Thick Non-Crossing Paths and Minimum-Cost Flows in Polygonal Domains

Joseph S. B. Mitchell
Applied Math and Statistics
Stony Brook University
jsbm@ams.stonybrook.edu

Valentin Polishchuk
Helsinki Institute for Information Technology
valentin@compgeom.com

ABSTRACT

We study the problem of finding shortest non-crossing thick paths in a polygonal domain, where a *thick* path is the Minkowski sum of a usual (zero-thickness, or *thin*) path and a disk. Given K pairs of terminals on the boundary of a simple n-gon, we compute in $O(n+K)$ time a representation of the set of K shortest non-crossing thick paths joining the terminal pairs; using the representation, any particular path can be output in time proportional to its complexity.

We compute K shortest thick non-crossing paths in a polygon with h holes in $O\left((K+1)^h h!\,\mathrm{poly}(n,K)\right)$ time, using an efficient method to compute any one of the K thick paths if the "threadings" of all paths amidst the holes are specified. We show that if h is not constant, the problem is NP-hard; we also show the hardness of approximation. We give a pseudopolynomial-time algorithm for some rectilinear versions of the problem.

We apply our thick paths algorithms to obtain the first algorithmic results for the *minimum-cost continuous flow* problem — an extension of the standard discrete minimum-cost network flow problem to continuous domains. The results are based on showing a continuous analog of the Network Flow Decomposition Theorem.

Categories and Subject Descriptors: F.2.2 [Nonnumerical Algorithms and Problems]: Geometrical problems and computations; Routing and layout
General Terms: Algorithms, Theory
Keywords: Shortest paths; Data structures; Optimal Flows

1. Introduction

One of the most studied subjects in computational geometry is the shortest path problem [25]: given a set of obstacles and a pair of points (s,t), find a shortest s-t path avoiding the obstacles. The *non-crossing paths* problem is an extension of the shortest path problem: given a set of obstacles and K pairs of points (s_k,t_k), find a collection of K non-crossing s_k-t_k paths such that the paths are optimal according to some criterion. The objective may be either to minimize the sum of the lengths of the paths (*minsum* version) or to minimize the length of the longest path (*minmax* version). The general problem (with K being part of the input) is NP-hard even in the absence of obstacles and under any objective — minsum or minmax [3]. On the contrary, in a *simple* polygon with all pairs (s_k,t_k) on the boundary, it is possible to build in linear time a data structure such that a shortest path can be output in time proportional to its complexity [28].

"Thick" shortest path problem arises in a variety of applications, including VLSI, air traffic management (ATM), robotics. A *thick* path is the Minkowski sum of a curve and the unit disk. The problem of finding *multiple* thick paths (the *Thick Non-Crossing Paths Problem*), which we consider in this paper, is an extension of both the shortest non-crossing paths and the thick shortest path problems.

Related Work Our problem can be viewed as a variation of the Fat-Edge Graph Drawing problem [10], which, in turn, is an extension of the Continuous Homotopic Routing problem [8, 12, 21, 22]. A related problem is that of finding shortest paths homotopic to a given collection of paths [4, 11]. The novelty of our work lies in considering the problem in simple polygons and polygonal domains; the previous research concentrated on *point* obstacles for the paths. Our data structure for storing thick shortest paths shares similar ideas with the ones used in [10, 22, 28].

Some heuristics for finding thick non-crossing paths in polygonal domains are suggested in the VLSI literature [17], but no complexity analysis nor performance guarantees are given there. A very restricted version is considered in [2]. In a rectilinear environment, fast algorithms are known for some special cases of the minsum version [19, 30].

Flows in the Continuum

In certain applications the task is to route a large "swarm" of small objects (agents) through a polygonal environment. Finding and describing a specific path for each individual object may be an unnecessary complication. The problem then is to find an optimal *flow* (a vector field) in the domain so that any object could make its way through the domain simply by following one of the flow streamlines.

Related Work A variety of classical results exists for *discrete* network flows. The Maxflow-Mincut Theorem asserts that the value of the maxflow in a network equals the capacity of the mincut. The Flow Decomposition Theorem (FDT) states that a mincost flow can be decomposed into a set of paths. Establishing the continuous versions of the Maxflow-Mincut [24, 29] and the FDT (proven here) is fundamental to exploring the mapping between discrete graph notions and their continuous analogues in geometry.

SCG'07, June 6–8, 2007, Gyeongju, South Korea.

Motivation

Our particular motivation for the problem comes from ATM applications in routing safe lanes ("flows") of air traffic while avoiding hazardous weather systems. Each lane is a thick path, determined by the *protected airspace zone* that specifies the horizontal separation standard for flights. The thick non-crossing paths problem arises in optimizing the set of lanes; it is a min-cost flow problem in the continuum, as we prove in our Flow Decomposition Theorem.

Another motivation comes from data transportation in sensor networks. Suppose the data, produced at a set of sources, has to be delivered to a set of destinations by propagating through a set of sensors evenly distributed in a domain. Connecting the source-destination pairs with shortest possible paths may create undesirable congestion. One way to avoid the congestion is to introduce a capacity bound on the amount of information passing through any point of the domain. Then, a collection of optimal *well-separated* source-destination paths may be sought that provides more balanced utilization of the domain while keeping the information tracks short.

Our Contributions

• Section 3: In $O(n+K)$ time we compute a (linear-space) representation of the set of K all-shortest non-crossing thick paths in a simple n-gon for a given set of K terminal pairs. The representation allows us to output the shortest thick path joining a given pair of terminals, in time proportional to the path's complexity.

• Section 4: We give an $O\left((K+1)^h h!\,\mathrm{poly}(n,K)\right)$ algorithm for finding thick non-crossing paths in a polygonal domain with h holes. We show that if h is not constant, the minmax version of the problem is NP-hard (weakly if $K=2$, strongly for large K). We also show that unless P=NP there exists no FPTAS, and give pseudopolynomial-time algorithms for rectilinear versions of the problem.

• Section 5: We state and prove the Flow Decomposition Theorem for minimum-cost flows in the continuum. We define the *Geometric Balanced Transshipment Problem with Source-Sink Separation*; we apply our theorem to solve the problem.

Our algorithms can be used to find optimal paths under any objective that is a non-decreasing function of the length of an individual thick path.

2. Preliminaries

Let P be a simple polygon with n vertices — a simply-connected open subset of the plane whose boundary ∂P consists of n segments disjoint other than at endpoints. In this paper we will also use "polygon" to refer to a set whose boundary consists of line segments and *circular arcs*; the complexity of such a polygon is the number of its boundary segments and arcs. For any set $S \subset \mathbb{R}^2$ we speak of in the sequel, by ∂S we will mean the part of the boundary of S lying inside P.

A *path* π is a simple curve; $|\pi|$ denotes its length. For $r>0$ let $\mathcal{C}_r$ be the open disk of radius r centered at the origin; we denote $\mathcal{C}_1$ with just $\mathcal{C}$. For a set $S \subset \mathbb{R}^2$ let $(S)^r$ be the Minkowski sum $S \oplus \mathcal{C}_r$. We define a *thick path* Π within P *with reference path* π to be the Minkowski sum $\Pi = (\pi)^1$ such that Π does not intersect the exterior of P. The *length* of a thick path $\Pi = (\pi)^1$ is the length of π.

For two points v, u on the boundary of a simple polygon Q let $Q(v,u)$ denote the part of the boundary of Q from v to u clockwise. Let π be a u-v path within Q, let $B(\pi)$ be the part of Q to the right of the closed curve v–$Q(v,u)$–u–π–v. The points in $B(\pi)$ (resp. in $Q \setminus B(\pi)$) are said to be *below* (resp. *above*) π. If π is the *shortest* u-v path, $B(\pi)$ is called the *slice* of (v,u) and is denoted $sl(v,u)$.

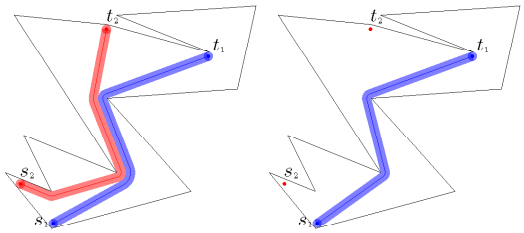

Figure 1: Left: an example. Right: a selfish thick s_1-t_1 path leaves no space for a thick s_2-t_2 path.

Let $P^1 = P \setminus (\partial P)^1$ be P offset by 1 inside. We assume that P^1 is still a simple polygon; otherwise the connected components of P^1 can be treated separately. If $\Pi = (\pi)^1$ is a thick path within P, then necessarily π is a path within P^1.

Problem Formulation Let $\mathcal{ST} = \{(s_k, t_k), k = 1...K\}$ be K pairs of points on the boundary of P^1. Borrowing terminology from the VLSI community, we call the points in $\mathcal{ST}$ *terminals*. Let π_k be an s_k-t_k path within P^1; we call s_k the *start* and t_k the *destination* of the k^{th} path. Let Π_k be the thick path within P with π_k as the reference path, $\Pi_k = (\pi_k)^1$. Thick paths $\Pi_1, \ldots, \Pi_K$ are called *non-crossing* if $\Pi_i \cap \Pi_j = \emptyset$ for $i, j = 1 \ldots K, i \neq j$. Note that we allow the thick paths to share parts of the boundary with each other; we only require that the *interiors* of the paths are disjoint.

