathcal{Q}$. We have that $\mathcal{E}_{\pi,2} \geq \sum_{\overline{w} \in \mathsf{D}} \left(\|\pi \cap \overline{w}\| - \sqrt{2}\Delta_{\overline{w}} \right)$.*

Proof: Let Ψ be a decomposition of $\mathcal{Q}$ into interior disjoint axis-parallel rectangles such that Ψ contains all of the rectangles of D. Let $X = \mathrm{len}_x(\pi \cap \mathcal{Q})$ and $Y = \mathrm{len}_y(\pi \cap \mathcal{Q})$. For

a window $\overline{w}$, let $X_{\overline{w}} = \mathrm{len}_x(\pi \cap \overline{w})$ and $Y_{\overline{w}} = \mathrm{len}_y(\pi \cap \overline{w})$. Clearly, $X \leq \sum_{\overline{w}\in\Psi} X_{\overline{w}}$ and $Y \leq \sum_{\overline{w}\in\Psi} Y_{\overline{w}}$, since $I_x(\pi) = \bigcup_{\overline{w}\in\Psi} I_x(\pi \cap \overline{w})$ and $I_y(\pi) = \bigcup_{\overline{w}\in\Psi} I_y(\pi \cap \overline{w})$. Let s and t be the two endpoints of π. We have that

$$\mathcal{E}_{\pi,2} = \|\pi\| - \|s-t\| = \|\pi\| - \sqrt{\mathrm{len}_x(st)^2 + \mathrm{len}_y(st)^2} \geq \|\pi\| - \sqrt{X^2+Y^2},$$

since $\mathrm{len}_x(st) \leq X$ and $\mathrm{len}_y(st) \leq Y$. On the other hand, by Lemma 3.7, we get $\sqrt{X^2+Y^2} \leq \sum_{\overline{w}\in\Psi} \sqrt{X_{\overline{w}}^2 + Y_{\overline{w}}^2}$, since $\sum_{\overline{w}\in\Psi} X_{\overline{w}} \geq X$ and $\sum_{\overline{w}\in\Psi} Y_{\overline{w}} \geq Y$. Therefore,

$$\begin{aligned}\mathcal{E}_{\pi,2} &\geq \|\pi\| - \sqrt{X^2+Y^2} \geq \|\pi\| - \sum_{\overline{w}\in\Psi} \sqrt{X_{\overline{w}}^2 + Y_{\overline{w}}^2} = \sum_{\overline{w}\in\Psi} \Big(\|\pi\cap\overline{w}\| - \sqrt{X_{\overline{w}}^2 + Y_{\overline{w}}^2} \Big) \\ &= \sum_{\overline{w}\in\Psi} \rho(\overline{w},\pi) = \sum_{\overline{w}\in\overline{\mathsf{D}}} \rho(\overline{w},\pi) + \sum_{\overline{w}\in\Psi\setminus\overline{\mathsf{D}}} \rho(\overline{w},\pi) \geq \sum_{\overline{w}\in\overline{\mathsf{D}}} \rho(\overline{w},\pi),\end{aligned}$$

because the surplus $\rho(\overline{w},\pi)$ is always non-negative. Now, since

$$\rho(\overline{w},\pi) = \|\pi\cap\overline{w}\| - \sqrt{X_{\overline{w}}^2 + Y_{\overline{w}}^2} \geq \|\pi\cap\overline{w}\| - \sqrt{\Delta_{\overline{w}}^2 + \Delta_{\overline{w}}^2} = \|\pi\cap\overline{w}\| - \sqrt{2}\Delta_{\overline{w}},$$

we obtain $\mathcal{E}_{\pi,2} \geq \sum_{\overline{w}\in\overline{\mathsf{D}}} \rho(\overline{w},\pi) \geq \sum_{\overline{w}\in\overline{\mathsf{D}}} \big(\|\pi\cap\overline{w}\| - \sqrt{2}\Delta_{\overline{w}} \big)$, as claimed. ∎

Theorem 3.9 *Let $\pi = \langle p_1, p_2, \ldots, p_k \rangle$ be an arbitrary path that visits k points of P, and let $\mathfrak{u} \geq 2$ be an arbitrary fixed integer. One can compute, in $n^{O(\mathfrak{u})}$ time, a path that starts at p_1 and visits k points of P, and its length is at most $\|\pi\| + \mathcal{E}_{\pi,\mathfrak{u}}/\mathfrak{u}$.*

Proof: Set $m = \lceil 2\sqrt{2}\mathfrak{u} \rceil$, and use the algorithm presented above. The running time bound follows readily. Thus, we only need to argue that the path computed is indeed within the claimed bound on the length.

