Touring a Sequence of Polygons

Moshe Dror
The University of Arizona
Tucson AZ 85721
mdror@eller.arizona.edu

Alon Efrat
The University of Arizona
Tucson AZ 85721
alon@cs.arizona.edu

Anna Lubiw
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
alubiw@uwaterloo.ca

Joseph S. B. Mitchell
Stony Brook University
Stony Brook, NY 11794-3600
jsbm@ams.sunysb.edu

January 6, 2004

Abstract

Given a sequence of k polygons in the plane, a start point s, and a target point, t, we seek a shortest path that starts at s, visits in order each of the polygons, and ends at t. If the polygons are disjoint and convex, we give an algorithm running in time $O(kn\log(n/k))$, where n is the total number of vertices specifying the polygons. We also extend our results to a case in which the convex polygons are arbitrarily intersecting and the subpath between any two consecutive polygons is constrained to lie within a simply connected region; the algorithm uses $O(nk^2\log n)$ time. Our methods are simple and allow shortest path queries from s to a query point t to be answered in time $O(k\log n + m)$, where m is the combinatorial path length. We show that for nonconvex polygons this "touring polygons" problem is NP-hard.

The touring polygons problem is a strict generalization of some classic problems in computational geometry, including the safari problem (find a shortest tour visiting a set of convex cages attached to the inside wall of a simple polygon), the zoo-keeper problem, and the watchman route problem in a simple polygon (find a shortest tour that sees all points of the polygon). Our new results give an order of magnitude improvement in the running times of the safari problem and the watchman route problem: We solve the safari problem in $O(n^2\log n)$ time and the watchman route problem (through a fixed point s) in time $O(n^3\log n)$, compared with the previous time bounds of $O(n^3)$ and $O(n^4)$, respectively. Additionally, our work is motivated by a cutting stock problem in which one must find an optimal cutting path for cutting polygonal parts from a sheet of material, while satisfying certain constraints.

1 Introduction

A number of problems in computational geometry—including the zookeeper problem, the safari problem, and the watchman route problem—involve finding a shortest path, within some constrained region, that visits a set of convex polygons in an order that is given, or can be deduced. See below for definitions of the three problems mentioned above, and see Figure 2 for examples. When region constraints permit it, a locally shortest path may travel straight through a polygon; otherwise, it "reflects" off edges or vertices of the polygons. The three problems mentioned above have in common that a path is globally shortest if it is locally shortest—i.e., it "reflects" appropriately whenever it changes direction.

In this paper we give the first polynomial-time algorithm for the general *touring polygons problem*: to find a shortest path, between two specified points, that visits in order a sequence of k possibly intersecting convex polygons while lying in some constrained regions. We crucially use the property that local optimality of a path implies global optimality. One consequence is that we never need to compute distances; we only need to compute reflections of points with respect to lines. If the given polygons are not convex, this property fails, and we prove that the touring polygons problem is NP-hard.

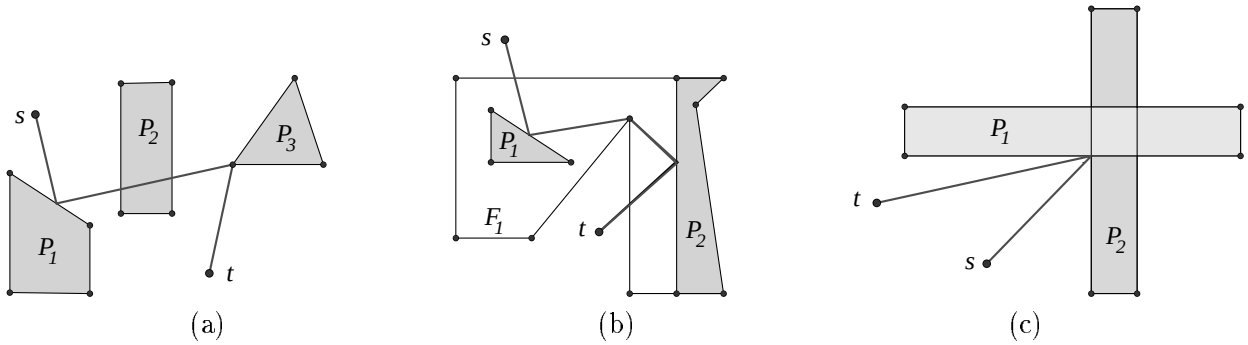

Figure 1: The TPP problem: (a) unconstrained; (b) with a fence F_1; (c) with intersecting polgyons.

Many geometric shortest path problems, e.g., shortest paths among obstacles, are solved using the general technique of computing a *shortest path map* with respect to a fixed starting point s, which is a subdivision of the plane into regions such that the shortest paths from s to all points t in one region are combinatorially equivalent. (In particular, the most efficient algorithm for the zookeeper problem works this way.) One cannot directly apply this technique to our problem since the number of combinatorially distinct shortest paths in the touring polygons problem can be exponential in the input size, as we show in Theorem 3. We use instead a new technique based on subdividing the plane only according to the combinatorial type of the *last step* of the path. We call this a *last step shortest path map*. Our algorithms are based on computing the last step shortest path maps iteratively, one for each of the k polygons that must be visited, in the order they are given. Given the sequence of such maps, a query for the shortest path from s to any query point t can be answered in time $O(k \log(n/k))$: a query for t in the final last step shortest path map tells us whether the path to t goes through, or reflects from P_k, thus determining a query point for the previous last step shortest path map.

The general *touring polygons problem* (TPP) is as follows. We are given a start point $s = P_0$, a target point $t = P_{k+1}$, and a sequence of polygons, $P_1, \ldots, P_k$, possibly intersecting, each considered to be a closed region in the plane (i.e., each includes interior plus boundary). For each $i = 0, \ldots, k$, we are given a simply connected polygonal region F_i, called a "fence", with F_i containing P_i and P_{i+1}. We say that a path π *visits* polygon P_i if $\pi \cap P_i \neq \emptyset$. We say that π visits the sequence $P_1, \ldots, P_k$ if there exist points $q_1 \in P_1$, $q_2 \in P_2, \ldots, q_k \in P_k$ such that $q_1, \ldots, q_k$ appear in order along π. We let $p_0 = s$ and let p_i denote the first point of π (i.e., the point closest to s along the path) that lies within P_i *and* comes after, or equal to, p_{i-1} along π; such points exist if π visits the sequence $P_1, \ldots, P_k$. Our goal is to compute a shortest path that begins at s, visits $P_1, P_2, \ldots, P_k$, in that order, and arrives at t, with each subpath of π from p_{i-1} to p_i lying within the fence F_{i-1}. In the *unconstrained* TPP, each of the fences F_i is the entire plane, $\mathbb{R}^2$.

An *i-path* is a path that starts at s and visits the sequence of polygons $P_1, \ldots, P_i$. A *fenced i-path* to a point $q \in F_i$ is an i-path satisfying the relevant fence constraints, i.e., having the subpath from p_{j-1} to p_j lying within F_{j-1}, for $j = 1, \ldots, i$ and the subpath from p_i to q in F_i. Using

this notation, the TPP asks for a shortest fenced k-path to t. For a point x, let $\pi_i(x)$ denote a shortest fenced i-path that ends at x. We let n denote the total number of vertices in the polygons $P_1, \ldots, P_k$ and the fences $F_0, \ldots, F_k$.

The formulation of the TPP allows for both the cycle and the path versions of the problem: If the destination point t is the same as the starting point s, we are computing a shortest cycle ("tour") through s visiting the P_i's.

If the order in which the polygons P_i must be visited is not specified, then the touring polygons problem becomes the Traveling Salesperson Problem with Neighborhoods, which is NP-hard, since the case of point polygons is the Traveling Salesperson Problem. See [20].

The touring polygons problem can be modeled as a special kind of 3-dimensional shortest path problem among polyhedral obstacles. Imagine k very large sheets of paper stacked up in closely spaced parallel planes orthogonal to the z-axis. The i-th sheet has a hole, P_i, cut out of it; each sheet of paper is a polyhedral obstacle. The point s is placed at a z-coordinate just below the first sheet and the point t at a z-coordinate just above the kth sheet. Projecting this configuration to the (x, y) plane yields the original touring polygons input. A solution to the touring polygons problem is the projection of a shortest path in 3-dimensions that starts from s, threads its way through the polygonal holes, and arrives at t.

Summary of Results:

(1) We give an $O(kn\log(n/k))$ time ($O(n)$ space) algorithm for the unconstrained TPP with disjoint convex polygons P_i. No polynomial-time algorithm was previously known for this problem. For fixed s, the data structure supports $O(k\log(n/k))$-time shortest path queries to t.

(2) We give an $O(nk^2\log n)$ time ($O(nk)$ space) algorithm for the general TPP with possibly intersecting convex polygons P_i and arbitrary fences F_i. Actually, we only require convexity of the part of P_i's boundary from which fenced rays may bounce.

(3) We show that the full combinatorial shortest path map for the TPP has worst-case size $\Theta((n-k)2^k)$. It can be computed in output-sensitive time and supports $O(k+\log n)$-time shortest path queries.

