k-Pairs Non-Crossing Shortest Paths in a Simple Polygon

Evanthia Papadopoulou

Northwestern University, Evanston, Illinois 60208, USA

Abstract. This paper presents an $O(n+k)$ time algorithm to compute the set of k *non-crossing* shortest paths between k source-destination pairs of points on the boundary of a simple polygon of n vertices. Paths are allowed to overlap but are not allowed to cross in the plane. A byproduct of this result is an $O(n)$ time algorithm to compute a balanced geodesic triangulation which is easy to implement. The algorithm extends to a simple polygon with one hole where source destination pairs may appear on both the inner and outer boundary of the polygon. In the latter case, the goal is to compute a collection of non-crossing paths of minimum total cost.

1 Introduction

The *k-pairs non-crossing shortest path problem* in a simple polygon is defined as follows. Given a simple polygon P of n vertices and k source-destination pairs of points $(s_i, t_i), i = 1, \ldots, k$, along the boundary of P, find the collection of k *non-crossing* shortest paths connecting every pair (s_i, t_i) that lie entirely in P. *Non-crossing* paths are allowed to overlap i.e., share vertices or edges, but they are not allowed to cross each other. Note that if the source-destination pairs are arbitrarily located on the polygon boundary there need not be any set of non-crossing paths between them. In the case where P contains a hole and the k source-destination pairs of points may appear in both the outer and the inner boundary of P the problem is to compute the collection of k *non-crossing* paths whose total length is minimum.

The *non-crossing* shortest path problem was introduced by D.T. Lee in [7]. In [12] Takahashi *et al* considered the following problem: Given an undirected plane graph with nonnegative edge lengths, and k terminal pairs on two specified face boundaries, find k *non-crossing* paths in G, each connecting a terminal pair, whose total length is minimum. They presented an $O(n \log n)$ time algorithm, where n is the total number of vertices in the graph (including terminal pairs), using divide and conquer. In a subsequent paper [11, 10], Takahashi *et al* consider the non crossing shortest path problem in a polygonal domain of n rectangular obstacles and the L_1 metric and provide an $O((n+k)\log(n+k))$ time algorithm. Motivation for non-crossing shortest path problems comes from VLSI layout design (see [7, 10, 12, 11]).

In the case of a simple polygon or a simple polygon with one hole, we can obtain an $O(k + n \log k)$ time algorithm using a divide and conquer approach

similar to [12, 11] and the $O(n)$ time algorithm of Lee and Preparata[8] for the single pair shortest path problem in a simple polygon. Alternatively, for a simple polygon without holes, the problem can be solved by answering k shortest path queries using the data structure of Guibas and Hershberger [4]. The data structure of [4] can be constructed in $O(n)$ time and space and can answer shortest path queries between pairs of points in a simple polygon in $O(\log n + l)$ time where l is the number of links of the shortest path.

In this paper we present an $O(n + k)$ time algorithm for the k pairs non-crossing shortest path problem in a simple polygon and a polygon with one hole. The collection of non-crossing shortest paths is organized in a forest that allows shortest path queries between any given pair to be answered in time proportional to the size of the path. We also make an observation that can reduce the time complexity of the algorithm of Takahashi *et al* [11, 10] from $O((n + k)\log(n + k))$ to $O(k + n\log n)$. (Note that k can be large compared to n.) A byproduct of our algorithm is a simple $O(n)$ method for computing a *balanced geodesic triangulation* of a simple polygon. A *balanced geodesic triangulation* is a decomposition of a simple polygon into triangle-like regions, called *geodesic triangles*, with the property that any line segment interior to P crosses only $O(\log n)$ geodesic triangles [2, 3]. Such a decomposition finds often application in problems involving simple polygons e.g., ray shooting or shortest path queries [2, 3] and can be computed in $O(n)$ time [2]. The idea is to compute the collection of shortest paths between the following pairs of vertices $(v_1, v_{n/3}), (v_{n/3}, v_{2n/3}), (v_{2n/3}, v_1), (v_1, v_{n/6}), (v_{n/6}, v_{n/3}), (v_{n/3}, v_{n/2})$, etc. until a pair consists of consecutive vertices. Our $O(n)$ time algorithm for computing the collection of non-crossing shortest paths among the above $O(n)$ pairs of vertices gives an alternative method for computing a balanced geodesic triangulation which is simple to implement. In general, a collection of non-crossing shortest paths between pairs of points on the boundary of simple polygon can be used to obtain useful polygon decompositions by including in the decomposition shortest paths between certain boundary points.