Thus, consider the (conceptual) execution of the recursive algorithm over the path π, performing the recursive calls according to π. Specifically, let $\overline{w}$ be a *minimal* window that is visited by the recursive algorithm when applied to π. If an m-perfect cut (for π) exists with respect to $\overline{w}$, then we use it to cut the window $\overline{w}$ into two parts and proceed recursively on each side of the cut. If such an m-perfect cut does not exist for $\overline{w}$ (that is, $\overline{w}$ is m-dense), then we (conceptually) stop, and use the results returned by the recursive call on this window. We claim that for these specific choices, the recursive algorithm computes a path σ, such that $\|\sigma\|$ is as required. Since the recursive algorithm returns a path no longer than σ, this would imply the theorem.

Let $\overline{\mathsf{D}}$ be the set of m-dense windows (which by the algorithm execution are minimal windows) visited by the algorithm when applied to π. Let $\mathcal{S} = \mathcal{S}^{\mathfrak{u}}_{\mathrm{opt}}(\pi)$ be an optimal $\mathfrak{u}$-skeleton for π, and let $\pi_1, \ldots, \pi_{\mathfrak{u}-1}$ be the breakup of π into subpaths by the vertices of $\mathcal{S}$. By Lemma 3.8, we have that

$$\begin{aligned}\mathcal{E}_{\pi,\mathfrak{u}} &= \sum_{j=1}^{\mathfrak{u}-1} \mathcal{E}_{\pi_j,2} \geq \sum_{j=1}^{\mathfrak{u}-1} \sum_{\overline{w}\in\overline{\mathsf{D}}} \Big(\|\pi_j\cap\overline{w}\| - \sqrt{2}\Delta_{\overline{w}} \Big) = \sum_{\overline{w}\in\overline{\mathsf{D}}} \sum_{j=1}^{\mathfrak{u}-1} \Big(\|\pi_j\cap\overline{w}\| - \sqrt{2}\Delta_{\overline{w}} \Big) \\ &= \sum_{\overline{w}\in\overline{\mathsf{D}}} \Big(\|\pi\cap\overline{w}\| - \sqrt{2}(\mathfrak{u}-1)\Delta_{\overline{w}} \Big) \geq \sum_{\overline{w}\in\overline{\mathsf{D}}} \frac{\|\pi\cap\overline{w}\|}{2},\end{aligned}$$

since $\|\pi \cap \overline{w}\| \geq m\Delta_{\overline{w}}$, for each $\overline{w} \in \overline{\mathsf{D}}$ (by Claim 3.3), and $m = \lceil 2\sqrt{2}\mathfrak{u} \rceil \geq 2\sqrt{2}\mathfrak{u}$. Now, note that $\pi(\overline{w}) \cap \overline{w}$ is a subset of $\pi \cap \overline{w}$, and henceforth it holds

$$\mathcal{E}_{\pi,\mathfrak{u}} \geq \sum_{\overline{w} \in \overline{\mathsf{D}}} \frac{\|\pi \cap \overline{w}\|}{2} \geq \sum_{\overline{w} \in \overline{\mathsf{D}}} \frac{\|\pi(\overline{w}) \cap \overline{w}\|}{2}. \tag{1}$$

For an m-dense window $\overline{w} \in \overline{\mathsf{D}}$, the path σ output by the algorithm (when applied to π) inside $\overline{w}$ is of length $\leq (1+1/m) \cdot \|\pi(\overline{w}) \cap \overline{w}\|$, as this is the performance guarantee provided by Mitchell's analysis [Mit99]. Namely, the error introduced by the approximation inside $\overline{w}$ is bounded by $\|\pi(\overline{w}) \cap \overline{w}\| / m$. For windows (visited by the algorithm when applied to π) that are not m-dense, the path σ within them is identical to the path π. Thus, for the path σ, it follows from Eq. (1) that