(4) We show that the unconstrained TPP is NP-hard in general, for intersecting non-convex polygons P_i. A fully polynomial time approximation scheme follows from results for shortest paths among obstacles in $\mathbb{R}^3$.

(5) We give substantial improvements in the running time for two classical problems in computational geometry, namely (a) the safari problem, for which our results imply an $O(n^2\log n)$ algorithm (improving upon $O(n^3)$); and (b) the (fixed-source) watchman route problem, for which our results imply an $O(n^3\log n)$ algorithm (improving upon $O(n^4)$). (By the results of Tan [26], our improved bound for the fixed-source watchman route problem also implies an improved bound—$O(n^4\log n)$ instead of $O(n^5)$—for the "floating" watchman route problem.)

One of the main significances of our new method is not just that it yields a substantially faster algorithm for the watchman route problem, but also that it avoids using dynamic programming and complicated path "adjustments" arguments (which were the source of the original erroneous claims of the problem's polynomial-time solvability).

(6) Our results apply also to give the first polynomial-time solution to the *parts cutting problem* (below). Our algorithm solves this problem in time $O(kn\log(n/k))$.

Our paper is organized as follows: Section 2 describes in detail the applications of our problem, including the watchman route problem, the zookeeper and safari problems, and the "parts cutting problem". In Section 3 we describe the solution to the TPP with disjoint convex polygons and in Section 4 we generalize to the case of intersecting polygons and fence restrictions. Section 5 analyzes the size of the full combinatorial shortest path map for the TPP, and Section 6 proves NP-hardness for non-convex polygons.

2 Applications and Related Work

2.1 The Parts Cutting Problem

Suppose we have a sheet of some material such as wood or sheet metal, and a collection $P_1, \ldots, P_k$ of disjoint polygons on the sheet that must be cut out with a laser or saw as efficiently as possible. We limit the options by requiring that the edges of each polygon P_i be cut in one continuous cycle, starting at some point $p_i \in \partial P_i$. The cutter starts at point s and the objective is to determine the points p_i that minimize the length of the polygonal chain $(s, p_1, p_2, \ldots, p_k)$, since the total length of the cutter path is the length of this chain plus a constant equal to the sum of the perimeters of the parts P_i. We thus want a minimum length path that visits the boundaries of all the polygons. We suppose, furthermore, that we are given the order in which the polygons must be visited. (Certain applications specify the order in which parts must be cut; without a given order, the problem is obviously NP-hard.) This problem is known as the *parts cutting problem*. See Figure 2(a). The parts cutting problem is exactly the unconstrained TPP with disjoint P_i's and $F_i = \mathbb{R}^2$ for all i. This formulation assumes that the cutting tool is like a laser in that it can be turned off as it crosses a polygon.

Our algorithm solves the parts cutting problem in time $O(kn \log(n/k))$; see Section 3.

In Dror [16] the convex polygon cutting problem is considered in the L_∞ metric (which models the case in which the cutting tool consumes time proportional to the longer of the x- or y-movements); linear programming is used to obtain a polynomial-time solution.

2.2 The Safari and Zookeeper Problems

For the safari and zookeeper problems, we are given a simple polygon P, and disjoint convex polygons ("cages") $P_1, \ldots, P_k$ inside P, each sharing exactly one edge with P. In the zookeeper problem we seek a shortest tour inside P that visits each of the P_i's but never enters any of them. In the safari problem we seek a shortest tour inside P that visits each of the P_i's, and *is* allowed to enter them. (We are safe inside our jeep.) See Figure 2(b), (c). In this paper we consider only the "fixed-source" versions of these problems in which the tour must go through a given point, s, on the boundary of P. (See Tan [27] and Tan and Hirata [24] for results on the "floating" version of the zookeeper and safari problems.)

The zookeeper problem, introduced by Chin and Ntafos [8], has a recent $O(n \log n)$ time algorithm due to Bespamyatnikh [3] utilizing the full shortest path map. The safari problem was introduced by Ntafos [21], who claimed an $O(n^3)$ time algorithm, which was then improved to $O(n^2)$ [24]. However, Tan and Hirata [28] found an error in the earlier analysis and presented the current best algorithm with running time $O(n^3)$. In contrast with the zookeeper problem, the full shortest path map for the safari problem has exponential size, as we show in Theorem 3.

Our algorithm solves a much more general problem than the safari and zookeeper problems, and improves the running time for the safari problem to $O(kn \log n)$. Although these problems seem different from the TPP in that no ordering of the P_i's has been specified, it is easy to argue

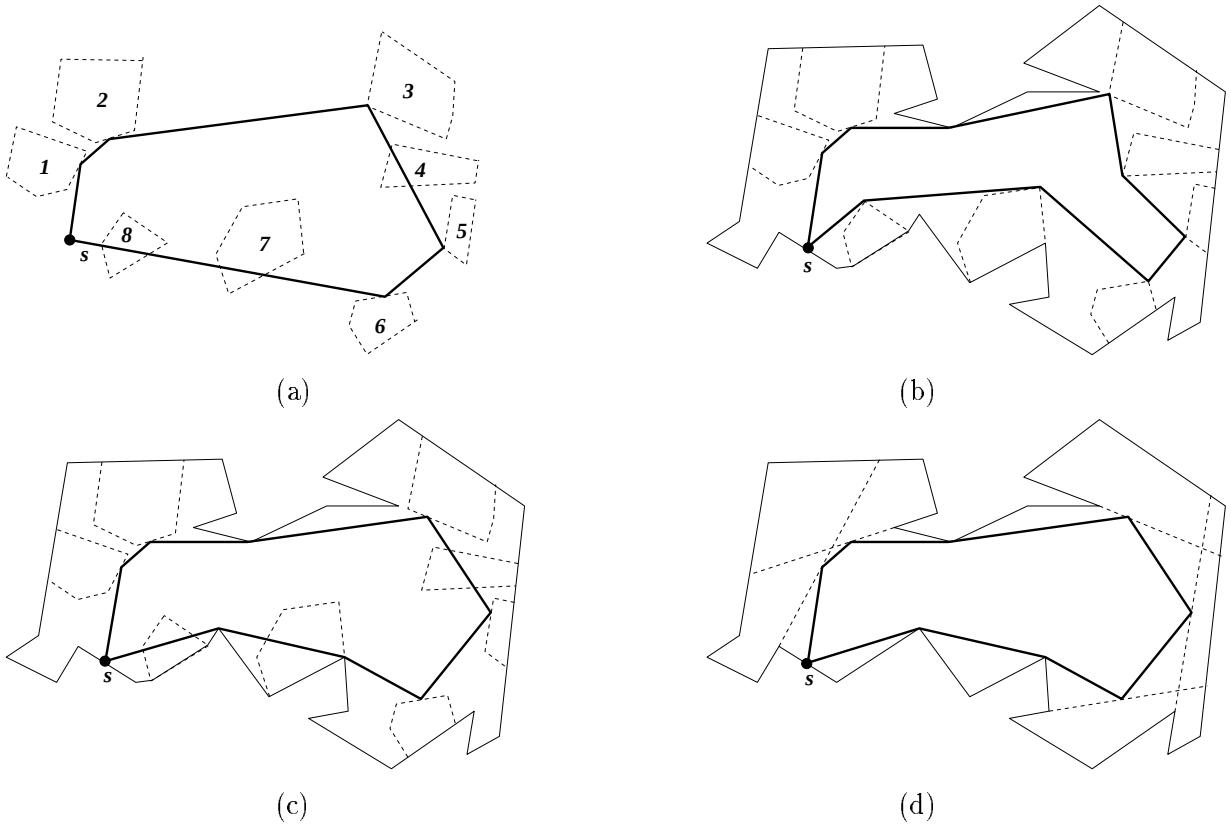

Figure 2: Examples of application problems: (a) parts cutting problem; (b) zookeeper problem; (c) safari problem; (d) watchman route problem.

that a shortest tour must visit the P_i's in the same order as they meet the boundary of P ([8, 21]). (Note that this is not true for the *path* version of the problem, but only for the cycle version; see the thesis of H. Jonsson.) [We need a reference here.] Thus, in order to model the safari and ⇐
zookeeper problems as special cases of the TPP it only remains to specify the fence constraints. For the safari problem, the tour must lie inside the polygon P. For the zookeeper problem, the tour must lie inside the smaller polygon $P' = P - \cup_i P_i$. We could set each F_i to be the whole polygon P (for the safari problem) or the polygon P' (for the zookeeper problem), but our improved time bound is achieved using smaller fences, as described in Section 4.

2.3 The Watchman Route Problem

In the watchman route problem (WRP) we are given a simple polygon P, and the goal is to find a shortest tour inside P so that every point in P is seen from some point of the tour. See Figure 2(d). We consider the "fixed-source" case of the WRP, in which a point s on the boundary of P is specified as the starting point of the tour; the "floating" WRP, with no point s specified, can be solved in time $O(n)$ times that of the fixed WRP, as shown recently by Tan [26].