2 Preliminaries

Let P be a simple polygon and let ∂P be its boundary. Given two points $x, y \in P$, $\pi(x, y)$ denotes the shortest path from x to y that lies entirely in P. $\pi(x, y)$ is assumed to be directed from x to y. We are given a set of k source-destination pairs along ∂P, $\mathcal{ST} = \{(s_i, t_i), 1 \leq i \leq k\}$. We shall adopt the convention that sources $s_i, 1 \leq i \leq k$, are ordered clockwise along ∂P and that for any pair, s_i appears before t_i when we traverse ∂P clockwise starting at s_1. Source s_1 is regarded as the starting point along ∂P. Unless otherwise noted, we assume a clockwise traversal of ∂P starting at s_1.

Let ∂P be mapped onto a unit circle. Then the k source-destination pairs of points get mapped to k circle chords, $\overline{s_i t_i}, i = 1, \ldots, k$. Figure 1 shows the mapping. If chord $\overline{s_i t_i}$ intersects $\overline{s_j t_j}$ then clearly $\pi(s_i, t_i)$ and $\pi(s_j, t_j)$ must cross each other. We say that pairs (s_i, t_i) and (s_j, t_j) are *non-crossing*, if and only if the corresponding chords in the unit circle $\overline{s_i t_i}$ and $\overline{s_j t_j}$ do not intersect. For

example, in figure 1, pairs $(s_1, t_1), (s_2, t_2), (s_3, t_3)$ and (s_5, t_5) are non-crossing while pairs (s_4, t_4) and (s_5, t_5) are crossing. It is not difficult to see that shortest paths between two pairs of points in $\mathcal{ST}$ are non-crossing if and only if the pairs are themselves non-crossing.

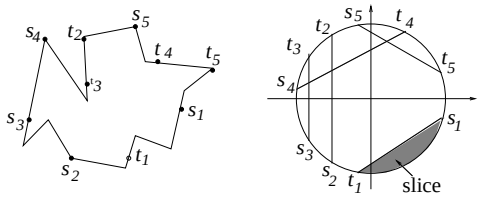

Fig. 1. Mapping of k source-destination pairs of vertices into k chords on the unit circle.

Lemma 1. *Given a simple polygon P of n vertices and k source-destination pairs along ∂P, (s_i, t_i), $1, \ldots, k$, detecting if any two shortest paths connecting s_i to t_i cross each other can be done in $O(n + k)$ time.*

Given the unit circle representing ∂P and a pair (s_i, t_i), the region of the unit circle enclosed by the chord $\overline{s_i t_i}$ and the arc from s_i to t_i is referred to as a *slice* and is denoted as $sl(s_i, t_i)$. (A clockwise traversal is assumed.) For the pair (s_1, t_1) we define two slices, $sl(s_1, t_1)$ and $sl'(s_1, t_1)$, where $sl'(s_1, t_1)$ is the complement of $sl(s_1, t_1)$ i.e., the region of the unit circle enclosed by the chord $\overline{s_i t_i}$ and the arc from t_1 to s_1. In figure 1, $sl(s_1, t_1)$ is shown shaded. Note that paths $\pi(s_i, t_i)$ and $\pi(s_j, t_j)$ are *non-crossing* if and only if the corresponding slices, $sl(s_i, t_i)$ and $sl(s_j, t_j)$, do not intersect or one of them totally contains the other. In the former case, the two non-crossing pairs are said to be in *series*, and in the latter, they are said to be in *parallel*. The destination points of pairs that are in parallel are in reverse order as the source points. Figure 2 shows an example of a set of non-crossing slices; slices $sl(s_1, t_1), sl(s_3, t_3), sl(s_6, t_6)$ are in series while $sl(s_1, t_1), sl(s_2, t_2)$ are in parallel.

Consider now a tree, T_{sl}, representing the set of non-crossing slices where the root corresponds to the unit disk and each node represents a slice. The children of the root are $sl(s_1, t_1)$ and $sl'(s_1, t_1)$. Slices that are totally contained in $sl(s_i, t_i)$ are descendants of $sl(s_i, t_i)$ in the tree. The immediate children of $sl(s_i, t_i)$ (resp. $sl'(s_1, t_1)$) are those slices in series that are totally contained in $sl(s_i, t_i)$ (resp. $sl'(s_1, t_1)$) but are not contained in any other descendent of $sl(s_i, t_i)$. Figure 2 shows the tree representation of the corresponding slices. The nodes are labeled by the slice number; slice 0 refers to the unit disk. Note that T_{sl} may be balanced or completely unbalanced.