$$\|\sigma\| - \|\pi\| \leq \sum_{\overline{w} \in \overline{\mathsf{D}}} \frac{\|\pi(\overline{w}) \cap \overline{w}\|}{m} = \frac{2}{m} \sum_{\overline{w} \in \overline{\mathsf{D}}} \frac{\|\pi(\overline{w}) \cap \overline{w}\|}{2} \leq \frac{2\mathcal{E}_{\pi,\mathfrak{u}}}{m} < \frac{\mathcal{E}_{\pi,\mathfrak{u}}}{\mathfrak{u}},$$

since $m \geq 2\sqrt{2}\mathfrak{u}$. ∎

It is possible to prove Lemma 3.8 and Theorem 3.9 directly, by arguing that the skeleton can be replaced by an alternative skeleton that is longer and is still shorter, by the excess in the dense windows, than an optimal path. (Since excess is a global property that is not directly defined for windows, the resulting argument in somewhat more complicated.) We provide the more technical proof above, since it brings to the forefront the notion of surplus. Note that the surplus is decomposition sensitive, as such, it might be much smaller than the excess. Therefore, the analysis of Theorem 3.9 is probably loose, as the bound on the error depends solely on the surplus in the dense windows.

Remark 3.10 As mentioned in Remark 3.4, we ignored the use of bridges in describing Mitchell's algorithm [Mit99]. Our analysis implies that the use of those bridges is restricted only to dense windows, where all we need is the performance guarantees already provided by Mitchell's analysis. In particular, for those dense windows, we can also use Arora's algorithm. This is the main insight we use in extending our algorithm to higher dimensions.

4 An $(\varepsilon, \mathfrak{u})$-approximation algorithm for k-TSP in $\mathbb{R}^d$

In this section, we present an $(\varepsilon, \mathfrak{u})$-approximation algorithm for k-TSP in higher dimensions, by combining Mitchell's methods together with Arora's k-TSP algorithm [Aro98]. Throughout the section, we are concerned with $\mathbb{R}^d$, where $d > 2$ is a fixed constant.

Let $\mathcal{Q}$ be an axis-parallel d-dimensional hypercube that contains the point set P. In the following, m is a fixed constant, π is a given path visiting k points of P. The following definitions are analogous to the ones in Section 3.

Definition 4.1 A closed, axis-parallel d-dimensional box $\overline{w}$ is a *window* if $\overline{w} \subseteq \mathcal{Q}$. The *extent* of a window $\overline{w}$ is the largest side length of $\overline{w}$, and is denoted by $\Delta_{\overline{w}}$.

A $(d-1)$-dimensional hyperplane ℓ is a *cut* for π, with respect to $\overline{w}$, if ℓ is axis-parallel and ℓ intersects $\overline{w}$; ℓ is an *m-perfect cut* for π, with respect to $\overline{w}$, if ℓ intersects the segments of $\pi(\overline{w}) \cap \overline{w}$ at most m times.

Definition 4.2 The combinatorial type of a window $\overline{w}$ with respect to π is the subset of P inside (or on the boundary of) $\overline{w}$ and a listing, for each facet of $\overline{w}$, of the identities of the line segments of $\pi(\overline{w})$ that intersect it. We say that $\overline{w}$ is a *minimal window* if there is no window $\overline{w}'$ that is strictly contained in $\overline{w}$ with the same combinatorial type as $\overline{w}$.

For a minimal window $\overline{w}$, if there is no m-perfect cut for π, with respect to $\overline{w}$, then it is *m-dense*. Namely, any axis-parallel $(d-1)$-dimensional hyperplane that intersects $\overline{w}$ has more than m intersection points with $\pi(\overline{w}) \cap \overline{w}$.

The following claim is an immediate extension of Claim 3.3.

Claim 4.3 *If a window $\overline{w}$ is m-dense then $\|\overline{w} \cap \pi\| \geq m \cdot \Delta_{\overline{w}}$.*

To bootstrap our algorithm, we need a $(1+\varepsilon)$-approximation algorithm for the **WindowTSP** problem in $\mathbb{R}^d$. To this end, note that the **WindowTSP** problem seeks a set of $O(md)$ paths (with prespecified endpoints) that collectively visits a prespecified number of points inside the given window; it is not hard to adapt Arora's technique to solve the **WindowTSP** problem in $n^{O(md)} \cdot O(m \log n)^{(m\sqrt{d})^{O(d)}}$ time, such that the solution has a total length $\leq (1+1/m)L(\overline{w})$, where $L(\overline{w})$ denotes the length of an optimal solution inside the specified window $\overline{w}$. (This problem can be solved even faster, but it has no impact on the overall performance of our algorithm.) The required adaption is straightforward and we omit the tedious but easy details; see [Aro98, AK03]. We denote this subroutine by k**DenseAprxTSP**.