The WRP was introduced by Chin and Ntafos [9, 7], who claimed in [7] an $O(n^4)$ time algorithm for the fixed WRP in a simple polygon. However, years later, there was a flaw discovered in their algorithm, and several attempts were made to correct it (some of which were themselves flawed). The best current algorithm for the fixed WRP, based on a relatively complex dynamic programming algorithm, is due to Tan, Hirata, and Inagaki [25] and runs in time $O(n^4)$.

Our results imply a new, simpler algorithm for the fixed WRP that runs in time $O(n^3 \log n)$. In order to see that the WRP is a special case of the TPP, we recall the notion (e.g., from [7]) of *essential cuts* and the *essential pockets* they define: A *cut*, c_i, with respect to s is a chord defined by extending the edge e incident on a reflex vertex, v_i, where e is chosen to be that edge incident on v whose extension creates a strictly convex vertex at v in the piece containing s. The portion of P on the side of c_i not containing s is the *pocket*, P_i, induced by the cut c_i. A pocket is *essential* if no other pocket is fully contained within it. A tour sees all of P if and only if it visits all essential pockets. Let $P_1, \ldots, P_k$ denote the sequence of essential pockets, clockwise around ∂P starting from s. The P_i's are not necessarily convex but our algorithm for the general TPP only requires convexity of the portion of P_i from which paths can bounce, which in this case is a straight line segment. It is known that the shortest watchman tour visits this sequence of essential pockets in order. [I think we should justify this in the journal version! Anna.] We then get an instance of the ⇐
TPP by defining each fence F_i to be the bounding polygon P, and, as we will show in Section 4, our algorithm runs in time $O(n^3 \log n)$, improving the previous $O(n^4)$ bound.

3 The Unconstrained TPP for Disjoint Convex Polygons

In this section we give an efficient algorithm for the convex parts cutting problem, which is the special case of the TPP in which the polygons P_i are disjoint and convex and there are no fence constraints. Given points s and t, and a sequence of disjoint convex polygons $P_1, \ldots, P_k$, our goal is to find $\pi_k(t)$, a shortest path that starts at s, visits the polygons in order, and arrives at t—i.e., find a shortest k-path from $s = P_0$ to t.

First, in Section 3.1, we show that it suffices to find a locally shortest path—one that bounces correctly. Then, in Section 3.2 we show that locally shortest paths can be captured by last step shortest path maps; and in Section 3.3, we present an algorithm to build shortest path maps, and, from them, compute locally shortest paths.

3.1 Local Optimality

Clearly we only need to consider k-paths that are polygonal chains. Such a path is *locally shortest* if bends occur only at points on the boundaries of the P_i's, and for each such bend point $b \in \partial P_i$, moving b slightly in either direction on the boundary of P_i while keeping other bends fixed makes the path longer. This implies that for a bend point b on the interior of an edge e of P_i, the angle between the incoming segment $\overline{ab}$ and e must be equal to the angle between the outgoing segment $\overline{bc}$ and e—i.e., the path must reflect on e, so that $b \in \overline{ac'}$, where c' is the reflection of c with respect to the line through e. See Figure 3.

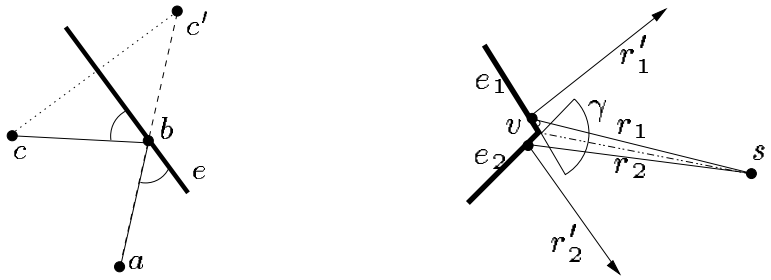

Figure 3: Locally shortest paths reflecting from an edge and a vertex.

For a bend point at a vertex $v = e_1 \cap e_2$ the condition of being locally shortest is that the outgoing path segment from v must leave v in the cone, γ, bounded by the rays r_1' and r_2' formed by reflecting the segment $\overline{sv}$ in the edges e_1 and e_2, respectively.

The following is a special case of Lemma 7.

Lemma 1. *For any $p \in \mathbb{R}^2$ and any $i \in \{0, \ldots, k\}$, there is a unique locally shortest i-path, $\pi_i(p)$, to p. Thus, local optimality is equivalent to global optimality.*

Let T_i be the *first contact set* of P_i, i.e., the points where a shortest $(i-1)$-path first reaches a point of P_i after visiting $P_1, \ldots, P_{i-1}$. See Figure 4. As discussed in Section 1, any i-path π, contains points $p_0 = s$, and p_i, defined to be the first point of π that lies in P_i and does not come before p_{i-1} along π. Using this notation, T_i is the locus of possible points p_i taken over i-paths to all possible target points in the plane. Since the P_j's are disjoint, T_i is contained in the boundary of P_i. The more general Corollary 1 then implies:

Lemma 2. *Each first contact set T_i is a (connected) chain on the boundary of P_i.*

3.2 The Last Step Shortest Path Map

Locally shortest i-paths may leave points of T_i only in certain directions, either continuing in a straight line, or properly reflecting. For p in T_i let $r_i^s(p)$ be the set of rays of locally shortest i-paths going straight through p, and let $r_i^b(p)$ be the set of rays of locally shortest i-paths properly reflecting at p. (The mnemonic is "s" for straight, "b" for bounce.) Lemma 1 implies that for p in an edge of T_i, $r_i^s(p)$ and $r_i^b(p)$ each contain a single ray, and for p a vertex of T_i, $r_i^s(p)$ contains a single ray and $r_i^b(p)$ consists of a cone.

Let $r_i(p) = r_i^s(p) \cup r_i^b(p)$ and $R_i = \cup_{p \in T_i} r_i(p)$. The set R_i is an infinite family of rays with the property that each point in the plane is reached by exactly one ray; we say that R_i is a *starburst* with *source* T_i. (See the more general Lemma 6.) Associated with R_i is a subdivision, $\mathcal{S}_i$, of the plane that groups together points reached by rays of R_i that leave from the same vertex of T_i, or

leave from the same side of the same edge of T_i. We call the polygonal subdivision $\mathcal{S}_i$ the *last step shortest path map* (with respect to s) associated with P_i, since it encodes the information about the last segment in a shortest i-path from s to any point p. See Figure 4 for an example. In more detail, $\mathcal{S}_i$ decomposes the plane into cells σ of two types: (1) cones with an apex at a vertex v of T_i, whose bounding rays are those of $r_i^b(v)$; and (2) unbounded three-sided regions associated with an edge e of T_i. Cells of type (2) can be further classified as being *reflection cells* or *pass-through cells*, depending on whether the rays reflect off e or enter the interior of P_i at e. For cells of type (1), v is the *source* of the cell; for cells of type (2), e is the *source*. The union of all pass-through cells defines the *pass-through region*.