We seek a collection of thick non-crossing paths that is *simultaneously* optimal both for the minsum and the minmax version: we require that for any $k = 1...K$ the s_k-t_k path in the collection is as short as possible given the existence of (arbitrary) paths connecting the other terminals, (s_i,t_i), $i = 1, ..., k-1, k+1, ...K$. We call the collection of such paths *all-shortest*[1]. See Fig. 1, left, for an example.

Following [28], we make, WLOG, the following *Positioning* assumption: Starting from s_1 and going around ∂P^1 clockwise one encounters $s_1, s_2, ..., s_K$ in this order, and for any k, s_k appears before t_k. We also assume that the problem instance is feasible, i.e., that the polygon is "wide" enough to accommodate the thick paths.

The three basic problems that arise when dealing with multiple short paths are as follows. (a) *Report* π_k: route one of the K thick shortest paths. In this setting we do not care how the other paths "look like". We only need the exact description of the s_k-t_k shortest path. (b) *Store* K: build a data structure holding all K thick shortest paths. The data structure must support efficient reporting of any single path. (c) *Report* K: Output all K paths. Of course, (c) can be solved either by solving (a) for all $k = 1...K$, or by solving (b) and then querying the data structure. Thus, in the sequel, we concentrate on solving (a) and (b).

Thin Paths We start by recollecting and extending known results on finding K non-crossing shortest *thin* paths [28].

[1]To uniquely define the solution to the minmax version we require that each path in the collection is *locally optimal*.

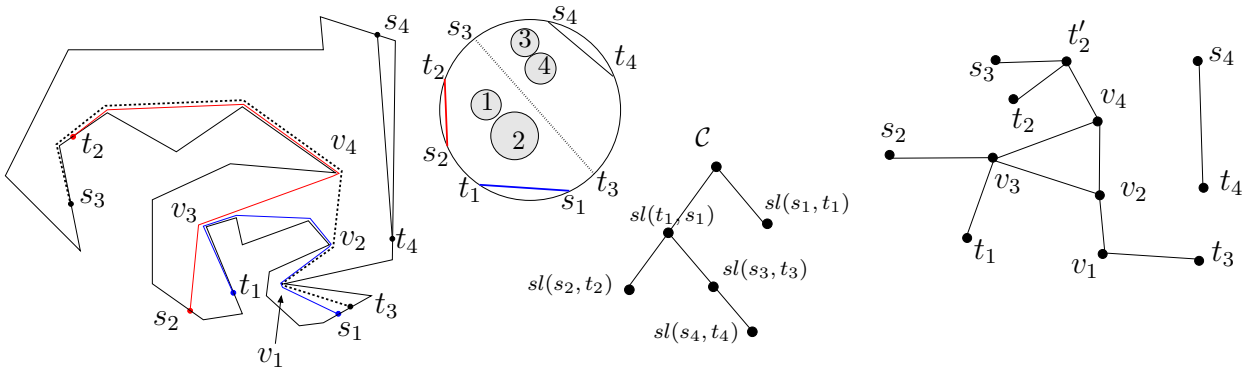

Figure 2: From left to right: an instance of the problem; the mapping of ∂P and the terminals to the unit circle; the tree of slices $\mathcal{T}_{sl}$; the data structure $\mathcal{G}$ storing the paths. The vertices $v_1 ... v_4$ and t_2' are the internal nodes of $\mathcal{G}$; t_2' is a dummy node, added to keep t_2 a leaf.

First, the boundary of the polygon is mapped to the unit circle $\partial\mathcal{C}$; the terminals are identified with their images. Then, a chord $s_k t_k$ is drawn between the terminals in every pair $(s_k, t_k), k = 1...K$. If two of the chords cross, then the problem instance is infeasible. Otherwise, the *tree of slices* $\mathcal{T}_{sl}$ on $\mathcal{C} \cup sl(t_1, s_1) \cup_{k=1}^{K} sl(s_k, t_k)$ is built in which the root is the whole circle $\mathcal{C}$, the root's immediate children are $sl(s_1, t_1)$ and $sl(t_1, s_1)$, and the parent-child relation is defined by containment of the slices (Fig. 2, ignore the shaded disks 1–4 now; see also [28] for details).

According to [28], the collection of the shortest paths between the pairs in $\mathcal{ST}$ forms a *forest*. In fact, the collection may not necessarily form a forest, since there may exist cycles of edges of the paths (Fig. 2); the term "forest" should be replaced with the term "graph"[2] [27]. The size of the "forest" of [28] is $O(n + K)$, and any shortest path can be reported in time proportional to the number of edges in the path. The forest is computed in a bottom-up fashion using $\mathcal{T}_{sl}$, starting in phase 1 with the paths at the leaves of the tree, and in the phase q considering the paths at level $Z - q + 1$, where Z is the height of $\mathcal{T}_{sl}$. In order to achieve linear time, [28] conducts a careful refinement of the funnel paradigm for computing shortest paths in a simple polygon [13]: When a shortest s_k-t_k path at a level q is routed, the funnel from s_k in the direction of t_k is extended until the funnel hits $\mathcal{Q}$, a connected subset of the part of the boundary of the polygon, which has been already used by paths from levels $1, ..., q - 1$. The funnel then continues to extend in the direction of t_k from the other end of $\mathcal{Q}$. This leads to

THEOREM 2.1. *[27, 28] In linear time a data structure can be built such that the shortest path between a pair of terminals can be output in time proportional to its complexity.*

Our solution for the case of thick paths in simple polygons builds and extends on ideas from [28].

3. Thick Paths in Simple Polygons

We first describe the solution for the case $K = 2$; the solution for arbitrary K is derived from it. We assume, WLOG, that s_1, s_2, t_2, t_1 appear in this order clockwise around ∂P^1. Define the *bottom* B (resp. *top* T) of P^1 to be the portion

[2]In Section 3 we build the corresponding graph for the thick paths and augment it with additional information so that it has the required features of the forest of [28]: linear size, while supporting the efficient reporting of a path.

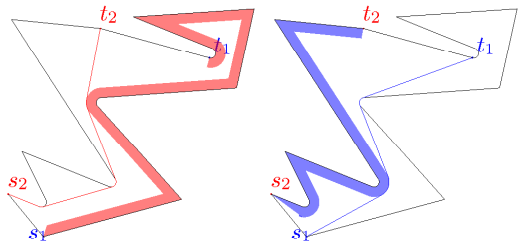

Figure 3: $B = P^1(t_1, s_1)$, $\pi_2 \cap (B)^2 = \emptyset$. $T = P^1(s_2, t_2)$, $\pi_1 \cap (T)^2 = \emptyset$. Otherwise, the other path cannot by routed.

of the boundary of P^1 between t_1 and s_1 (resp. s_2 and t_2): $B = P^1(t_1, s_1), T = P^1(s_2, t_2)$ (Fig. 3).

LEMMA 3.1. $\pi_1 \cap (T)^2 = \emptyset$. $\pi_2 \cap (B)^2 = \emptyset$.

PROOF. *(Sketch.)* Otherwise the other path cannot be routed. □

Let π_1^* (resp. π_2^*) be the shortest s_1-t_1 (resp. s_2-t_2) path within P^1 routed treating $(T)^2$ (resp. $(B)^2$) as obstacles; let $\Pi_1^* = (\pi_1^*)^1$, $\Pi_2^* = (\pi_2^*)^1$. By Lemma 3.1, the very existence of thick s_1-t_1 and s_2-t_2 paths does not allow any s_1-t_1 path to enter $(T)^2$ and any s_2-t_2 path to enter $(B)^2$. Thus, each of Π_1^*, Π_2^* is as short as possible given the existence of the other thick path. We show that Π_1^* and Π_2^* are, in fact, *disjoint*, and thus, provide the solution to the thick non-crossing all-shortest paths problem.

Enclose π_1^* in an (open) "tube" τ of width 2: $\tau = (\pi_1^*)^2$. By the boundary, $\partial\tau$, of τ we will understand the boundary points of τ that lie above π_1^*. (Observe, that $\partial\tau$ may be disconnected since τ may run outside P^1.) We need to prove the following (the details of the proof are deferred to the full paper [26]):

LEMMA 3.2. $\pi_2^* \cap \tau = \emptyset$.