For an instance of the **WindowTSP** problem, the algorithm works as follows. If $\overline{w}$ has at most m points of P in its interior, then the subproblem is solved by brute force. Otherwise, the algorithm chooses the smaller value returned by the following two options.

(a) Use k**DenseAprxTSP** to solve this problem, providing a solution with total length at most $(1+1/m)L(\overline{w})$, where $L(\overline{w})$ denotes the length of an optimal solution inside $\overline{w}$.

(b) Solve the problem recursively, optimizing over all choices associated with an m-perfect cut of window $\overline{w}$. (As in the $\mathbb{R}^2$ case, before performing the recursive calls, we need to shrink the windows formed by the cut into minimal windows.)

   (i) There are $O(d \cdot n^2)$ choices for a cut. More specifically, there are d choices of (axial) directions; and we can always let the cut pass through either a point of P or an intersection point between $\pi(\overline{w})$ and the boundary of $\overline{w}$. Since $\pi(\overline{w})$ is a subset of the set of $\binom{n}{2}$ possible segments (namely, all segments connecting a pair of points of P), it follows that there are $O(n^2)$ possible intersection points between $\pi(\overline{w})$ and the boundary of $\overline{w}$.

   (ii) There are $O(k)$ choices of the number of points visited in new subproblems, subject to the requirement that the total number of points visited within the two subproblems is equal to the number specified in the given instance.

   (iii) There are $O(n^{2m})$ choices of new boundary information on the cut. Specifically, we select $\leq m$ segments (each determined by a pair of points of P) that cross the cut. We require that the boundary information of the new subproblems be consistent with the boundary information of the given instance.

(iv) There are a constant number of choices (since m and d are fixed constants) of connectivity constraints for the two new subproblems determined by the cut, subject to the requirement that these constraints be consistent with the constraints of the given instance.

Let k**TSPAprxAlg** denote this recursive algorithm. One can easily use memoization to turn it into an efficient dynamic programming algorithm. There are $O\big(k \cdot n^{2d} \cdot (n^{2m})^{2d}\big) = O\big(k\, n^{(4m+2)d}\big)$ possible subproblems, since there are $O(k)$ choices for the number of points that should be visited within $\overline{w}$, $O(n^{2d})$ choices of $\overline{w}$, and $O(n^{2m})$ choices of crossing segments on each of the $2d$ facets of $\overline{w}$. (The number of possible connectivity constraints is a constant since m and d are fixed constants.)

Remark 4.4 Before analyzing this algorithm, observe that it can be viewed as the combination of Mitchell's method [Mit99] with Arora's k-TSP algorithm [Aro98]. Specifically, the algorithm top structure follows Mitchell's method. However, conceptually, whenever the algorithm "encounters" a dense window, it uses k**DenseAprxTSP** (which is an easy extension of Arora's k-TSP algorithm).

To see why we had to modify Mitchell's algorithm, observe that the algorithm in Section 3 cannot be used in higher dimensions directly, because part of Mitchell's k-TSP algorithm relies on a crucial property of m-guillotine subdivisions in the plane. Namely, it introduces (and accounts for the additional length of) bridges on the path to decrease the interaction of the path with the outside world when considering dense windows. It is not known how to extend this directly to higher dimensions. However, the need for bridges arises only when a window is dense. In a dense window (in higher dimensions) we can circumvent this issue altogether by using Arora's algorithm (namely k**DenseAprxTSP**). Similarly, using Arora's algorithm on its own does not suffice here, since it introduces errors (by deflecting paths through "portals") even in windows which are not dense.

Analysis. To analyze the algorithm, we extend the definition of surplus (see Definition 3.6) to higher dimensions in a natural way. The following lemma is the analog of Lemma 3.8.

Lemma 4.5 *Let $\overline{\mathsf{D}}$ be a set of interior disjoint windows (inside $\mathcal{Q}$), and let π be a polygonal path inside $\mathcal{Q}$. We have that $\mathcal{E}_{\pi,2} \geq \sum_{\overline{w}\in\overline{\mathsf{D}}} \Big(\|\pi \cap \overline{w}\| - \sqrt{d}\Delta_{\overline{w}} \Big)$.*

The following theorem is similar to Theorem 3.9.