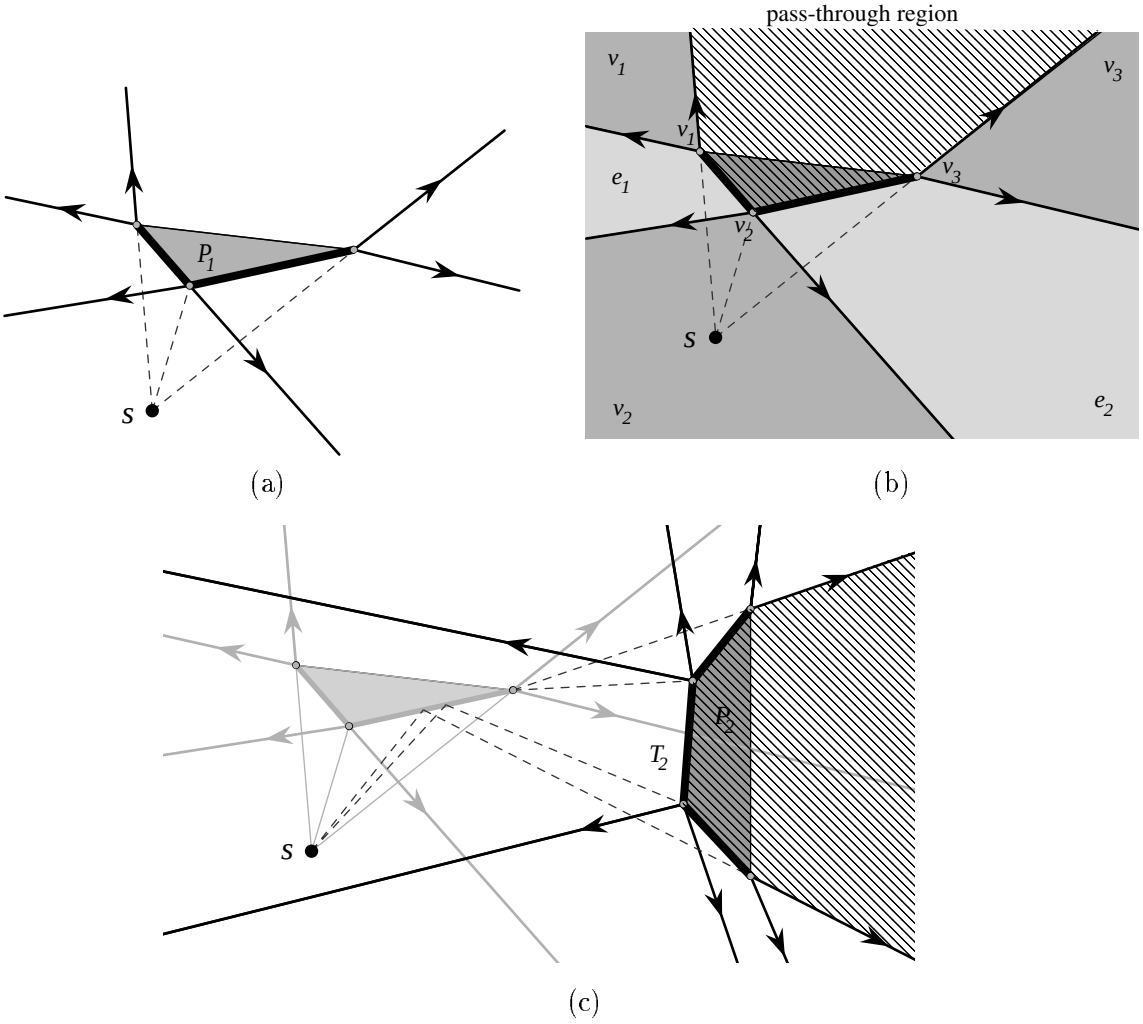

Figure 4: Example of the last step shortest path map: (a) for polygon P_1: T_1 (drawn heavy) and the rays of R_1 bounding the regions of S_1; (b) regions of S_1 labelled as "pass-through" or with their vertex or edge source; (c) for polygon P_2: T_2 (drawn heavy) and the rays of R_2 bounding the regions of S_2—the pass-through region is shaded.

3.3 The Algorithm

Using the last step shortest path maps, $\mathcal{S}_1, \ldots, \mathcal{S}_i$, we can readily compute a shortest i-path to any query point $q \in \mathbb{R}^2$ as follows: See Figure 5 for an example. Locate q in $\mathcal{S}_i$. If the source of the

cell σ containing q is a vertex, v, of T_i, then the last segment of $\pi_i(q)$ is $\overline{vq}$, and we recursively compute the shortest $(i-1)$-path to v (locating v in $\mathcal{S}_{i-1}$, etc.). If the source of σ is an edge e of T_i, then there are two subcases: (a) cell σ is a pass-through cell, in which case $\pi_i(q) = \pi_{i-1}(q)$ and we recursively compute the shortest $(i-1)$-path to q; or (b) cell σ is a reflection cell, in which case we let q' be the reflection of q in the line through e, and recursively compute the shortest $(i-1)$-path to q'; replacing the portion of this path from e to q' by the segment from e to q yields the shortest i-path to q.

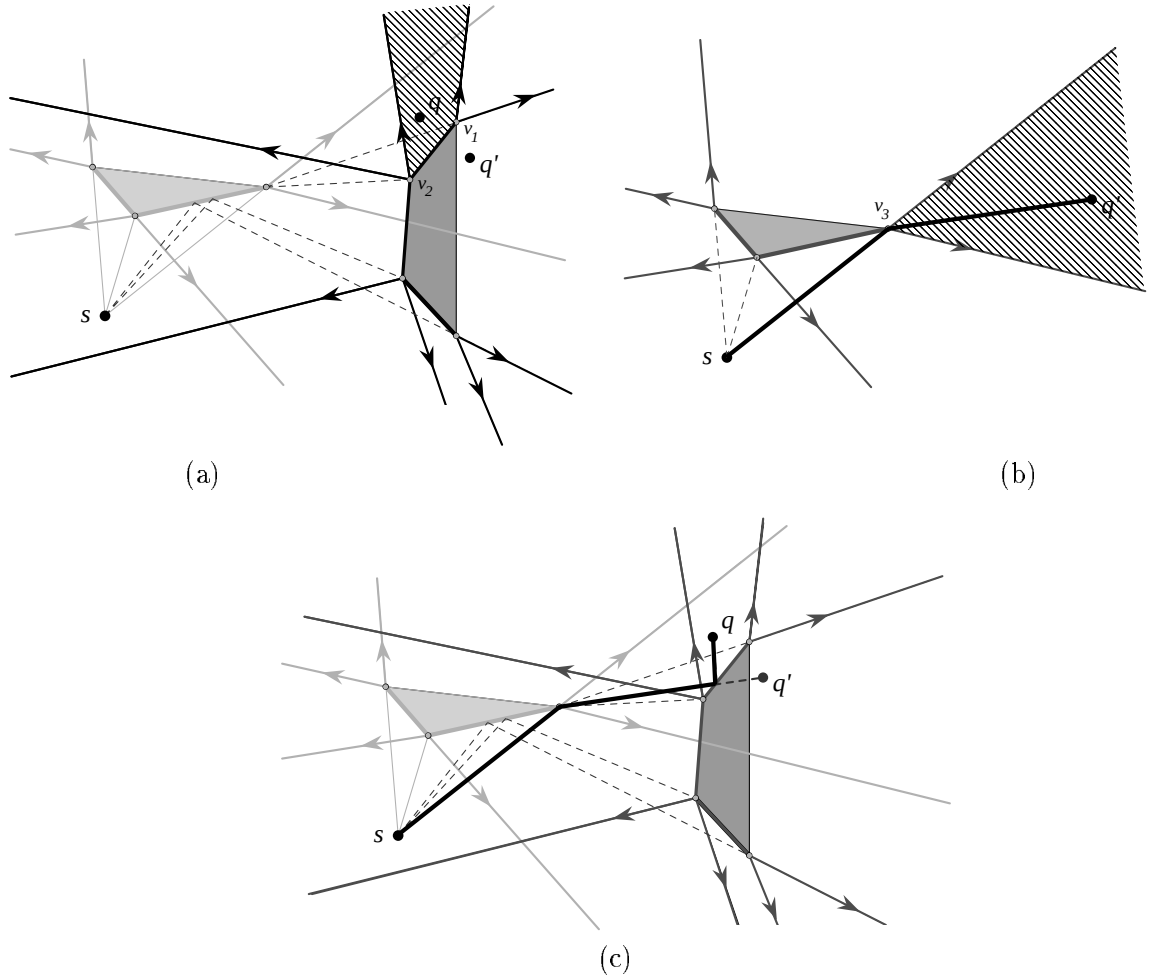

Figure 5: Example query for a shortest 2-path to q: (a) q located in a reflection cell of S_2 with source (v_1, v_2), and the reflection, q', of q in the line through this edge; (b) q' located in S_1 and a shortest 1-path to it; (c) the final path from s to q.

Lemma 3. *Given $\mathcal{S}_1, \ldots, \mathcal{S}_i$, the path $\pi_i(q)$ can be determined in time $O(k \log(n/k))$ for any query point $q \in \mathbb{R}^2$.*

Proof. The complexity of each $\mathcal{S}_i$ is clearly $O(|P_i|)$ (for $i = 1 \ldots k$), and can be stored in a point location data structure supporting $O(\log |P_i|)$-time queries for q in $\mathcal{S}_i$. The time to find $\pi_i(q)$ is then $O(\sum_1^k \log(|P_i|))$, which attains its maximum when each $|P_i| = n/k$. □

We construct each of the subdivisions $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ iteratively, using the data structures $\mathcal{S}_1, \ldots, \mathcal{S}_{i-1}$ in the construction of $\mathcal{S}_i$. We store the subdivisions in a point location data structure.

The algorithm is very simple: For each vertex v of P_i, we compute $\pi_{i-1}(v)$. If this path arrives at v from the inside of P_i then v is not a vertex of T_i. Otherwise it is, and the last segment of $\pi_{i-1}(v)$ determines the rays $r_i^b(v)$ and $r_i^s(v)$ that define the subdivision $\mathcal{S}_i$. Thus, combining with Lemma 3 we have:

Theorem 1. *For the unconstrained TPP for disjoint convex polygons with input size n, a data structure of size $O(n)$ can be built in time $O(kn\log(n/k))$ that enables shortest i-path queries to any query point q to be answered in time $O(i\log(n/k))$.*

4 The General TPP

We turn our attention now to the general touring polygons problem with intersecting polygons and fence constraints. See Figure 1(b), (c). As stated in the introduction, we assume that fence F_i contains P_i and P_{i+1}, and we require that the subpath from p_i to p_{i+1} lie inside F_i. (Recall that p_i is the first point of π that lies within P_i and comes after or equal to p_{i-1} along π.) We solve a more general case than that of convex P_i's. Define the *facade* of P_{i+1} in F_i to be $\partial P_{i+1} \cap cl(F_i - P_{i+1})$. We assume that the facade of P_{i+1} in F_i is a (single) convex chain respecting the inside of P_{i+1}; i.e., adding the segment to close the chain yields a convex polygon (possibly degenerating to a line segment) whose inside corresponds to the inside of P_{i+1}. The facade represents the points on the boundary of P_{i+1} from which rays in F_i can bounce. For example, in Figure 1(b) polygon P_2 is not convex but its facade—a single line segment—is.