PROOF. *(Sketch)* We consider the parts of τ induced by different parts of π_1^* one by one. We show that π_2^* "enters" and "exits" τ the same number of times. We prove that we can replace the subpath of π_2^* between the first entry and exit points by the part of $\partial\tau$ and that the new path is shorter (the new path will still be feasible since it goes by the *boundary* of τ). □

THEOREM 3.3. *π_1^* and π_2^* can be found in linear time.*

PROOF. $(B)^2$ and $(T)^2$, since they are an offset of P^1 by 2, are obtained by taking Minkowski sum of portions of P's boundary (simple chains) with disk of radius 3. Thus, $(B)^2$ and $(T)^2$ are found in linear time by computing the medial axis of a simple chain (part of ∂P) [7]. We then compute π_1^*, π_2^* as shortest paths within the resulting free space splinegons, using the linear-time algorithm of [23]. □

The Routing Paradigm The following are corollaries to Lemma 3.2:

COROLLARY 3.4. *The paths Π_1^* and Π_2^* are all-shortest.*

COROLLARY 3.5. *π_1^* (resp. π_2^*) is the shortest s_1-t_1 (resp. s_2-t_2) path within P^1, routed treating $(\pi_2^*)^2$ (resp. $\tau = (\pi_1^*)^2$) as obstacle.*

COROLLARY 3.6. *Let $\pi^* = (\partial\Pi_1^*) \cap (\partial\Pi_2^*)$ be the points, boundary to both Π_1^* and Π_2^*. Then π^* is a path.*

COROLLARY 3.7. *Let s and t be the endpoints of π^*. Then, if the distance from each of s, t to ∂P is at least 2, π^* is the* shortest *s-t path routed treating $(\partial P)^2$ as obstacle.*

The corollaries assert that if Π_1^* and Π_2^* ever "meet" (so that there exist points, boundary to both), they then "go together" for some time, but after that, diverging on their way to the destinations, never meet again. While the paths go together, they do it, of course, in the optimal way.

Loosely speaking, each of the paths is routed "greedily", as opposed to "selfishly". A selfishly routed path would only care about its own length, and would "rush" to the destination in the quickest way, thus, possibly, making the routing of the other path infeasible (see Fig. 1). Our *greedy* routing assumes that each path leaves *just enough* space for the other path to "squeeze in". On the other hand, the obstacles for a path are created in a "conservative" way: no obstacle is larger than it is necessary for the existence of the other path. Thus, the paths routed greedily amidst the conservative obstacles are all-shortest.

General Case: Arbitrary K We give $O(n+K)$ time solutions to Problems (a) *Report* π_k and (b) *Store* K. This gives a way to solve Problem (c) *Report* K in $O(K(n+K))$ time, which (as we show) is worst-case optimal.

As with thin paths (Section 2), we begin by mapping ∂P^1 to $\partial\mathcal{C}$ and drawing the chords $s_k t_k$, $k = 1...K$ (Fig. 2). Let $\mathcal{ST}^{\text{ord}} = (\nu_1, ..., \nu_{2K})$ be the set $\{s_1, ..., s_K, t_1, ..., t_K\}$ ordered clockwise around ∂P^1.

DEFINITION 3.8. *Let v, u be two consecutive points in $\mathcal{ST}^{\text{ord}}$. Let γ be a path within $\mathcal{C}$ from a point on $\partial\mathcal{C}(v,u)$ to a point on the chord $s_k t_k$. The k^{th}* depth *of $P^1(v,u)$, denoted $d_k(v,u)$, is defined as the minimum, over all γ, of the number of (other) chords that γ crosses.*

For example, in Fig. 2 the 4^{th} depth of $P^1(s_2, t_2)$ is 2, the 1^{st} depth of $P^1(t_4, t_3)$ is 1.

Let $\mathcal{O}_k$ be the set of obstacles, obtained by inflating each part of ∂P^1 by 2 times its k^{th} depth (arithmetic modulo $2K$ is assumed in the indices): $\mathcal{O}_k =$

$$\bigcup_{j=1}^{2K} \left(P^1(\nu_j, \nu_{j+1})\right)^{2d_k(\nu_j,\nu_{j+1})} = \bigcup_{\text{edges } e\in\partial P} (e)^{d'_k(e)} \quad (1)$$

where for an edge e of ∂P, $d'_k(e)$ denotes the amount by which e is inflated.

LEMMA 3.9. *The sets in the right-hand side of (1) are pseudodiscs.*

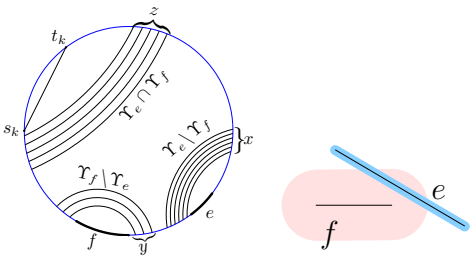

Figure 4: Right: The distance between e and f must be at least $2x+2y$. Left: The distance between e and f is less than $|d'_k(e) - d'_k(f)| = |2x - 2y|$.

PROOF. Recall that we assume that the input problem instance is feasible. Suppose that two sets in (1), say $(e)^{d'_k(e)}$ and $(f)^{d'_k(f)}$, intersect. Then e and f either both lie below π_k or both lie above, since, otherwise, the problem is infeasible. Let $\Upsilon_e \subset \mathcal{ST}$ (resp. $\Upsilon_f \subset \mathcal{ST}$) be the pairs of terminals that contribute to $d'_k(e)$ (resp. $d'_k(f)$). (In terms of the mapping of ∂P^1 onto $\partial\mathcal{C}$, Υ_e contains the chords crossed by any path from (the image of) $(e)^1$ to the chord $s_k t_k$.) Let $|\Upsilon_e \cap \Upsilon_f| = z$, $|\Upsilon_e| = x + z$, $|\Upsilon_f| = y + z$, (Fig. 4, top); then, for the instance to be feasible, the distance between e and f, must be at least $2x + 2y$.

On the other hand, by our construction, $d'_k(e) = 2(x+z)+1$, $d'_k(f) = 2(y+z)+1$. If $(e)^{d'_k(e)}$ and $(f)^{d'_k(f)}$ violate the pseudodisc property, then the distance between segment e and segment f is strictly less than $|d'_k(e) - d'_k(f)| = |2x-2y|$ (see Fig. 4, bottom); since $|2x - 2y| \le \max\{2x, 2y\} \le 2x + 2y$, we have a contradiction. □

If π_k^*, $k = 1...K$, is the path routed amidst $\mathcal{O}_k$ as obstacles, then each path from $\{\Pi_1^*, ..., \Pi_K^*\} = \{(\pi_1^*)^1, ..., (\pi_K^*)^1\}$ is as short as possible given the existence of the others. Moreover, the paths $\Pi_1^*, ..., \Pi_K^*$ are non-crossing, by the same argument as in the proof of Lemma 3.2. Thus,

THEOREM 3.10. *One of the K shortest thick non-crossing paths in a simple polygon can be found in $O(n+K)$ time.*

PROOF. Finding the k^{th} depths of the intervals of ∂P^1 can be done in $O(K)$ time.

Let n_j, $j = 1...2K$ be the complexity of $P^1(\nu_j, \nu_{j+1})$; $\sum n_j = O(n+K)$. Since, by Lemma 3.9, (1) is a collection of pseudodiscs, the complexity of $\mathcal{O}_k$ is $O(\sum n_j) = O(n+K)$ [1, 18], and $\mathcal{O}_k$ can be found in $O(\sum n_j) = O(n+K)$ time by adapting the algorithm for computing the medial axis of a simple polygon in linear time [7]. The free space for π_k^* is a splinegon and, thus, routing π_k^* amidst $\mathcal{O}_k$ can be done in linear time [23]. □

Local optimality A thick path $\Pi = (\pi)^1$ is called *locally optimal* if it cannot be shortened while staying feasible; local optimality means, in particular, that the reference path π is a sequence of line segments and circular arcs with the segments bi-tangent to the arcs, and that at any point, the curvature of π is at most 1. The discussion above shows that each path in a collection of shortest thick paths is locally optimal (this property will be used in Section 5):

LEMMA 3.11. *$\forall\ k = 1...K$, Π_k^* is locally optimal.*

Implicit Paths Representation

The algorithm above can be run for each $k = 1...K$ to get the K shortest paths in $O(K(n+K))$ time. We remark that the

$O(n+K)$ space requirement of our algorithm is the *working space* requirement. The complexity of the k^{th} path may be as high as $\Omega(n+k)$, in which case the size of the output may be $\Omega(K(n+K))$, and $\Omega(K(n+K))$ *output space* may be needed just to store the paths. On the other hand, this shows that our algorithm is worst-case optimal if we require that the paths are output using explicit encoding: each path is given as the sequence of line segments and circular arcs.

The Data Structure Alternatively, we can store the thick shortest paths in a data structure, of size $O(n+K)$, such that any path can be output in time proportional to its complexity. Let $\mathbb{P} = \bigcup_{k=1}^{K} cl(\Pi_k)$ be the set of points that belong to (the closure of) a thick shortest path; $cl(S)$ denotes the closure of a set $S \subset \mathbb{R}^2$. Our data structure $\mathcal{G}$ is then a graph, each connected component of which corresponds to a connected component of $\mathbb{P}$ (Fig. 2).

We first describe what the *edges* of $\mathcal{G}$ are. The *nodes* of $\mathcal{G}$ correspond to those vertices of P that belong to more than one edge of $\mathcal{G}$. We use the term *nodes* for the vertices of $\mathcal{G}$ saving the term *vertices* for the vertices of P.

The reference path of any thick shortest path can be described by a sequence of vertices of P by which it goes (i.e., the vertices that cause the path to bend); the curvature (radius of the obstacle) at each vertex must also be specified. Then, to output the path in time proportional to its complexity, at each vertex a geometric primitive of computing a bi-tangent (or tangent, at the endpoints of the path) must be executed.

Let $\mathbb{S} = (v_{k_1}, ..., v_{k_r})$ be a sequence of vertices of P, consecutive along a path π_k. Suppose some other path(s) also have $\mathbb{S}$ as a subsequence of consecutive vertices. The edges of $\mathcal{G}$ are *maximal* such shared sequences of vertices; with each edge its sequence is stored. The *thickness* of an edge is the number of paths sharing it.