Theorem 4.6 *Let $\pi = \langle p_1, p_2, \ldots, p_k \rangle$ be an arbitrary path that visits k points of P, and let $\mathfrak{u} \geq 2$ be an arbitrary fixed integer. One can compute, in $n^{O(\mathfrak{u}d\sqrt{d})} \cdot (\mathfrak{u}\sqrt{d}\log n)^{(\mathfrak{u}d)^{O(d)}}$ time, a path that starts at p_1 and visits k points of P, and its length is at most $\|\pi\| + \mathcal{E}_{\pi,\mathfrak{u}}/\mathfrak{u}$.*

Proof: Set $m = \Big\lceil 2\sqrt{d} \cdot \mathfrak{u} \Big\rceil$. Observe that the algorithm k**TSPAprxAlg** uses the algorithm k**DenseAprxTSP** inside the dense windows, which provides the required approximation guarantee. The argument now follows the proof of Theorem 3.9 (almost) verbatim, and is thus omitted. ∎

Note, that the algorithm in this section also works for the planar case (namely $d = 2$).

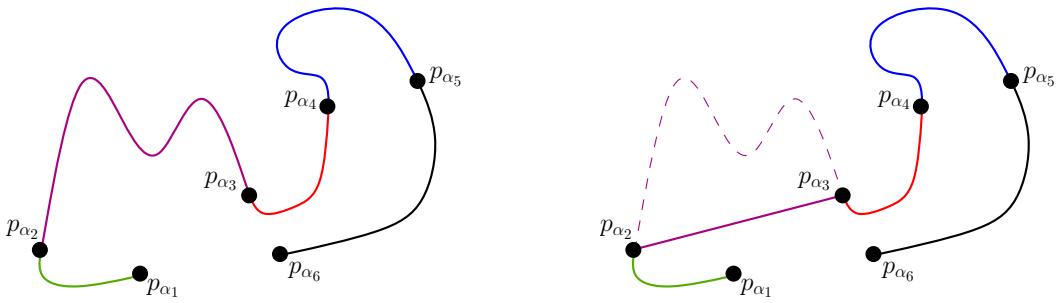


Figure 6: (i) The path π^*_{opt} is divided into $\mathfrak{u} = 5$ subpaths, each of which visits an (roughly) equal number of points. (ii) Since $\mathcal{E}_2 \geq \sum_{i=1}^{\mathfrak{u}} \mathcal{E}_i/\mathfrak{u}$, we obtain the desired path π' by connecting p_{α_2} to p_{α_3}.

5 A PTAS for orienteering

Next, we apply the algorithm of Theorem 4.6 to the rooted orienteering problem.

Lemma 5.1 *Given a set P of n points in $\mathbb{R}^d$, a budget $\mathcal{B}$, and a root $r = p_1$, let $\pi^*_{\mathrm{opt}} = \langle p_1, p_2, \ldots, p_k \rangle$ be an optimal rooted orienteering path starting at r with budget $\mathcal{B}$. One can compute, in $n^{O(d\sqrt{d}/\varepsilon)} \cdot (\sqrt{d}\log n/\varepsilon)^{(d/\varepsilon)^{O(d)}}$ time, a path such that it starts at r and visits at least $(1-\varepsilon)k$ points of P, and its length is at most $\mathcal{B}$.*

Proof: Set $\mathfrak{u} = \lceil 2/\varepsilon \rceil$. Let $\pi^*_{\mathrm{opt}}(i,j) = \langle p_i, p_{i+1}, \ldots, p_j \rangle$ denote the portion of the path π^*_{opt} from p_i to p_j, and let $\mathcal{E}(i,j) = \left\|\pi^*_{\mathrm{opt}}(i,j)\right\| - \|p_i - p_j\|$ denote its 2-excess. Let $\alpha_i = \lceil (i-1)(k-1)/\mathfrak{u} \rceil + 1$. By the definition, we have $\alpha_1 = 1$ and $\alpha_{\mathfrak{u}+1} = k$, and furthermore, each subpath $\pi^*_{\mathrm{opt}}(\alpha_i, \alpha_{i+1})$ visits

$$\alpha_{i+1} - \alpha_i - 1 = \left(\left\lceil \frac{i(k-1)}{\mathfrak{u}} \right\rceil + 1 \right) - \left(\left\lceil \frac{(i-1)(k-1)}{\mathfrak{u}} \right\rceil + 1 \right) - 1 \leq \left\lfloor \frac{k-1}{\mathfrak{u}} \right\rfloor \tag{2}$$

points (excluding the endpoints p_{α_i} and $p_{\alpha_{i+1}}$).