4.1 Local Optimality

In the presence of fences, a locally shortest path may bend at a fence vertex, with the usual condition for shortest paths inside a polygon—namely that the angle interior to the polygon be reflex. See Figure 1(b). Intersecting polygons also complicate things: a path is locally shortest at an intersection point p of ∂P_{i-1} and ∂P_i if it is shorter than paths that visit P_{i-1} and then P_i close to p. See Figure 1(c).

We again use last step shortest path maps. As before, we let T_i be the *first contact set* of P_i, i.e., the locus of points where a locally shortest fenced $(i-1)$-path first reaches a point of P_i after visiting $P_1, \ldots, P_{i-1}$, and then travelling though F_{i-1} to P_i. Because polygons may intersect, first contact may occur when a locally shortest fenced $(i-1)$-path ends at a point that is already inside P_i; thus T_i need not be part of P_i's boundary. For example, if s is contained in P_1, then T_1 is just the point s. In general, T_i will consist of the portions of T_{i-1} that lie inside P_i together with parts of the boundary of P_i. As before, we define R_i to be the set of rays leaving points of T_i (either straight through or bouncing) to begin locally shortest fenced i-paths. [Make some figures.]

The goal of this subsection is to prove that a locally shortest path is globally shortest. In order to do that we need to prove some results on T_i and R_i. We begin by describing T_i in terms of T_{i-1} and the fence F_{i-1}. T_i can be described as the set of points $p \in P_i$ such that there is a shortest path inside F_{i-1} starting along a ray of R_{i-1} from some point of T_{i-1}, and intersecting P_i for the first time at p. Let A_{i-1} be the last rays of such paths. We include in A_{i-1} the rays of R_{i-1} leaving points $p \in T_{i-1} \cap P_i$. Sources for the rays of A_{i-1} are vertices of T_{i-1}, points on edges of T_{i-1}, and vertices of F_{i-1}. In the absence of fences, A_{i-1} was a subset of R_{i-1}, but now F_{i-1} intervenes.

Recall that a set of rays is a *starburst* if the rays cover the plane, and are disjoint except that they may share source points. We will assume by induction that T_{i-1} is a tree and R_{i-1} is a starburst. We will prove that any two distinct rays of A_{i-1} do not intersect; that R_i is a starburst;

and that T_i is a tree. [We need a *reflecting* starburst where two rays off an edge are reflections of each other.] ⇐

We begin with a basic result to handle fences, generalizing the uniqueness of locally shortest s-t paths in a polygon to the case where the source is a starburst.

Lemma 4. *Suppose F is a simple polygon, and we have rays R emanating in a starburst from a connected source T inside F. Then any point p interior to F is reached by a unique locally shortest path starting from a ray of R.*

Proof. This is true by definition of a starburst if all points in F are reached directly by rays of the starburst, i.e., every point $p \in F$ is reached by a ray $r \in R$, emanating from some $t \in T$, with the line segment $\overline{tp}$ contained in F. Otherwise there must be a ray r of the starburst that travels in F to a reflex vertex v of F and then cuts off a hidden pocket P in F. By induction on the number of pockets, we can assume that locally shortest paths from the starburst are unique in $F - P$. In particular, r itself provides a unique locally shortest path. Thus locally shortest paths from the starburst to points inside P go through v. To complete the proof, observe that locally shortest paths inside P from v are unique—this is the single source case. □

Lemma 5. *If R_{i-1} is a starburst from a tree T_{i-1}, then the first contact set, T_i, is connected, and no two distinct rays of A_{i-1} intersect (except that rays may begin at a common point).*

Proof. Note that disjointness of the rays of A_{i-1} is more general than the lemma above since the intersection point of the rays may potentially lie outside F_{i-1}, and in fact we crucially need the assumption that the facade of P_i in F_{i-1} is connected and convex. See Figure.

The proof is by induction on the complexity of the fence. Without a fence, disjointness of the rays of A_{i-1} is trivial, but connectedness of T_i requires proof, particularly for intersecting polygons. In the general induction step, connectedness of T_i comes for free, but disjointness of A_{i-1} requires some work.

In more detail, we apply induction on the number of reflex vertices v of the fence F_{i-1} such that the shortest path to v starting from a ray of R_{i-1} extends to a ray that cuts off a hidden pocket at v.

The basis for the induction is when there are no such hidden pockets. Then $A_{i-1} \subseteq R_{i-1}$, so the rays are disjoint. We must prove that T_i is connected.

T_i consists of the parts of the tree T_{i-1} inside P_i together with some portions of the boundary of P_i. If T_{i-1} lies entirely inside P_i, then T_i is T_{i-1} and we are done. Otherwise, every point of T_i interior to P_i connects to a point of T_i on the boundary of P_i, so it suffices to prove that points of T_i on the boundary of P_i are connected in T_i. Consider two points a and b of T_i on the boundary of P_i. If they are both points of T_{i-1}, consider the path joining them in T_{i-1}. If this path lies inside P_i we are done. Otherwise, by chopping up the path wherever it crosses the boundary of P_i, it suffices to consider the case where the path lies entirely outside P_i. A polygon is formed by the path, together with one chain, β, of ∂P_i between a and b. [Say this better.] We will show that β is contained in T_i; thus a and b are connected in T_i. Consider any point c along β. Consider a shortest path to c starting with a ray of R_i. The last ray of this path cannot reach c from inside P_i, otherwise the extension of the ray would hit T_i. Thus c is is a first contact point, and is in T_i.

We must still consider the case where one point, say a, is not in T_{i-1}. Since a is in T_i, there must be a shortest path to a starting from a ray of R_i, and entering P_i for the first time at a.

[To be filled in.]

This completes the proof that T_i is connected in the basis case of our induction proof.

For the induction step, consider a ray

□

We now discuss what it means for rays to "bounce" at a vertex of T_i.

Claim 1. *Let v be a vertex of T_i, with edges e^1 and e^2 incident to v and consecutive in clockwise order around v. Let R^1 [R^2] be the set of rays leaving points on e^1 [e^2], and entering the region between e^1 and e^2 in clockwise order. Let r^1 and r^2 be the limiting rays of these sets as the source approaches v along the edge. Then the rays that "bounce" (i.e., form locally shortest paths) at v consist of all rays between r^1 and r^2.*

Lemma 6. *If R_{i-1} is a starburst from a tree T_{i-1}, then R_i is a starburst. [We use more than the hypothesis that R_{i-1} is a starburst—in particular we use the fact that the two rays leaving an interior point of an edge of T_{i-1} are reflcetions of each other in that edge.]*

Proof. We already have that no two distinct rays of A_{i-1} intersect. All of these become rays of R_i, though their sources move from points on T_{i-1} and F_i to points on T_i. It remains to show that the various rays that are formed when rays of A_{i-1} "bounce" off points of ∂P_i remain non-intersecting, and that they fill in the "gaps" to reach the whole plane. We address the issue of intersections first, dealing with rays that bounce from interior points of edges of T_i, and then with rays that bounce from vertices of T_i.

Two rays that bounce off edges of P_i cannot intersect each other because P_i has a convex facade.

We next consider the case of one ray bouncing off an edge e of P_i intersecting one ray of A_{i-1}. Suppose that a ray r from a source point t_1 of T_{i-1} bounces at edge e of P_i to form ray b. Suppose, for ease of description, that b bounces to the left of r. Suppose by contradiction that b is intersected by a ray s coming from a point t_2 of T_{i-1} inside P_i. This intersection must occur after s exits P_i. Because s and r do not intersect, t_2 must lie on the left of r. Now t_1 and t_2 are connected by a path, say γ, in T_{i-1}. The path γ must cross into P_i. Let p be the first point of γ (traversed from t_1 to t_2) on ∂P_i. Because γ cannot cross r or s, point p must lie between the points where r and s cross ∂P_i. Let e' be the edge of T_{i-1} containing p. Consider the two rays u and v of R_{i-1} that emanate from point p in R_{i-1}. We will argue that one of these rays must intersect r or s, contradicting the fact that R_{i-1} was a starburst. We crucially use the fact that u and v are reflections of each other in the edge e'. Let r' and b' be rays emanating from p parallel to r and b respectively. The wedge in which u can live extends from r' counterclockwise to the part of e' inside P_i (which comes before e). The reflecting wedge in which v can live extends from e' to a ray that comes before b'. Any possible ray v in this wedge will intersect either the path γ or the ray s. This is a contradiction.

The only rays left to consider for possible intersections are those that "bounce" at vertices of T_i on ∂P_i. These cannot produce an intersection, since, by Claim 1, they simply fill in the gaps between the rays emanating from interior points of edges.