A node of $\mathcal{G}$ corresponds to a vertex of P at which $\mathbb{P}$ "branches", i.e., a vertex that belongs to more than one edge of $\mathcal{G}$. The leaves of $\mathcal{G}$ are the terminals $(s_k, t_k), k \in \{1...K\}$. If $\mathbb{P}$ branches at a terminal, a dummy node can be added to $\mathcal{G}$ so that the terminal and the branch point are two different nodes: the branch point is an internal node, the terminal is a leaf (like t_2 in Fig. 2).

Each internal node of $\mathcal{G}$ stores the list of incident edges sorted angularly clockwise starting with the edge closest to the boundary of P. Let $(e_1, ..., e_L)$ be the list of edges incident to a node u of $\mathcal{G}$. For $1 \le l \le L$, the total thickness of $e_1 \dots e_{l-1}$ (and that of $e_{l+1} \dots e_L$) is also stored at u, along with the edge e_l. Later, when actual routing of a path through u is performed, these thicknesses will give the curvature of the reference path at u.

Let $\Pi_k = (\pi_k)^1$ be a thick path following an edge e of $\mathcal{G}$. All the vertices in e are shared by all the paths going through it. Hence, the curvatures of π_k on both sides of the path do not change as π_k goes along e: the (reciprocals of the) curvatures show how many other thick paths "pad" π_k on each of its sides. Thus,

OBSERVATION 1. *To route a path using $\mathcal{G}$, it is enough to know the curvatures of the path only at the endpoints of the edges of $\mathcal{G}$, i.e., at the nodes of $\mathcal{G}$.*

The other piece of information stored with each edge e is the maximal and minimal index of the paths going through the edge, $e_{\min}$ and $e_{\max}$. Since $s_1 ... s_K$ appear in this order around ∂P^1, all paths with indexes in $[e_{\min}, e_{\max}]$ follow e; all paths with indices outside the interval do not follow the edge. To facilitate routing through $\mathcal{G}$, the pointer to each incident edge e at a node of $\mathcal{G}$ is augmented with the indexes $e_{\min}$ and $e_{\max}$; this way, when a path, routed from the start, arrives at the node, it can determine in constant time which incident edge to follow on the way to the destination.

LEMMA 3.12. *The size of $\mathcal{G}$ is $O(n+K)$.*

PROOF. The nodes of $\mathcal{G}$ are among the terminals in $\mathcal{ST}$ and the vertices of P. Since the paths $\Pi_1 ... \Pi_K$ are non-crossing, $\mathcal{G}$ is planar. The total amount of information stored at a node is proportional to the degree of the node. The total amount of information stored at the edges of $\mathcal{G}$ is proportional to the size of P. □

Reporting a Thick Path Using $\mathcal{G}$ Routing a path π_k is done in two steps. First, we establish which edges of $\mathcal{G}$ the path follows. Since s_k is a leaf, there is a unique edge joining s_k to the rest of $\mathcal{G}$. At a node u of $\mathcal{G}$ the index of the path and the pointers to the edges incident to u tell what the next edge of π_k is. This is repeated until t_k is reached and all the edges of $\mathcal{G}$ that π_k follows are known.

Second, at each node u of $\mathcal{G}$ that the path goes through, the curvature of the path is read off, and the tangent or bi-tangent is computed to extend π_k to go one more link towards t_k. Specifically, let $e = (u, v)$ be the edge that the path takes after u. The curvature κ of π_k at u can be read off the thickness of e at u. According to Observation 1, the curvature of π_k at every internal vertex of e is still κ. Let $e_1 = (v, w)$ be the next edge of $\mathcal{G}$ that π_k goes through. When the next to the last vertex of e is reached by π_k, the new curvature of the path (the curvature at v) is read off the thickness of e_1 at v, and the routing continues.

The operations described can be performed in constant time per vertex of P along the path. Thus,

LEMMA 3.13. *Given $\mathcal{G}$, a thick shortest path can be reported in time proportional to its combinatorial complexity.*

Computing $\mathcal{G}$ We can use the approach of [28] to actually compute $\mathcal{G}$ in linear time. The idea is to use $\mathcal{T}_{sl}$ to compute $\mathcal{G}$ in a "bottom-up" fashion. First, the paths at the bottom level of $\mathcal{T}_{sl}$ are found; this can be done in linear time by the same argument as in [28]. After the paths at level q are computed, the paths at level $q+1$ are routed in time proportional to the complexity of the part of ∂P^1, not used by the paths at the levels 1 to q. The details of the algorithm are the same as in [28]: when a path π_k at the level $q+1$ is routed, the funnel from s_k in the direction of t_k is extended until the funnel hits $\mathcal{Q}$ — a connected subset of the part the boundary already used by the paths from levels 1 to q. The funnel then continues to extend in the direction of t_k from the other end of $\mathcal{Q}$.

The geometric primitives for extending the funnel — computing tangents and bi-tangents, walking through a curved trapezoidal decomposition, checking for intersections, maintaining the upper hull by "wrapping" — can still be implemented to run in constant amortized time per vertex [23]. Thus, $\mathcal{G}$, the data structure for storing thick shortest paths, can be computed within the same time bounds as the data structure of [28] for usual, thin paths.

LEMMA 3.14. *$\mathcal{G}$ can be computed in $O(n+K)$ time.*

From Lemmas 3.12–3.14 follows

THEOREM 3.15. *In linear time a data structure can be computed such that any path in the collection of the all-shortest thick paths in a simple polygon, can be output in time proportional to the combinatorial complexity of the path.*

4. Polygons with Holes

It was shown in [3] that the non-crossing paths problem is NP-hard: Given a set $\mathcal{ST} = (s_k, t_k)_{k=1}^K$ of K pairs of points in the plane, find non-crossing s_k-t_k paths, shortest under the minsum or the minmax objective function. Here, a path is allowed to go around the terminals of other paths; in this respect, every terminal can be treated as a hole (a point obstacle). The hardness of the problem stems from the fact that the terminals lie on (the boundaries of) many different holes. In contrast, if the homotopies of the paths are given, the shortest paths within the same homotopy types can be found efficiently [4, 11].

In this paper we concentrate on the case in which all terminals lie close to the boundary of the outer polygon P. This special case is important in our motivating application (ATM). It is also one of the special cases considered in the study [24] of max-flows in polygonal domains, as it arises in the VLSI formulations of wire routing [17, 30]. So, we assume that $\mathcal{ST} \subset \partial P^1$ (and that the Positioning assumption holds).

As in the case of simple polygons, the algorithm developed in this section works for any objective function that is a non-decreasing function of the length of an individual path. When speaking of a collection of paths, *shortest* will mean a collection that is optimal under a given objective; a *shortest path* will mean a path in an optimal collection. As in the case of simple polygons, we will ensure the uniqueness of the solution by requiring that the optimal collection is Pareto optimal: no paths can be made shorter without increasing the length of another path. In particular, this means that the optimal paths are locally optimal.

We first show how to define the free space for *one* thick path, given the "threadings" of all paths and the order in which the path visits the holes; the shortest path within the free space will be a member of the optimal collection. The number of the threadings of the shortest paths is polynomial in K for fixed h (Lemma 4.1); for a large number h of holes, we show that the minmax version of the problem is weakly NP-hard if $K = 2$ and strongly NP-hard if K is the part of the input. We also prove hardness of approximation and give a pseudopolynomial-time algorithm for rectilinear versions of the problem (see full version [26]).

Small Number of Holes In some applications it is often the case that the number of holes h is small. E.g., in ATM, the holes represent large weather systems and it is often the case that there are not too many of them.

We define the *threading type*, or *threading* of a path to be a vector $\chi \in \{above, below\}^h$ whose j^{th} component indicates whether the hole H_j is above or below the path. The number of different threadings of each of the shortest paths can thus be 2^h. Since, in principle, each of the K shortest paths can have any of the 2^h threadings, the total number of threadings of K paths could potentially be as high as 2^{hK}. Assuming $\mathcal{ST} \subset \partial P^1$ reduces the number of different threadings substantially:

LEMMA 4.1. *If $\mathcal{ST} \subset \partial P^1$, the number of threading types of K shortest paths is at most $(K+1)^h$.*

PROOF. K shortest paths cut P^1 into $K+1$ pieces (see Fig. 2). The threadings of the paths can be specified by indicating how the holes are put into the pieces. □

When the threadings of the K shortest paths are given, the (conservative) obstacles for each of the K paths can be constructed as follows. First, each hole is offset by 1 outside; no reference path is allowed to pass closer than 1 to any hole. Then, as in the case of simple polygons, for $k = 1...K$, $j = 1...h$, we define the k^{th} depth, $d_k(j)$, of the hole H_j as the minimum number of chords that a path from (the image of) H_j crosses before reaching the chord $s_k t_k$ (we assume the holes are mapped to $\mathcal{C}$ together with the pairs in $\mathcal{ST}$). For example, the 4^{th} depth of the hole 1 in Fig. 2 is 1, the 2^{nd} depth of the hole 2 is 0, etc. Then the obstacle $\mathcal{O}_k$ for the reference path π_k^* is the union of correspondingly inflated parts of ∂P^1 and boundaries of the holes: $\mathcal{O}_k =$

$$\bigcup_{l=1}^{2K} \left(P^1(\nu_l, \nu_{l+1})\right)^{2d_k(\nu_l,\nu_{l+1})} \quad \cup \quad \bigcup_{j=1}^{h} (\partial H_j)^{2d_k(j)} \qquad (2)$$

where we have retained the notation from Section 3. Each of the sets in (2) can be broken down to the Minkowski sums of edges of P with the disks. By an argument as in the proof of Lemma 3.9, $\mathcal{O}_k$ is the union of $O(n+K)$ pseudodiscs, and, thus, has complexity $O(n+K)$ [1, 18] and can be built in $O((n+K)\log(n+K))$ time using a randomized algorithm (see [9, Chapter 13] and [15] for further discussions and more references). We still refer to the obstacles in $\mathcal{O}_k$ as *holes* even though the original holes of the domain, when inflated according to (2), may overlap and intersect the outer boundary of the domain; in particular, the domain may become disconnected, so by *domain* we will understand the connected component that contains s_k and t_k.