Consider the subpaths $\pi^*_{\mathrm{opt}}(\alpha_1, \alpha_2), \ldots, \pi^*_{\mathrm{opt}}(\alpha_{\mathfrak{u}}, \alpha_{\mathfrak{u}+1})$ of π^*_{opt} and their 2-excesses $\mathcal{E}_1 = \mathcal{E}(\alpha_1, \alpha_2), \ldots, \mathcal{E}_{\mathfrak{u}} = \mathcal{E}(\alpha_{\mathfrak{u}}, \alpha_{\mathfrak{u}+1})$, respectively. Clearly, there exists an index ν, $1 \leq \nu \leq \mathfrak{u}$, such that $\mathcal{E}_\nu \geq (\sum_{i=1}^{\mathfrak{u}} \mathcal{E}_i)/\mathfrak{u}$.

By connecting the vertex p_{α_ν} directly to the vertex $p_{\alpha_{\nu+1}}$ in π^*_{opt}, we obtain a new path $\pi' = \langle p_1, p_2, \ldots p_{\alpha_\nu}, p_{\alpha_{\nu+1}}, p_{\alpha_{\nu+1}+1}, \ldots, p_k \rangle$. Observe that $\|\pi'\| = \left\|\pi^*_{\mathrm{opt}}\right\| - \mathcal{E}_\nu$, and by Eq. (2), π' visits at least $k - (\alpha_{\nu+1} - \alpha_\nu - 1) \geq k - \lfloor (k-1)/\mathfrak{u} \rfloor \geq (1 - 1/\mathfrak{u})k$ points of P. See Figure 6.

Consider the $(\mathfrak{u}+1)$-skeleton $\mathcal{S}' = \langle p_{\alpha_1}, p_{\alpha_2}, \ldots, p_{\alpha_{\mathfrak{u}+1}} \rangle$ of π'. By the definition of $\mathcal{E}_i$, we have that $\|\mathcal{S}'\| = \left\|\pi^*_{\mathrm{opt}}\right\| - \sum_{i=1}^{\mathfrak{u}} \mathcal{E}_i$. Therefore, by the definition of $\mathcal{E}_{\pi',\mathfrak{u}+1}$, we have that

$$\mathcal{E}_{\pi',\mathfrak{u}+1} \leq \|\pi'\| - \|\mathcal{S}'\| = \left(\left\|\pi^*_{\mathrm{opt}}\right\| - \mathcal{E}_\nu\right) - \left(\left\|\pi^*_{\mathrm{opt}}\right\| - \sum_{i=1}^{\mathfrak{u}} \mathcal{E}_i \right) = \sum_{i=1}^{\mathfrak{u}} \mathcal{E}_i - \mathcal{E}_\nu.$$

By applying Theorem 4.6 to the path π', one can compute a path ξ that visits $(1-1/\mathfrak{u})k \geq (1-\varepsilon)k$ points of P, of length

$$\begin{aligned}\|\xi\| &\leq \|\pi'\| + \frac{\mathcal{E}_{\pi',\mathfrak{u}+1}}{\mathfrak{u}+1} \leq \left(\left\|\pi^*_{\mathrm{opt}}\right\| - \mathcal{E}_\nu\right) + \frac{1}{\mathfrak{u}+1}\left(\sum_{i=1}^{\mathfrak{u}} \mathcal{E}_i - \mathcal{E}_\nu\right) \\ &= \left\|\pi^*_{\mathrm{opt}}\right\| + \frac{1}{\mathfrak{u}+1}\left(\sum_{i=1}^{\mathfrak{u}} \mathcal{E}_i - (\mathfrak{u}+2)\mathcal{E}_\nu\right) \leq \left\|\pi^*_{\mathrm{opt}}\right\| \leq \mathcal{B},\end{aligned}$$

since $\sum_{i=1}^{\mathfrak{u}} \mathcal{E}_i - (\mathfrak{u}+2)\mathcal{E}_\nu \leq 0$, implied by $\mathcal{E}_\nu \geq (\sum_{i=1}^{\mathfrak{u}} \mathcal{E}_i)/\mathfrak{u}$. ∎

Of course, the value of k is not known in advance. Therefore, the algorithm tries all possible values of k from 1 to n, and returns the maximum value such that k points of P can be visited within the budget $\mathcal{B}$.