Finally, we must argue that the rays of R_i reach every point in the plane. This follows from Claim 1, together with the fact that T_i is connected (Corollary **??**). □

Corollary 1. *If R_{i-1} is a starburst from a tree T_{i-1}, then T_i is a tree.*

Proof. This follows from Corollary **??** plus the fact that the source of a starburst cannot contain a cycle (otherwise the ray of the starburst to a point inside the cycle would hit the cycle). □

As an immediate consequence, we obtain

Lemma 7. *For any $i \in \{0, \ldots, k\}$ and any $p \in F_i$ there is a unique locally shortest fenced i-path to p. Thus, local optimality is equivalent to global optimality.*

4.2 The Last Step Shortest Path Map

As before, we use a *last step shortest path map*. Actually, we use several such maps—with and without fences. We associate with the starburst R_i a subdivision of the plane, $\mathcal{S}_i^R$, that groups together points reached by rays of R_i that leave from the same vertex of T_i, or leave from the same side of the same edge of T_i. A *pass-through* cell is one in which the rays leaving the source are inside P_i in a very small neighbourhood of the source. In particular, any cell of S_i^R whose source is a vertex or edge of some T_j, $j < i$, is a pass-through cell. For purposes of space efficiency, we group together all of the pass-through cells of $\mathcal{S}_i^R$ into one (possibly disconnected) "pass-through" region. Other cells of $\mathcal{S}_i^R$ have as their source either a vertex or an edge of T_i.

The structure $\mathcal{S}_i^R$ tells us how shortest paths leave P_i ignoring the fence F_i. To take the fence into consideration, we define a subdivision $\mathcal{S}_i^F$ of F_i that groups together points reached by shortest paths in F_i that start from rays of R_i, and whose last edges leave from the same vertex of F_i, or leave from the same vertex of T_i or leave from the same side of the same edge of T_i. Again, we group together all pass-through cells. (To add some intuition, for $P_0 = s$, $\mathcal{S}_0^R$ has a single region containing all rays leaving s, but $\mathcal{S}_0^F$ is a standard shortest path map inside the fence F_0.)

Finally, we define the subdivision of the plane, $\mathcal{S}_i^A$, based on the rays A_i that arrive at P_{i+1}. Here we group together points of the plane reached by rays of A_i that emanate from the same vertex of T_i, or the same vertex of F_i, or from the same side of the same edge of T_i. Again, we group together all pass-through cells.

4.3 The Algorithm

We describe below how to compute the last step shortst path maps, but first we show how to use them to answer queries.

Answering Shortest Path Queries. Given a point $q \in F_i$, we can find a shortest fenced i-path to q by calling **Query**$(q, \mathcal{S}_i^F)$. We answer such a query by locating q in $\mathcal{S}_i^F$, and, based on the results, passing q, or an appropriate reflection of q, to appropriate level-$(i-1)$ data structures. As we recurse, we will need to query $\mathcal{S}_{i-1}^A$ rather than $\mathcal{S}_{i-1}^F$, since our query point will not be guaranteed to lie inside F_{i-1}, but will be guaranteed to be hit by a ray of A_{i-1}. We say that a path *respects* F_i if it lies in F_i except that the last edge may possibly exit F_i after entering P_{i+1}. Our algorithm to construct the shortest path maps will also need to query $\mathcal{S}_i^R$. We thus describe a generic query in any of these structures.

For $X = F$ or A or R, **Query**$(q, \mathcal{S}_i^X)$ locates q in $\mathcal{S}_i^X$, and then follows these cases:

Case 1. q is in a cell whose root is a vertex $v \in F_i$ or a vertex $v \in T_i$. In this case, the last edge of the desired path is the line segment (v, q), and we can find the actual path by using stored information at v (which we will have queried previously).

Case 2. q is in the "pass-through" region. In this case, a shortest fenced $(i-1)$-path to q automatically visits P_i, and the portion of the path from the entry point of P_i to q respects F_i. Thus, we desire a shortest fenced $(i-1)$-path to q. Note that q need not be inside F_{i-1}. However, it is reached by a ray of A_{i-1}, and so we obtain the correct answer by calling **Query**$(q, \mathcal{S}_{i-1}^A)$.

Case 3. q is in a cell whose root is an edge e of T_i. Then the desired path to q bounces off edge e. In this case, let q' be the reflection of q with respect to the line through edge e. A shortest fenced $(i-1)$-path to q' automatically visits P_i, and the final portion of it respects

F_i. Thus, we desire a shortest fenced $(i-1)$-path to q'. The last segment of such a path will cross edge e at a point p, say, and adding the line segment from p to q gives us the final path. Note that q' need not be inside F_{i-1}. However, it is reached by a ray of A_{i-1}, and so we obtain the correct answer by calling **Query**$(q', \mathcal{S}^A_{i-1})$.

Provided we have the required planar subdivisions on hand, and their sizes are polynomial in n, the above algorithm proves:

Lemma 8. *For any point q in F_i we can find the last ray of a shortest fenced i-path to q in time $O(i \log n)$.*

To compute the actual path we need to take into account the length of the path:

Lemma 9. *For any point q in F_i we can find the shortest fenced i-path to q in time $O(i \log n + m) = O(i \log n + f)$, where m is the combinatorial length of the path, and $f = \sum_{j \le i} f_i$ is the total size of all the relevant fences.*

Constructing the Last Step Shortest Path Maps. Assuming we have $\mathcal{S}^F_{i-1}$ and $\mathcal{S}^F_j, \mathcal{S}^A_j$ for $j < i$, we show how to construct $\mathcal{S}^A_{i-1}$ and T_i and $\mathcal{S}^R_i$, and then, from those, how to construct $\mathcal{S}^F_i$.

Constructing $\mathcal{S}^A_{i-1}$, T_i, $\mathcal{S}^R_i$. We compute all potential vertices of T_i that are not vertices of previous T_j's as follows: throw in all vertices of the facade of P_i plus all intersection points of the facade of P_i and the facade of P_j for $j < i$. For each such potential vertex v, we find the last edge of a locally shortest fenced $(i-1)$-path to v by calling **Query**$(v, \mathcal{S}^F_{i-1})$.

On the basis of how the last edge of the path arrives at v, we determine whether v is indeed a vertex of T_i. If it is, the arriving ray becomes part of $\mathcal{S}^A_{i-1}$, and we compute the rays $r^s_i(v)$ and $r^b_i(v)$ to become part of $\mathcal{S}^R_i$. We preprocess $\mathcal{S}^A_{i-1}$ and $\mathcal{S}^R_i$ for point location queries.

Constructing $\mathcal{S}^F_i$. We preprocess F_i for ray shooting queries; this takes $O(|F_i|)$ time, and allows queries in $O(\log n)$ time [17]. We call **Query**$(v, \mathcal{S}^R_i)$, for each reflex vertex v of F_i, to find the last edge l of a shortest fenced i-path to v ignoring the fence F_i. Using a ray shooting query in F_i, we determine if l is contained in F_i. If not, then we ignore v. Otherwise we have found the shortest fenced i-path to v. Let r be the continuation of l beyond v until its first intersection point with ∂F_i. As in the proof of Lemma 4, we can identify the hidden pocket beyond r. We can compute a standard shortest path map for source point v inside this pocket in linear time [17]. Putting together this information for all reflex vertices v of F_i, and combining with the relevant portions of R_i, yields the subdivision $\mathcal{S}^F_i$ of F_i, which we then process for point location queries.

Running time for the algorithm. Let f_i be the size of F_i, and p_i be the size of the facade of P_i in F_i. Let $f = \sum_i f_i$, and $p = \sum_i p_i$. In our general case f and p are $O(n)$.

Let t_i be the number of vertices of T_i that are not vertices of a previous T_j, and let $t = \sum_i t_i$.

Lemma 10. *t is $O(pk)$.*

Proof. Each vertex of T_i that is not a vertex of some previous $T_j, j < i$, is an intersection point of an edge of the convex facade of P_i with an edge of the convex facade of P_j for some $j < i$. Since the facades are convex, the number of intersection points is at most $\sum_{i,j} O(p_i + p_j)$, which attains its maximum value when $p_i = \lceil p/k \rceil$ for each i. Hence, the total number of these vertices is $O(pk)$. □

The complexity of $\mathcal{S}^R_i$ is $O(t_i)$, the complexities of $\mathcal{S}^F_i$ and $\mathcal{S}^A_{i+1}$ are $O(t_i + t_{i+1} + f_i)$. In the general case these are both $O(nk)$ and, by Lemma 8 a query to find the last edge of a shortest fenced i-path $(i \le k)$ to a point q takes $O(k \log n)$ time.

We perform such a query for each vertex of each F_i, and each vertex of each T_i that is not already a vertex of a previous T_j. Thus we do $p + f + t$ queries, and the total time required is $O((p + f + t)k \log n)$. In our general case, this is $O(nk^2 \log n)$.

Theorem 2. *The TPP for arbitrary convex polygons P_i and fences F_i can be solved in time $O(nk^2 \log n)$, using $O(nk)$ space.*

The Watchman Route Problem. For the watchman route problem in polygon P, the polygons P_i are defined by the $k = O(n)$ essential cuts of P. Each P_i may have $O(n)$ vertices, but p_i, the size of the facade of P_i, is just 2. Thus p is $O(n)$. Each fence, F_i, is all of P, so f is $O(n^2)$. Applying Lemma 10 we have that t is $O(nk)$. From our analysis above, our algorithm takes time $O((p + f + t)k \log n)$. Plugging in, we get $O(n^3 \log n)$.