After the obstacles corresponding to the given threadings are built, we need to find, for each $k = 1...K$, the shortest s_k-t_k path with the given threading ζ_k. Let $\Xi(\zeta_k)$ be the family of homotopy types of simple s_k-t_k paths with the threading ζ_k. Let $\Xi^*(\zeta_k) \subset \Xi(\zeta_k)$ be the family of homotopy types for which the *locally shortest* path is simple. Only one homotopy type $\chi^* \in \Xi(\zeta_k)$ can be a homotopy type of a shortest path with the threading ζ_k; clearly, $\chi^* \in \Xi^*(\zeta_k)$. (Actually, two optimal homotopy types can arise in case of degeneracies, which we will ignore WLOG.) We show how to enumerate all homotopy types in $\Xi^*(\zeta_k)$ by scrolling through all permutations of the holes and, for each permutation, bridging the holes to transform the domain into a simple polygon.

Let A and B be the sets of holes that are above and below (resp.) the kth path, according to ζ_k. Given an ordered sequence $\mathbb{H} = (H_1, ..., H_h)$ of the holes and a threading ζ_k, we define the operation of *bridging* as follows: take the holes from $\mathbb{H}$ one by one and connect each hole with a shortest path to the point just above s_k (if the hole belongs to A) or just below s_k (if the hole belongs to B); treat the holes and the already-built bridges as obstacles. By construction, the bridges do not cross each other. Bridging the holes transforms the domain into a (weakly) simple polygon $P_k = P_k(\mathbb{H}, \zeta_k)$.

Let π^* be the locally shortest path of homotopy χ^*, i.e., the shortest path with threading ζ_k. The following lemma shows that while scrolling through all $h!$ sequences of the holes, χ^* will be "spotted".

LEMMA 4.2. *There exists an ordered sequence $\mathbb{H} = (H_1, ..., H_h)$ of the holes such that the homotopy type of every path in the simple polygon $P_k = P_k(\mathbb{H}, \zeta_k)$ is χ^*, where the homotopy type is viewed w.r.t. the original domain, without the bridges.*

PROOF. *By induction on the number, h^*, of holes touched*

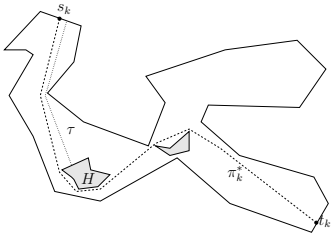

Figure 5: If $H \in A$, H is also above any s_k-t_k path in P_k. π_k^* is dashed; the bridge τ is dotted.

by π^*. The base: If $h^* = 0$, then any sequence $\mathbb{H}$ works, since no bridge crosses π^*. Indeed, suppose the bridge τ from a hole $H' \in A$ crosses π^*. Since τ starts and ends above π^*, τ could be shortened by following π^* between the first and the last points of the crossing, contradicting the fact that τ, by construction, is the *shortest* path from H' to (just above) s_k.

The inductive step: Let H be the first hole touched by π^*; say, $H \in A$. The bridge τ routed, in the absence of any other bridges, from H to (just above) s_k does not cross π^* (otherwise, τ could be shortened). Thus, we bridge H to (just above) s_k using τ (Fig. 5), resulting in a scene with $h-1$ holes, with every s_k-t_k path leaving H on the same side (above/below, viewed w.r.t. the original domain) as π^* leaves it. (Otherwise, the path would cross τ, which is now a part of the outer boundary.) Thus, we can start over: take the next hole touched by π^* and bridge it with a bridge not crossing π^*, and so on. □

The above proof is not constructive since we do not know in advance the order in which π^* touches the holes; thus, we resort to scrolling through all possible orders. Given the threadings of the K paths, for each $k = 1...K$, we go through all of the $h!$ permutations of the holes, bridging the holes in the order given by the permutation, and find the shortest s_k-t_k path; we keep the overall best collection of paths. The running time is $O(h!hK\tau(n+K))$ per threading.[3] Refer to Fig. 6 for an example.

THEOREM 4.3. *K shortest thick non-crossing paths can be found in $O\left((K+1)^h h!hK\tau(n+K)\right)$ time.*

Remarks (1) An alternative approach to find the representative paths for every homotopy type in $\Xi^*(\zeta_k)$ is to decompose the free space into "corridors" and use the universal cover [14] lifting every corridor according to the threading ζ_k[4].
(2) Cabello et al. [6] and Bespamyatnikh [5] suggested efficient schemes for encoding the homotopy types of paths in the plane: Given a path, the algorithms in [5, 6] output its encoding in polynomial time. In our setting, a solution to the *inverse* problem is desired: given an encoding, produce a path of the corresponding homotopy type.
(3) It was shown in [11, 4] how to compute efficiently K shortest thin paths, homotopically equivalent to a given set of paths. Unfortunately, for the case of thick paths, we do not have a more efficient solution than just routing the paths one by one. It is possible that using the approach of [4, 11], finding K thick shortest paths with given threadings can be done more efficiently.
(4) As in the case of simple polygons, the shortest thick paths can be stored in a linear-size data structure such that a shortest path can be reported in time proportional to its combinatorial complexity.
(5) Our algorithms generalize straightforwardly to the case in which each path has its own thickness.
(6) The rectilinear versions of our problem can be solved in polynomial time by introducing an appropriate path-preserving graph [20, 31]. See the full paper [26] for details.
(7) *Hardness results*: Finding K short disjoint paths in a planar graph is NP-hard [16]. Disjoint paths in a plane graph correspond to thick non-crossing paths in a polygonal domain, created by "fattening" the graph edges. This leads to a proof of NP-hardness for the minmax version of our problem. See the full paper [26] for details, as well as hardness of approximation and a pseudopolynomial-time algorithm for rectilinear instances.

[3]We denote by $\tau(M)$ the time complexity of finding one shortest path among obstacles whose boundaries consist of a total of M line segments and circular arcs.
[4]We thank Jack Snoeyink for this observation.

5. Minimum-Cost Flows

In this section we consider the *minimum-cost continuous flow* (or, the *continuous transshipment*) problem — an extension of the standard discrete min-cost network flow problem to continuous domains. We begin by modifying slightly the statement of the thick non-crossing paths problem so that it fits into the framework of flows in the continuum [24, 29]: we "clip off" the semicircular parts at the ends of thick paths and have the terminals of the paths be segments on the boundary of the polygon. This allows treating the paths as (the support of) a flow between the terminals. The fact that (the support of) a flow can be decomposed into the flow's streamlines suggests that a flow can be decomposed into (an *infinite* number of) thin paths. We exploit the idea of "gluing" thick paths (Cor. 3.6) to show that the flow can actually be decomposed into a *finite* set of *thick* paths; the size of the decomposition is linear in the size of the problem input. This is the statement of our Flow Decomposition Theorem, the continuous analogue of the famous network flow theorem. We use the theorem to reduce the continuous transshipment problem to that of finding thick non-crossing paths in the domain. The algorithms for the latter problem developed in the previous sections allow us to solve a class of instances of the transshipment problem efficiently.

Figure 7: Canonical part: chop sausage ends.

Canonical Part of a Thick Path For two points a, b let $\frac{c}{2}(a,b)$ be the (open) semicircle, with diameter ab, to the left of the segment ab. Let $\Pi = (\pi)^1$ be a thick s-t path. Let $s's''$ (resp. $t't''$) be the diameter of $(s)^1$ (resp. $(t)^1$), perpendicular to π at s (resp. at t); let s', t' (resp. s'', t'') lie below (resp. above) π. We define the *canonical part*, $\Pi^{\square}$, of Π to be its part between $s's''$ and $t't''$: $\Pi^{\square} = \Pi \setminus \frac{c}{2}(s', s'') \setminus \frac{c}{2}(t'', t')$ (Fig. 7).