Theorem 5.2 *Given a set P of n points in $\mathbb{R}^d$, a budget $\mathcal{B}$, and a root r, let k_{opt} be the number of points of P visited by an optimal orienteering path starting at r with budget $\mathcal{B}$. One can compute, in $n^{O(d\sqrt{d}/\varepsilon)} \cdot (\sqrt{d}\log n/\varepsilon)^{(d/\varepsilon)^{O(d)}}$ time, a path that starts at r and visits at least $(1-\varepsilon)k_{\mathrm{opt}}$ points of P, and its length is at most $\mathcal{B}$.*

6 Conclusions

In this paper, we defined the notion of $(\varepsilon, \mathfrak{u})$-approximation to k-TSP, and showed that Mitchell's k-TSP algorithm [Mit99] works actually as an $(\varepsilon, \mathfrak{u})$-approximation algorithm for the k-TSP problem in the plane. We used it to develop a $(1-\varepsilon)$-approximation algorithm for the orienteering problem. The analysis easily extends to handle the case where both the starting and ending vertex of the orienteering problem are specified. In particular, the algorithm can approximate the best orienteering cycle rooted at a point r.

Our algorithm sheds a light on the power of Mitchell's approach [Mit99] which has the advantage that it introduces errors only when the underlying path is "dense". This is in contrast to the Arora's technique [Aro98] which inherently introduces error in the approximation generated.

In the new analysis of the k-TSP algorithm the notion of surplus emerges naturally. We expect it to be much smaller than the excess in a lot of cases, and it might be of independent interest and useful in analyzing other algorithms.

There are numerous problems for further research, including:

- Can the running time be significantly improved?
- Can one extend the algorithms presented here to the problem of visiting points with time windows constraints [BES05, BBCM04, CP05], where one has to visit a point inside a prespecified time window? This problem seems to be more challenging. Currently, even a constant-factor approximation algorithm is not known for the simple case of visiting points on the line.

Acknowledgments

The authors would like to thank the anonymous referees for their useful comments.

References

[AK03] S. Arora and G. Karakostas. Approximation schemes for minimum latency problems. *SIAM J. Comput.*, 32(5):1317–1337, 2003.

[AMN98] E. M. Arkin, J. S. B. Mitchell, and G. Narasimhan. Resource-constrained geometric network optimization. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 307–316, 1998.

[Aro98] S. Arora. Polynomial time approximation schemes for euclidean TSP and other geometric problems. *J. Assoc. Comput. Mach.*, 45(5):753–782, Sep 1998.

[Aro03] S. Arora. Approximation schemes for NP-hard geometric optimization problems: a survey. *Math. Prog.*, 97:43–69, 2003.

[BBCM04] N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Approximation algorithms for deadline-TSP and vehicle routing with time-windows. In *Proc. 36th Annu. ACM Sympos. Theory Comput.*, pages 166–174, 2004.

[BCK+03] A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward TSP. In *Proc. 44th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 46–55, 2003.

[BES05] R. Bar-Yehuda, G. Even, and S. Shahar. On approximating a geometric prize-collecting traveling salesman problem with time windows. *J. Algorithms*, 55(1):76–92, 2005.

[CP05] C. Chekuri and M. Pál. A recursive greedy algorithm for walks in directed graphs. In *Proc. 46th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 245–253, 2005.

[Gar05] N. Garg. Saving an epsilon: a 2-approximation for the k-MST problem in graphs. In *Proc. 37th Annu. ACM Sympos. Theory Comput.*, pages 396–402, 2005.

[Mit99] 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.*, 28:1298–1309, 1999.

[Mit00] J. S. B. Mitchell. Geometric shortest paths and network optimization. In Jörg-Rudiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, chapter 15, pages 633–701. Elsevier, 2000.

[TV02] P. Toth and D. Vigo, editors. *The vehicle routing problem.* SIAM Monographs on Discrete Mathematics and Applications, 2002.