Let R be the set of all points in $\mathcal{ST}$. We assume that R is ordered according to a clockwise traversal of ∂P starting at $r_1 = s_1$. Given any two points x, y along ∂P, we say that y follows x, and denote as $x < y$, if and only if y follows x in a clockwise traversal of the polygon boundary starting at s_1. Given $R' \subseteq R$

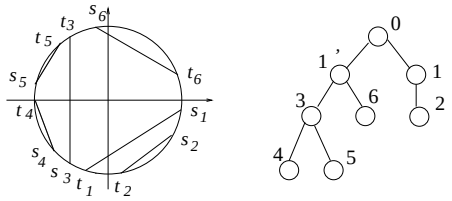

Fig. 2. Non-crossing slices and the tree representation.

and $r_j \in R'$, let $n_{R'}(r_j)$ denote the point in R' following r_j and $p_{R'}(r_j)$ denote the point in R' preceding r_j. For $R' = R$, $n_R(r_j) = r_{j+1}$ and $p_R(r_j) = r_{j-1}$. Let $R' \subseteq R$ be such that the shortest paths between any two pairs of distinct points are disjoint. The area enclosed by the collection of $\pi(r_j, n_{R'}(r_j))$ for every $r_j \in R'$ is called the *geodesic polygon* induced by R'. In figure 3, the shaded area is the geodesic polygon induced by $R' = \{r_1, r_4, r_6, r_7, r_{10}\}$. The last common vertex along $\pi(r_j, n_{R'}(r_j))$ and $\pi(r_j, p_{R'}(r_j))$ is an *apex* of the geodesic polygon, and is denoted as $\alpha(r_j)$. If r_j is the only common point of $\pi(r_j, n_{R'}(r_j))$ and $\pi(r_j, p_{R'}(r_j))$ then $\alpha(r_j) = r_j$. In figure 3, $\alpha(r_4) = a$ and $\alpha(r_1) = r_1$. The common part of $\pi(r_j, n_{R'}(r_j))$ and $\pi(r_j, p_{R'}(r_j))$ (i.e., $\pi(r_j, \alpha(r_j))$) is referred to as the *geodesic link* of r_j. Two consecutive apexes of the geodesic polygon, $\alpha(r_j)$ and $\alpha(n_{R'}(r_j))$, are joined by their shortest path, $\pi(\alpha(r_j), \alpha(r_{j+1}))$, which is clearly a convex chain with convexity facing towards the interior of the geodesic polygon.

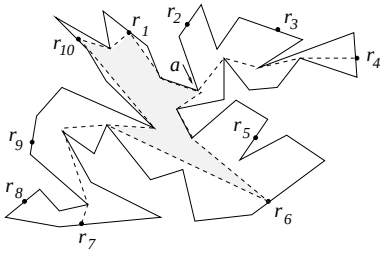

Fig. 3. The geodesic polygon induced by $R' = \{r_1, r_4, r_6, r_7, r_{10}\}$.

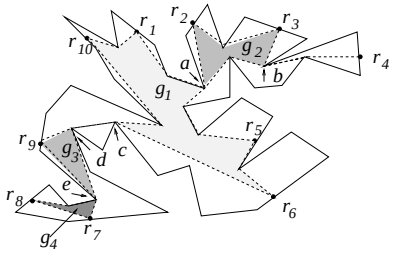

Fig. 4. The geodesic decomposition of $R = \{r_1, \ldots, r_{10}\}$.

If R' contains pairs of distinct points whose shortest paths share some common part, the collection of $\pi(r_j, n_{R'}(r_j))$ induces more than one geodesic polygon in P. In figure 4, set $R = \{r_1, \ldots, r_{10}\}$ induces four geodesic polygons g_1, g_2, g_3, g_4, where $g_1 = \{r_1, a, r_5, r_6, c, r_{10}\}$, $g_2 = \{a, r_2, r_3, b\}$, $g_3 = \{d, e, r_9\}$, and $g_4 = \{e, r_7, r_8\}$. Two paths $\pi(r_i, n_{R'}(r_i))$ and $\pi(r_j, n_{R'}(r_j))$, $j > i + 1$, that are not disjoint share a subpath of at least one vertex which is referred to as a *geodesic link*. For example in figure 4, segment $\overline{cd}$ is common to $\pi(r_6, r_7)$ and $\pi(r_9, r_{10})$ i.e., $\overline{cd}$ is a geodesic link. Note that a geodesic link may consist of a single vertex in which case the incident geodesic polygons have a common apex