For a point p on ∂P let e_p be the edge of P on which p resides. For $w \in \mathbb{R}^+$ we say that p is *w-inside* e_p if the distance from p to the endpoints of e_p is greater than w. For a point p that is w-inside e_p, let $p^{w-} \in e_p$ (resp. $p^{w+} \in e_p$) be the point at distance w from p going counterclockwise (resp. clockwise) along ∂P: $\{p^{w-}, p^{w+}\} = e_p \cap \mathcal{C}(p, w)$. As

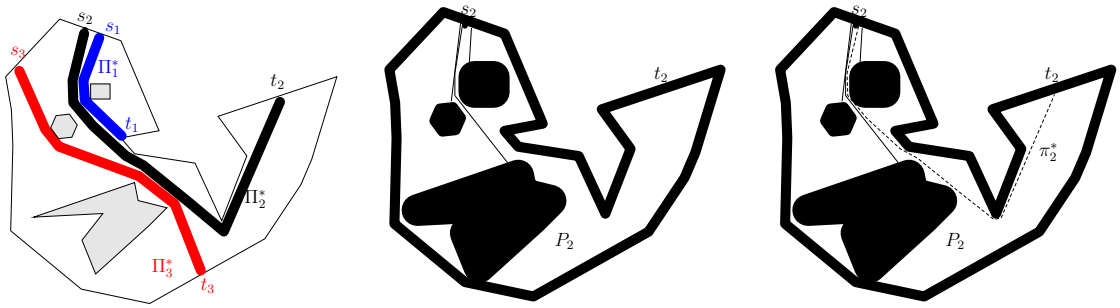

Figure 6: Left: A collection of optimal paths. Center: The free space P_2 for π_2^* after the threading and the order in which the holes are visited by the path are guessed correctly – the holes are inflated and bridged to the outer boundary. Right: The shortest s_2-t_2 path within P_2 is π_2^* (dashed).

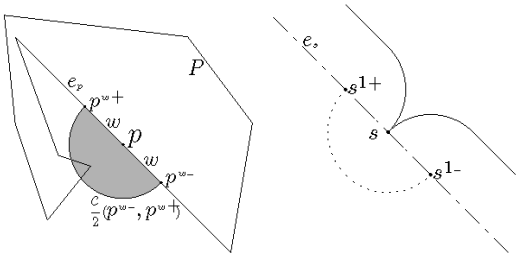

Figure 8: Left: $\mathcal{C}/2(p^{w-}, p^{w+}) \in \mathcal{R}_p$ (shaded) is attached to P along $p^{w-}p^{w+}$. Right: A thick path originating at s will be perpendicular to e_s. ∂P, $\partial\mathcal{P}$ and $\partial\mathcal{P}^1$ are shown with dashed, dotted and solid lines respectively.

in [24], we augment P with a Riemann sheet $\mathcal{R}_p$, attached to P along $p^{w-}p^{w+}$ and place the semicircle $\frac{\mathcal{C}}{2}(p^{w-}, p^{w+})$ in $\mathcal{R}_p$. (The reason for placing $\frac{\mathcal{C}}{2}(p^{w-}, p^{w+})$ in a separate sheet $\mathcal{R}_p$ and not in $\mathbb{R}^2$, the base sheet where P lives, is to make sure that $\frac{\mathcal{C}}{2}(p^{w-}, p^{w+})$ does not intersect P even if they overlap.) Refer to Fig. 8, top.

LEMMA 5.1. *Let the points $s, t \in \partial P$, such that $|st| > 2$, be 1-inside e_s and e_t respectively. Let $\mathcal{P}_{st} = P \cup \frac{\mathcal{C}}{2}(s^{1-}, s^{1+}) \cup \frac{\mathcal{C}}{2}(t^{1+}, t^{1-})$ be P augmented with the two semicircles that reside in the corresponding sheets, $\mathcal{R}_s$ and $\mathcal{R}_t$ (so that $\mathcal{P}_{st} \subset \mathbb{R}^2 \cup \mathcal{R}_s \cup \mathcal{R}_t$ is still a simple polygon). Let $\Pi = (\pi)^1$ be a thick s-t path within $\mathcal{P}_{st}$. Then π is perpendicular to e_s at s and to e_t at t.*

PROOF. Let $\mathcal{P}^1 = \mathcal{P}_{st} \setminus (\partial\mathcal{P}_{st})^1$ be the inward offset by 1 of $\mathcal{P}_{st}$. Since $\Pi = (\pi)^1$ is a thick path within $\mathcal{P}_{st}$, π is a path within $\mathcal{P}^1$. Locally at s, the boundary of $\mathcal{P}^1$ consists of two quarter-circles (parts of $\partial(s^{1-})^1$ and $\partial(s^{1+})^1$), which meet at s normally to e_s (Fig. 8). Thus, any path within $\mathcal{P}$ originating at s is normal to e_s at s. The same argument works for π at t. □

Stick Representation of a Thick Path By Lemma 3.11, the canonical part of a shortest thick path $\Pi = (\pi)^1$ can be written as $\Pi^{\square} = \cup_{x \in \pi} n_x$, where n_x is a segment of length 2 centered at $x \in \pi$ and perpendicular to π at x. Imagine that n_x is a stick whose center, x, moves along π. Then as x moves from s to t, n_x sweeps $\Pi^{\square}$.

In what follows, by a *w-thick path* $\Pi = (\pi)^w$ we will mean the canonical part of $(\pi)^w$; as before, a *thick path* means (the canonical part of) a 1-thick path. For $\vec{w} = (w_1, ..., w_K) \in \mathbb{R}^{+K}$, disjoint thick paths $\Pi_1 ... \Pi_K = (\pi_1)^{w_1} ... (\pi_K)^{w_K}$ are called a collection of *$\vec{w}$-thick* paths.

"Gluing" Thick Paths Let the origins of a collection of shortest thick non-crossing paths be spaced along a segment so that the distance between neighboring origins is equal to the sum of the corresponding path widths; let the destinations of the paths be spaced similarly on another segment. Then routing the paths can be reduced to routing just one thick path from the "midpoint" of the origins to the "midpoint" of the destinations (Fig. 9):

LEMMA 5.2. *Let points $s, t \in \partial P$ be 2-inside e_s, e_t; let $|st| > 4$. Let $s_1 = s^{1-}$, $s_2 = s^{1+}$, $t_2 = t^{1-}$, $t_1 = t^{1+}$. (By triangle inequality, $|s_1t_1|, |s_2t_2| > 2$.) Let $\Pi_1^*, \Pi_2^* = (\pi_1^*)^1, (\pi_2^*)^1$ be the two thick shortest paths between (s_1, t_1) and (s_2, t_2), let $\Pi^* = (\pi^*)^2$ be the shortest 2-thick s-t path within P. Then $cl(\Pi^*) = cl(\Pi_1^* \cup \Pi_2^*)$.*

PROOF. Let $\pi = (\partial\Pi_1^*) \cap (\partial\Pi_2^*)$ be the points, boundary to both Π_1^* and Π_2^*. Observe that $s, t \in \pi$. By Cor. 3.6, π is an s-t path. Since $|st| > 4$, by Cor. 3.7, π is the shortest 2-thick s-t path within P, i.e., $\pi = \pi^*$. The lemma follows now from the stick representations of Π_1^*, Π_2^* and Π^*. □

Lemma 5.2 generalizes straightforwardly to an arbitrary number of paths of arbitrary thicknesses.

The Segment Interconnection Problem Let $\mathcal{I}_{\mathcal{S}} = \cup_1^K s_k^{w_k-} s_k^{w_k+}$, $\mathcal{I}_{\mathcal{T}} = \cup_1^K t_k^{w_k+} t_k^{w_k-}$ be two collections of K segments each; s_k, t_k are the midpoints of $s_k^{w_k-} s_k^{w_k+}, t_k^{w_k+} t_k^{w_k-}$. Let $\mathcal{I}_{\mathcal{ST}} = \mathcal{I}_{\mathcal{S}} \cup \mathcal{I}_{\mathcal{T}}$, $\mathcal{ST} = \cup_1^K (s_k, t_k)$. We call both the segments in $\mathcal{I}_{\mathcal{ST}}$ and the points in $\mathcal{ST}$ *terminals*. Let the following properties hold: (1)$\mathcal{ST} \subset \partial P$; (2) s_k is w_k-inside e_{s_k}; (3) t_k is w_k-inside e_{t_k}; (4) the terminals in $\mathcal{ST}$ are sufficiently separated: $\forall k\, |s_k t_k| > 2w_k$, and $\forall p \in \{s_i, t_i\}$, $\forall q \in \{s_j, t_j\}, i \neq j$, $|pq| > w_i + w_j$: (5) the Positioning assumption (Section 2) holds, i.e., $s_1^{w_1-}, s_1, s_1^{w_1+}, ..., s_K^{w_K-}, s_K, s_K^{w_K+}$ appear in this order clockwise around ∂P and $\forall k, s_k^{w_k-}, s_k, s_k^{w_k+}$ appear before t_k^{w+}, t_k, t_k^{w-}. Consider the *Segment Interconnection Problem* (SIP) — the problem of connecting $s_k^{w_k-} s_k^{w_k+}$ to $t_k^{w_k+} t_k^{w_k-}$ by K non-overlapping thick "strips" of thicknesses $w_1 ... w_K$. By Lemma 5.1, the SIP can be formally stated as the problem of finding (the canonical parts of the) shortest thick non-crossing s_k-t_k paths within $\mathcal{P}_{\mathcal{ST}} = P \bigcup_1^K \left(\frac{\mathcal{C}}{2}(s_k^{w_k-}, s_k^{w_k+}) \cup \frac{\mathcal{C}}{2}(t_k^{w_k+}, t_k^{w_k-}) \right)$, i.e., P augmented with the semicircles, lying in the corresponding Riemann sheets.