Corollary 2. *The (fixed-source) watchman route in a simple polygon can be solved in time $O(n^3 \log n)$.*

The Safari Problem: For the safari problem, p is $O(n)$. Because the P_i's are disjoint, t_i is just p_i, and t is $O(n)$ as well. Finally, fences can be defined such that f is $O(n)$ ([3]). [Give more detail.] ⇐ Plugging into the running time of $O((p + f + t)k \log n)$ yields $O(nk \log n)$.

Corollary 3. *The safari problem can be solved in time $O(nk \log n)$.*

5 The Full Combinatorial Shortest Path Map

In this section we study the *full combinatorial shortest path map*, $\mathrm{SPM}_i(s)$, which is the decomposition of the plane into regions (*cells*) according to the *combinatorial type* of the shortest i-path from s. The *combinatorial type* of an i-path, π, from s to q is a listing of the edges and vertices of the polygons P_j and F_j that are intersected by π, together with an indication for each edge whether π crosses it or reflects from it.

Associated with each cell σ of $\mathrm{SPM}_i(s)$ is a vertex or line segment, $r(\sigma)$, on the boundary of P_i (in fact, on that part of ∂P_i in the set T_i), which is the locus of all first entrance points, p_i, associated with shortest i-paths from s to points $t \in \sigma$; we refer to $r(\sigma)$ as the *source* for cell σ. If $r(\sigma)$ is a line segment, then it is either a *pass-through source* for σ (if $\pi_i(t)$ passes through $r(\sigma)$, for $t \in \sigma$), or it is a *reflection source* for σ (if $\pi_i(t)$ reflects off the edge of P_i containing $r(\sigma)$, for $t \in \sigma$). If $r(\sigma)$ is a line segment, then it serves as the pass-through source for one cell (say, σ) and a reflection source for another "partner" cell (say, σ').

Theorem 3. *Given a sequence $P_1, \ldots, P_k$ of k disjoint convex polygons, the worst-case complexity of $SPM_k(s)$ is $\Omega((n-k)2^k)$.*

Proof. We first consider the case in which each polygon P_i is a single line segment (so $k = n$). As shown in Figure 6, the number of distinct combinatorial classes of shortest i-paths grows exponentially with i (2^i); thus, $\mathrm{SPM}_k(s)$ has $\Omega(2^k)$ complexity. (Here we are using the property that local optimality implies global optimality.)

In the case that $n > k$, we use the same kind of construction, but we replace the first line segment with a chain, P_1, of $n-k+1$ colinear segments, while $P_2, \ldots, P_k$ remain single line segments. Then, the number of distinct path classes is $\Omega((n-k)2^k)$. □

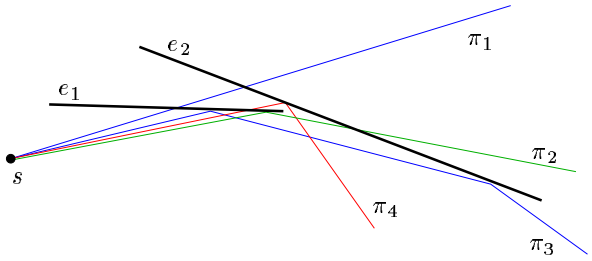

Figure 6: Illustration of the lower bound of Theorem 3: Each of the four paths $\pi_1, \pi_2, \pi_3, \pi_4$ is locally optimal. Path π_1 goes through e_1 and e_2, π_2 reflects on e_1 and goes through e_2, path π_3 reflects on e_1 and on e_2, and path π_4 goes through e_1 and reflects on e_2. By orienting e_1 and e_2 so that they are almost horizontal, all four classes of paths are arbitrarily close to being horizontal. Thus, the construction can be continued, with these four path classes meeting an edge e_3, where each can go through or reflect on e_3, resulting in eight classes of paths, etc.

Remark 1. *Note that we can add a bounding polygon, P, surrounding all of the segments of Figure 6 and touching an endpoint of each segment, while not interfering with the path classes. Thus, a similar lower bound applies to the case of the shortest path map associated with the safari problem.*

Theorem 4. *Given a sequence $P_1, \ldots, P_k$ of k arbitrary disjoint simple polygons, the complexity of $SPM_k(s)$ is $O((n-k)2^k)$.*

Proof. Walk along the boundary of P_i, keeping track of our location in the subdivision $\mathrm{SPM}_{i-1}(s)$. Consider that connected portion, C, of ∂P_i that lies within a cell σ of $\mathrm{SPM}_{i-1}(s)$. We can assume that C separates the interior of P_i from the source, $r = r(\sigma)$; i.e., we can assume that $C \subseteq T_i$. If C is a single line segment (on an edge of P_i), then in the decomposition $\mathrm{SPM}_i(s)$ C will serve as the source for two cells – one pass-through cell and one reflection cell. If C is a chain having one or more vertices on it, then each segment of C serves as a source for two cells of $\mathrm{SPM}_i(s)$ and each vertex of C within σ serves as the source for one cell of $\mathrm{SPM}_i(s)$.

Thus, in total we get that the number, m_i, of cells of $\mathrm{SPM}_i(s)$ is related to the number, m_{i-1}, of cells of $\mathrm{SPM}_{i-1}(s)$ by the relationship

$$m_i \leq 2m_{i-1} + |P_i|,$$

where $|P_i|$ denotes the number of vertices of polygon P_i. This recursion gives the claimed bound. □

In fact, by an incremental method suggested in the proof above, we obtain an output-sensitive algorithm; we defer the proof to the full paper.

Theorem 5. *For any sequence $P_1, \ldots, P_k$ of k simple polygons having a total of n vertices, one can compute $SPM_k(s)$ in time $O(k \cdot |SPM_k(s)|)$, after which a shortest k-path from s to a query point t can be computed in time $O(k + \log n)$.*

6 TPP on Nonconvex Polygons

We begin by observing that, just as shortest paths in the L_1 metric among orthohedral obstacles are readily solved in polynomial time by searching the grid induced by the input vertices, so is this special case of our problem:

Proposition 1. *The TPP in the L_1 metric is polynomially solvable (in $O(n^2)$ time and space) for arbitrary rectilinear polygons P_i and arbitrary fences F_i. The result lifts to any fixed dimension d if the regions P_i and the constraining regions F_i are orthohedral.*

Proof. One can readily argue that there exists an optimal path that lies on the grid induced by the coordinates of the vertices of the P_i's and the F_i's. Then, a simple search for shortest paths on the grid solves the problem. Slightly more efficient methods are known for the corresponding shortest path problem in three or more dimensions; see [14, 15, 13]. □

Next, we state our main lower bound result, which shows that even in the case of the L_1 metric, if the obstacles have edges of three orientations (axis-parallel, plus edges of angle ± 45 degrees) instead of just the two axis-parallel orientations, the problem becomes hard. The proof is based on a careful adaptation of the Canny-Reif [4] proof of NP-hardness of the three-dimensional shortest path problem, using some new gadgets.

Theorem 6. *The touring polygons problem is NP-hard, for any L_p metric ($p \geq 1$), in the case of nonconvex polygons P_i, even in the unconstrained ($F_i = \mathbb{R}^2$) case with obstacles bounded by edges having angles 0, 45, or 90 degrees with respect to the x-axis.*

Proof. (*Sketch*) Our proof is by reduction from 3-SAT. We show only the case of the L_1 metric. We construct five kinds of gadgets: 2-way path splitters (that double the number of shortest path classes), 3-way path splitters (that triple the number of shortest path classes, without changing their ordering), path shufflers (that perform a perfect shuffle of the input path classes), literal filters (which "select" paths that have a particular bit equal to 0 or 1), and clause filters (that ensure that each clause is true in a satisfying truth assignment).

Given an instance of 3-SAT, having n variables $b_1, \ldots, b_n$ and m clauses $C_i = (l_{i1} \vee l_{i2} \vee l_{i3})$, we construct a sequence of polygons, $(P_1, \ldots, P_{O(n+m)})$, along with points s and t, such that solving the TPP on this instance permits us to determine if the 3-SAT formula $\wedge_i C_i$ has a satisfying truth assignment. (Our polygons P_i take the form of line segments and unions of line segments whose angle with the x-axis is in $\{0, \pm 45, 90\}$.)

We use n 2-way splitters to create 2^n distinct path classes that form a "bundle" of parallel paths. Each path class encodes a truth assignment for the n variables. We use a sequence of clause filters, each consisting of three literal filters sandwiched between two 3-way splitters, to filter out those path classes whose corresponding variable assignments fail to satisfy each clause. The property of the literal filter is that only those path classes that have a particular bit set to 0 or 1 are able to pass through the filter without having a detour (a requirement to go through a horizontal segment) force the path to be longer than the others.

While our proof attempts to follow that of [4], there are important new aspects to our proof. First, we must be careful to make the layout in the plane, so we must modify the clause filter of [4], which exploited the third dimension to have the path classes pass through three literal filters in parallel. This modification utilizes the new 3-way splitter.