Let $\mathrm{SIP}(P, K, \vec{w}, \mathcal{I}_{\mathcal{S}}, \mathcal{I}_{\mathcal{T}})$ be the segment interconnection

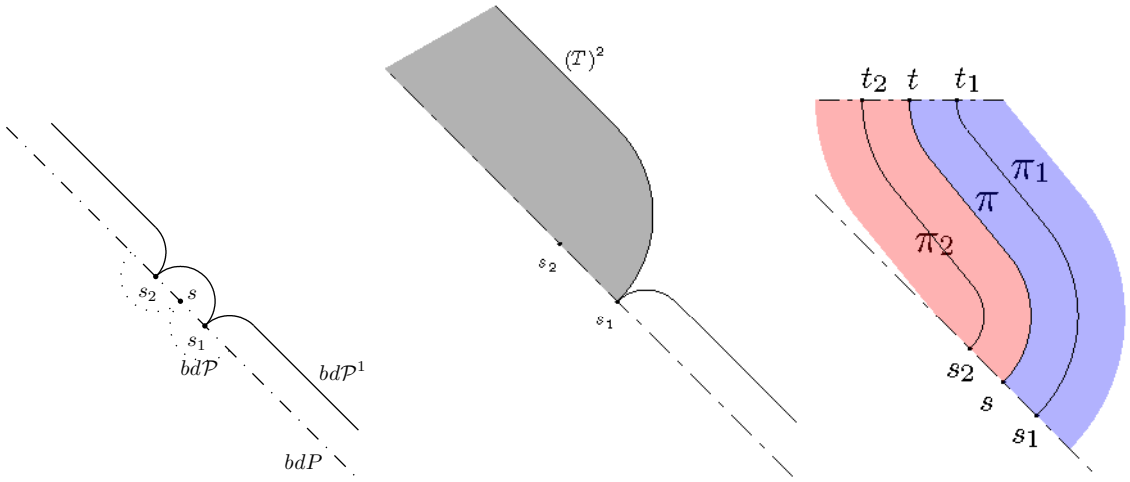

Figure 9: Left: $\mathcal{P} = P \cup \frac{C}{2}(s_1^{1-}, s_1^{1+}) \cup \frac{C}{2}(s_2^{1-}, s_2^{1+})$. ∂P, $\partial\mathcal{P}$ **and** $\partial\mathcal{P}^1$ **are shown with dashed, dotted and solid lines respectively. Center: recall (Section 3) that** π_1 **is routed within** $\mathcal{P}^1$ **treating** $(T)^2 = (\mathcal{P}^1(s_2, t_2))^2$ **as obstacles. Right:** $cl((\pi_1)^1 \cup (\pi_2)^1) = cl((\pi)^2)$.

problem as defined above. Let $\text{SIP}^{\Sigma}(\cdot)$ and $\text{SIP}^{\max}(\cdot)$ be optimal solutions to the minsum and minmax versions of the SIP(·).

THEOREM 5.3. $\text{SIP}^{\Sigma}(\cdot) = \text{SIP}^{\max}(\cdot)$.

PROOF. It follows from our results in Section 3 that there exists a collection of *all-shortest* $\vec{w}$-thick non-crossing $(s_k\text{-}t_k)$ paths within $\mathcal{P}_{\mathcal{ST}}$. By Lemma 5.1, the paths give the solution to the SIP(·). □

Remark Of course, there may be multiple optimal solutions to $\text{SIP}^{\max}(\cdot)$. The statement of Theorem 5.3 must be understood in the sense that there exists a solution that is (Pareto) optimal for both versions.

Continuous Transshipment Problem We recollect from [24, 29] some notions related to flows in polygonal domains. Let $\Gamma_s = \{I_1 ... I_S\}$, $\Gamma_t = \{O_1 ... O_T\}$ be two sets of disjoint (open) segments on ∂P. A *flow* in P is a vector field $\sigma : P \mapsto \mathbb{R}^2$ such that: (1) $\forall x \in P$, $\operatorname{div}\sigma(x) = 0$, i.e., there is no sources/sinks inside P; (2) $\forall x \in \partial P \backslash \Gamma_s \backslash \Gamma_t$, $\sigma(x) \cdot \mathbf{n}(x) = 0$, where $\mathbf{n}(x)$ is the outward pointing unit vector normal to ∂P at x, i.e., the flow penetrates ∂P only at $\Gamma_s \cup \Gamma_t$; (3) $\forall x \in P$, $|\sigma(x)| \le 1$, i.e., each point in P has *unit capacity*. The *value* of the flow is defined as $V = \int_{\Gamma_t} \sigma \cdot \mathbf{n}\, ds$. The *maximum flow* problem [24, 29] asks for a flow, maximizing V; in the *min-cost flow* problem (considered here), V is given and the flow, minimizing certain objective function, is sought.

The Objective Functions Let $\Gamma_s^* \subset \cup_1^S I_i$ be the part of ∂P through which σ actually enters P: $\Gamma_s^* = \{x \in \cup_1^S I_i \,|\, \sigma(x) \cdot \mathbf{n}(\mathbf{x}) \neq 0\}$. Define Γ_t^* similarly. For $s \in \Gamma_s^*$ let $\ell_s = \cup_{\tau=0}^{\infty} \left(s + \int_0^{\tau} \sigma\, dt\right)$ be the *streamline* of σ, going through s. Let $|\ell_s|$ denote the length of ℓ_s.

Strang [29] and Mitchell [24] suggested that the *cost* of a flow may be defined as: (I) the area of the support, supp σ, of the flow, where supp $\sigma = \{x \in P : |\sigma(x)| > 0\}$; or, (II) the "total length of the streamlines", $\int_{s \in \Gamma_s^*} |\ell_s| ds$; or, (III) the length of the longest streamline $\max_{s \in \Gamma_s^*} |\ell_s|$.

It is not hard to see that it suffices to consider only those flows σ that satisfy the following properties: (i) the absolute value of the flow vector is everywhere at its upper bound 1, $\forall x \in$ supp σ, $|\sigma(x)| = 1$; (ii) Γ_s^* and Γ_t^* each consists of at most $S + T$ connected components; (iii) there are no closed streamlines in σ; (iv) σ leaves Γ_s^* (and enters Γ_t^*) normally to ∂P, $\forall x \in \Gamma_s^* \cup \Gamma_t^*$, $|\sigma(x) \cdot \mathbf{n}(x)| = 1$; (v) the lengths $|\Gamma_s^*|$ and $|\Gamma_t^*|$ of Γ_s^* and Γ_t^* satisfy $|\Gamma_s^*| = |\Gamma_t^*| = V$. (Of course, (i)–(v) are not independent, but we do not care.)

Flow Decomposition Theorem For $s \in \Gamma_s^*$ let $d(s) \in \Gamma_t^*$ denote "the other end" of the streamline ℓ_s (the destination of a particle released at s), $d(s) = s + \int_0^{\infty} \sigma\, dt$. Let $o(\cdot)$ be the "inverse" of $d(\cdot)$, i.e., for $t \in \Gamma_t^*$ let $o(t) \in \Gamma_s^*$ be the "origin" of a particle that ended up at t: $o(t) = s | d(s) = t$.

For the flows, enjoying the Properties (i)–(v) above, $d(\cdot)$ is continuous almost everywhere on Γ_s^*. (In the intervals of continuity of $d(\cdot)$ the Fréchet distance $\mathcal{F}(\ell_a, \ell_b)$ between the streamlines ℓ_a and ℓ_b and the distance $|d(a)d(b)|$ between $d(a)$ and $d(b)$ satisfy $\mathcal{F}(\ell_a, \ell_b) = |d(a)d(b)| = |ab|$.) Similarly, $o(\cdot)$ is continuous almost everywhere on Γ_t^*. Let $\mathbb{O} = \{o_1 ... o_{S^*}\} \subset \Gamma_s^*$ and $\mathbb{D} = \{d_1 ... d_{T^*}\} \subset \Gamma_t^*$ be the points of discontinuity of $d(\cdot)$ and $o(\cdot)$. Let $\mathcal{I}_S$ (resp. $\mathcal{I}_T$) be the set of segments into which Γ_s^* (resp. Γ_t^*) is decomposed by $o_1 ... o_{S^*}$ (resp. $d_1 ... d_{T^*}$); these segments are the maximal contiguous subsets of Γ_s^*, Γ_t^* such that the flow from a subset of Γ_s^* goes to a subset of Γ_t^*. By definition, the number of segments in $\mathcal{I}_S$ equals the number of segments in $\mathcal{I}_T$; call this number K.

The support of the flow can be "slit" along the streamlines originating at o_i^-, o_i^+, $i = 1...S^*$ and the streamlines ending at d_j^-, d_j^+, $j = 1...T^*$ into K "smaller" flows going from the segments in $\mathcal{I}_S$ to the corresponding segments in $\mathcal{I}_T$. Formally, consider segments $e \in \mathcal{I}_S$ and $f = d(e) \in \mathcal{I}_T$. ($|e| = |f|$.) Let σ_e be the restriction of σ on the subset of P through which the streamlines from e to f pass: supp $\sigma_e = \{x \in P \,|\, \exists s \in e | x \in \ell(s)\}$, $\sigma_e = \sigma$ on supp σ_e, $\sigma_e = 0$ o.w. Then $\sigma = \sum_{e \in \mathcal{I}_S} \sigma_e$, i.e., a flow can be decomposed into "sub-flows", such that each sub-flow goes from a single source segment to a single sink segment.