Also, we must "link" the n stages of the literal filters that make up a clause filter, since all path classes must pass through the *same* sequence of polygons. This means that we construct polygons with bridging segments between individual shuffle gadgets, so that one polygon may consist of up to three shuffle gadgets; see Figure 9. (There are also linkages between the "blocker" segments that select the most significant bit in a literal filter and the other shuffle gadgets or blockers at the same stage of a clause filter. See the dashed segments representing such linkages in Figure 10(right).)

The final step collects all the path classes back into a single path class that terminates at t. For this we use n inverted (2-way) splitters.

Figure 7: Left: 2-way splitter gadget. Right: 3-way splitter gadget.

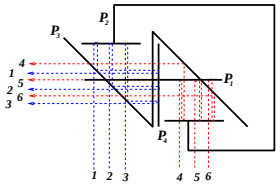

Figure 8: Shuffle gadget.

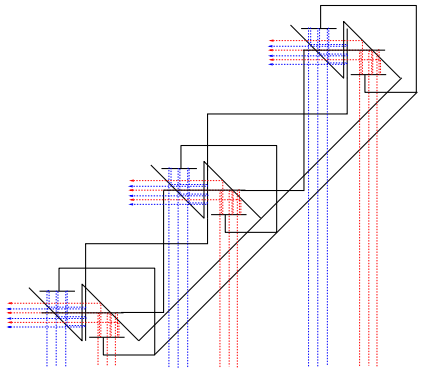

Figure 9: "Ganging" three shuffle gadgets in parallel.

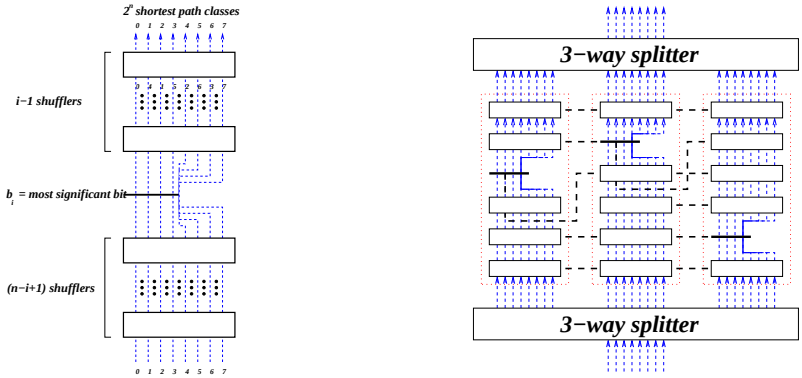

Figure 10: Left: Literal filter. Right: Clause filter.

We note that our construction utilizes polygons that *do* intersect; it is open whether or not the TPP remains NP-hard in the case that the polygons P_i are pairwise disjoint. □

The hardness shows that the general TPP is just as hard as the three-dimensional shortest path problem among obstacles. It is also "just as easy", in that known techniques to obtain an approximately shortest path among obstacles in three dimensions (e.g., [1, 10, 11, 12, 22]) apply to the TPP if we model it using "sheets of paper" that are sufficiently large and close together:

Proposition 2. *There is a fully polynomial time approximation scheme ($O(n^4/\epsilon^2)$) for the general version of the TPP, even if the constraining fences are allowed to be multiply connected.*

Proof. As described in the introduction, any instance of the unconstrained TPP can be modeled as a shortest path problem among "sheets of paper" with polygonal holes to represent the regions P_i. (The sheets of paper have a tiny separation of ϵ between any pair of consecutive sheets.) We can model the constraints imposed by fences similarly by constructing walls in the shape of the fence F_i, linking the ith sheet to the $i+1$st sheet. Then, we can simply apply known methods [1, 10, 11, 12, 22] for approximating shortest paths among obstacles in three dimensions to get the claimed theorem. □

7 Open Problems

Many of the directions in which it would be desirable to extend this work run into the barrier that local optimality no longer guarantees global optimality. Thus one really needs to compute distances, something that we avoided entirely in our algorithms for the TPP on convex polygons. One of the most intriguing open problems suggested by our results is to determine the complexity of the TPP for *disjoint* nonconvex simple polygons.

We conclude with some open problems of current interest to us:

1. Can our solution of the watchman route problem be improved and can one give any nontrivial lower bounds for the problem?
2. Our construction for the NP-hardness of the TPP (Theorem 6) utilizes polygons that intersect; it is open whether or not the TPP remains NP-hard in the case that the (non-convex) polygons P_i are pairwise disjoint.
3. What can be said about the case in which the fence regions F_i are multiply connected?

8 Acknowledgments

A. Efrat and J. Mitchell are partially supported by Honda Research Labs. J. Mitchell acknowledges support also from NASA Ames Research (NAG2-1325), NSF (CCR-0098172), and the U.S.-Israel Binational Science Foundation. A. Lubiw's research is supported by the Natural Sciences and Engineering Research Council of Canada.

References

[1] T. Asano, D. Kirkpatrick and C. Yap. Pseudo approximation algorithms, with applications to optimal motion planning. In *Proc. 18th Annu. ACM Sympos. Comput. Geom.*, 2002, 170-178.

[2] H. Baumgarten, H. Jung and K. Mehlhorn. Dynamic point location in general subdivisions. *J. Algorithms*, 17:342–380, 1994.

[3] S. Bespamyatnikh. An $O(n \log n)$ algorithm for the zoo-keeper's problem. *Computational Geometry: Theory and Applications*, 24:63–74, 2002.

[4] J. Canny and J. H. Reif. New lower bound techniques for robot motion planning problems. In *Proc. 28th Annu. IEEE Sympos. Found. Comput. Sci.*, 1987, 49–60.

[5] B. Chazelle, H. Edelsbrunner, M. Grigni, L. J. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12:54–68, 1994.

[6] B. Chazelle and L. J. Guibas. Visibility and intersection problems in plane geometry. *Discrete Comput. Geom.*, 4:551–581, 1989.

[7] W.-P. Chin and S. Ntafos. Shortest watchman routes in simple polygons. *Discrete Comput. Geom.*, 6:9–31, 1991.

[8] W.-P. Chin and S. Ntafos. The zookeeper route problem. *Information Sciences*, 63:245–259, 1992.

[9] W. Chin and S. Ntafos. Optimum watchman routes. *Information Processing Letters*, 28:39–44, 1998.

[10] J. Choi, J. Sellen, and C. K. Yap. Approximate Euclidean shortest paths in 3-space. *Internat. J. Comput. Geom. Appl.* , 7:271–295, 1997.

[11] J. Choi, J. Sellen, and C. K. Yap. Precision-sensitive Euclidean shortest path in 3-space. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, 1995, 350–359.

[12] K. L. Clarkson. Approximation algorithms for shortest path motion planning. In *Proc. 19th Annu. ACM Sympos. Theory Comput.*, 1987, 56–65.

[13] K. L. Clarkson, S. Kapoor, and P. M. Vaidya. Rectilinear shortest paths through polygonal obstacles in $O(n(\log n)^2)$ time. In *Proc. 3rd Annu. ACM Sympos. Comput. Geom.*, 1987, 251–257.

[14] M. de Berg, M. van Kreveld, and B. J. Nilsson. Shortest path queries in rectilinear worlds of higher dimension. In *Proc. 7th Annu. ACM Sympos. Comput. Geom.*, 1991, 51–60.

[15] M. de Berg, M. van Kreveld, B. J. Nilsson, and M. H. Overmars. Shortest path queries in rectilinear worlds. *Internat. J. Comput. Geom. Appl.* , 2:287–309, 1992.

[16] M. Dror. Polygon plate-cutting with a given order. *IIE Transactions*, 31:271–274, 1999.

[17] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

[18] J. Hershberger and J. Snoeyink. An efficient solution to the zookeeper's problem. In *Proc. 6th Canadian Conf. on Comp. Geometry*, 1994, 104–109.

[19] J. Hershberger and S. Suri. A pedestrian approach to ray shooting: shoot a ray, take a walk. *J. Algorithms*, 18:403–431, 1995.

[20] J. S. B. Mitchell, Geometric shortest paths and network optimization, In *Handbook of Computational Geometry* (J.-R. Sack and J. Urrutia, eds.), pages 633–701, Elsevier Science, 2000.

[21] S. Ntafos. Watchman routes under limited visibility. *Comput. Geom. Theory Appl.*, 1:149–170, 1992.

[22] C. H. Papadimitriou. An algorithm for shortest-path motion in three dimensions. *Inform. Process. Lett.*, 20:259–263, 1985.

[23] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, NY, 1985.

[24] X. Tan and T. Hirata. Shortest safari routes in simple polygons. *Lecture Notes in Computer Science*, Vol. 834, 1994, 523–531.

[25] X. Tan, T. Hirata, and Y. Inagaki. Corrigendum to "An Incremental Algorithm for Constructing Shortest Watchman Routes". *International J. of Comp. Geom. and App.* , 9:319–323, 1999.

[26] X. Tan. Fast computation of shortest watchman routes in simple polygons. *Information Processing Letters*, 77:27–33, 2001.

[27] X. Tan. Shortest zookeeper's routes in simple polygons. *Information Processing Letters*, 77:23–26, 2001.

[28] X. Tan and T. Hirata. Finding shortest safari routes in simple polygons. Manuscript (submitted), Tokai University, 2001.