Let $ab \in \mathcal{I}_S$, $cd \in \mathcal{I}_T$ be two segments such that the (sub-)flow σ_{ab} of value $2w$, $w \in \mathbb{R}$, goes from ab to cd:

$\forall x \in ab, d(x) \in cd$, $\forall y \in cd, o(y) \in ab$; by Properties (iv) and (v), $|ab| = |cd| = 2w$. Let π_{ad} and π_{bc} be the streamlines going from a to d and from b to c. For $\xi \in (0,1)$ let $\pi(\xi)$ be the (thin, usual) path from $ab(\xi) = (1-\xi)a + \xi b$ to $cd(\xi) = (1-\xi)d + \xi c$, routed treating $(\pi_{ad})^{2w\xi}$ and $(\pi_{bc})^{2w(1-\xi)}$ as obstacles. Then $\pi(\xi)$, the set of points at distance $2w\xi$ from π_{ab} and at distance $2w(1-\xi)$ from π_{cd} (so that $\mathcal{F}(\pi(\xi), \pi_{ab}) = 2w\xi$, $\mathcal{F}(\pi(\xi), \pi_{cd}) = 2w(1-\xi)$), is the streamline of σ going from $ab(\xi)$ to $cd(\xi)$. Since exactly one streamline passes through each point of the flow support, the support can be written as the union of the streamlines: supp $\sigma_{ab} = \cup_{\xi\in(0,1)}\pi(\xi)$. Lemma 5.4 below proves that supp σ_{ab} is in fact a shortest w-thick s-t path within P, where $s = ab(\frac{1}{2})$ and $t = cd(\frac{1}{2})$ are the midpoints of ab and cd.

LEMMA 5.4. *supp σ_{ab} is a w-thick shortest s-t path*

PROOF. For $M \in \mathbb{N}$ let ab^M, cd^M be the subdivisions of ab and cd into M segments of length $\delta = 2w/M$. Consider the Segment Interconnection Problem SIP$(P, M, \delta\vec{1}^M, ab^M, cd^M)$, where $\vec{1}^M \in \mathbb{R}^M$ is the vector of M ones. By Thm. 5.3, the solution to the SIP is given by a set of M $\frac{\delta}{2}$-thick shortest paths between the mid-points of the segments in ab^M and cd^M. By Lemma 5.2, these paths can be "glued" into one w-thick shortest s-t path. Since the above is true for arbitrary M, it is also true in the limit, $M \to \infty$, when the paths become the streamlines of the flow. □

Thus, the discrete network Flow Decomposition Theorem can be extended to the continuum:

THEOREM 5.5. **Continuous Flow Decomposition Theorem** *The support of a minimum-cost flow can be decomposed into a set of thick paths; the size of the decomposition is linear in the size of the description of the flow.*

In Section 3 we built a data structure for storing thick shortest paths; it is an extension of the data structure of Papadopolou [28] for storing *thin* paths (called in [28] the "forest" of shortest paths). The flows in the continuum provide a new view on these data structures: it can be seen that our data structure actually represents the support of the min-cost flow from $\cup_1^K(s_i^{1+}, s_i^{1-})$ to $\cup_1^K(t_i^{1-}, t_i^{1+})$.

Acknowledgments We thank Alon Efrat, Jimmy Krozel, and Evanthia Papadopoulou for helpful discussions and anonymous referees for their suggestions. This work was done while second author was at Stony Brook University. The work was supported by grants from NASA (NAG2-1620), Metron Aviation (NASA subcontract, NAS2-02075), and NSF (CCF-0431030, CCF-0528209).

6. References

[1] P. K. Agarwal and M. Sharir. Arrangements and their applications. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 49–119. Elsevier Science B.V. North-Holland, Amsterdam, 2000.

[2] A. Aggarwal, M. M. Klawe, S. Moran, P. W. Shor, and R. Wilber. Geometric applications of a matrix searching algorithm. In *Proc. 2nd Annu. ACM Sympos. Comput. Geom.*, pages 285–292, 1986.

[3] O. Bastert and S. P. Fekete. Geometric wire routing. Technical Report 332, Zentrum für Angewandte Informatik, 1998.

[4] S. Bespamyatnikh. Computing homotopic shortest paths in the plane. *J. Algorithms*, 49(2):284–303, 2003.

[5] S. Bespamyatnikh. Encoding homotopy of paths in the plane. In *Proc. LATIN'04*, LNCS 2976, pages 329–338, 2004.

[6] S. Cabello, Y. Liu, A. Mantler, and J. Snoeyink. Testing homotopy for paths in the plane. In *SCG '02: Proceedings of the eighteenth annual symposium on Computational geometry*, pages 160–169, New York, NY, USA, 2002.

[7] F. Chin, J. Snoeyink, and C. A. Wang. Finding the medial axis of a simple polygon in linear time. *Discrete Comput. Geom.*, 21(3):405–420, 1999.

[8] R. Cole and A. Siegel. River routing every which way, but loose. In *Proc. 25th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 65–73, 1984.

[9] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[10] A. Efrat, S. Kobourov, M. Stepp, and C. Wenk. Growing fat graphs. In *SCG '02: Proceedings of the eighteenth annual symposium on Computational geometry*, pages 277–278, New York, NY, USA, 2002. ACM Press.

[11] A. Efrat, S. G. Kobourov, and A. Lubiw. Computing homotopic shortest paths efficiently. In *Proceedings of the 10th Annual European Symposium on Algorithms*, pages 411–423, London, UK, 2002. Springer-Verlag.

[12] S. Gao, M. Jerrum, M. Kaufman, K. Mehlhorn, and W. Rülling. On continuous homotopic one layer routing. In *SCG '88: Proceedings of the fourth annual symposium on Computational geometry*, pages 392–402, New York, NY, USA, 1988. ACM Press.

[13] L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. Tarjan. Linear time algorithms for visibility and shortest path problems inside simple polygons. In *SCG '86*, pages 1–13, New York, NY, USA, 1986. ACM Press.

[14] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Comput. Geom. Theory Appl.*, 4:63–98, 1994.

[15] S. Hirsch and E. Leiserowitz. Exact construction of minkowski sums of polygons and a disc with application to motion planning. Technical report ECG-TR181205-01, Tel-Aviv University, 2002.

[16] H. v. d. Holst and J. C. d. Pina. Length-bounded disjoint paths in planar graphs. *Discr. Appl. Math.*, 120(1-3):251–261, 2002.

[17] C.-P. Hsu. General river routing algorithm. In *Proceedings 20th conference on Design automation*, pages 578–583, 1983.

[18] K. Kedem, R. Livne, J. Pach, and M. Sharir. On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete Comput. Geom.*, 1:59–71, 1986.

[19] Y. Kusakari, H. Suzuki, and T. Nishizeki. Finding a shortest pair of paths on the plane with obstacles and crossing areas. In J. S. et al., editor, *Algorithms and Computation*, pages 42–51. Springer, Berlin,, 1995.

[20] D. T. Lee, C. D. Yang, and C. K. Wong. Rectilinear paths among rectilinear obstacles. *Discr. Appl. Math.*, 70:185–215, 1996.

[21] C. E. Leiserson and F. M. Maley. Algorithms for routing and testing routability of planar VLSI layouts. In *Proc. 17th Annu. ACM Sympos. Theory Comput.*, pages 69–78, 1985.

[22] F. M. Maley. *Single-Layer Wire Routing and Compaction*. MIT Press, Cambridge, MA, 1990.

[23] E. A. Melissaratos and D. L. Souvaine. On solving geometric optimization problems using shortest paths. In *Proc. 6th Annu. ACM Sympos. Comput. Geom.*, pages 350–359, 1990.

[24] J. S. B. Mitchell. On maximum flows in polyhedral domains. *J. Comput. Syst. Sci.*, 40:88–123, 1990.

[25] J. S. B. Mitchell. Geometric shortest paths and network optimization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 633–701. Elsevier Science B.V. North-Holland, Amsterdam, 2000.

[26] V. Polishchuk. Thick Non-Crossing Paths and Minimum-Cost Continuous Flows in Geometric Domains. PhD thesis. Stony Brook University, 2007.

[27] E. Papadopoulou. Personal communication.

[28] E. Papadopoulou. k-pairs non-crossing shortest paths in a simple polygon. *Int. J. Comp. Geom. Appl.*, 9(6):533–552, 1999.

[29] G. Strang. Maximal flow through a domain. *Math. Program.*, 26:123–143, 1983.

[30] J. Takahashi, H. Suzuki, and T. Nishizeki. Finding shortest non-crossing rectilinear paths in plane regions. In *ISAAC*, pages 98–107, 1993.

[31] C. D. Yang, D. T. Lee, and C. K. Wong. On bends and lengths of rectilinear paths: a graph theoretic approach. *Internat. J. Comput. Geom. Appl.*, 2(1):61–74, 1992.