(e.g., vertex a in figure 4). The collection of geodesic polygons and links induced by $\pi(r_i, n_{R'}(r_i))$ for every $r_i \in R'$ is called the *geodesic decomposition* of P induced by R' and is denoted as $\gamma(R')$. In figure 4, the geodesic decomposition of R consists of four geodesic polygons g_1, g_2, g_3, g_4, The geodesic link joining g_1 and g_2 is vertex a, the geodesic link joining g_1 and g_3 is $\pi(c,d) = \overline{cd}$ and the geodesic link joining g_3 and g_4 is vertex e. The geodesic link of r_4 is $\pi(r_4, b)$. For every $r_j, j \neq 4$, $\alpha(r_j) = r_j$, and $\alpha(r_4) = b$.

Apex $\alpha(r_1)$ is considered to be the root of the geodesic decomposition. This induces a unique directed path from the root to every geodesic polygon which defines a parent-child relation between the geodesic polygons. The apex of a geodesic polygon g incident to its parent is called the *main apex* of g and is denoted as $\alpha(g)$. Given a geodesic polygon g and $r_j \in R'$ let $\alpha(r_j, g)$ be the apex of g that belongs to $\pi(\alpha(g), r_j)$. (If g contains $\alpha(r_j)$ then $\alpha(r_j, g) = \alpha(r_j)$). In figure 4, $\alpha(r_4, g_1) = a$ and $\alpha(r_4, g_3) = d$.

Given a vertex v along a convex chain C and a point p, we say that v is *supporting* (with respect to $\overline{pv}$) or that segment $\overline{pv}$ is supporting to C if the line containing $\overline{pv}$ is tangent to C.

3 The Algorithm

The collection of shortest paths between every pair of points in $\mathcal{ST}$ (given that $\mathcal{ST}$ is non-crossing) forms a forest denoted as $\mathcal{E}$. Our algorithm computes $\mathcal{E}$ in $O(n+k)$ time and processes it to answer shortest path queries between any pair in $\mathcal{ST}$ in time proportional to the length of the path. In particular, any tree of $\mathcal{E}$ can be transformed into a rooted tree by assigning the source of minimum index as the root. Let $nca_{\mathcal{E}}(s_i, t_i), (s_i, t_i) \in \mathcal{ST}$, denote the nearest common ancestor of s_i and t_i in $\mathcal{E}$. (Note that s_i and t_i must belong to the same tree of $\mathcal{E}$.) Then $\pi(s_i, t_i) = \pi(s_i, nca_{\mathcal{E}}(s_i, t_i) \cup \pi(nca_{\mathcal{E}}(s_i, t_i), t_i)$. Computing the nearest common ancestor in $\mathcal{E}$ for every pair $(s_i, t_i) \in \mathcal{ST}$ can be easily done in linear time in a bottom-up fashion using T_{sl} as a guide.

If all source-destination pairs are in series, we can easily compute the collection of shortest paths between them in linear time using any ordinary shortest path algorithm for a single pair of points[8, 1, 5]. In particular, we can use the Lee-Preparata algorithm [8] for each pair independently. The following lemma assures that the time complexity remains linear.

Lemma 2. *The collection of non-crossing shortest paths between k source-destination pairs of points on ∂P can be found in linear time if the pairs are in series.*

In general, we compute $\mathcal{E}$ in a bottom-up fashion using the tree of slices, T_{sl}, as a guide. In phase 1, we compute the geodesic decomposition of P induced by R. This reveals the collection of $\pi(s_i, t_i)$ for every pair (s_i, t_i) at a leaf node of T_{sl}. For any pair (s_i, t_i) at the bottom level of T_{sl}, we color all edges along $\pi(s_i, t_i)$ red, and add them to $\mathcal{E}$. The remaining edges of the geodesic decomposition are assumed to be white. Let R^q, for $1 \leq q \leq h$, where h is the height of T_{sl}, denote the set of points appearing at levels 1 to $(h-q+1)$ in T_{sl}. Note that $R^1 = R$ and

$R^h = \{s_1, t_1\}$. In phase q, $1 < q \leq h$, we compute the geodesic decomposition of P induced by R^q which reveals the collection of $\pi(s_i, t_i)$ for every pair (s_i, t_i) at level $(h - q + 1)$ of T_{sl}. Edges of the decomposition at phase q are either red or white. Red edges are those that have been added to $\mathcal{E}$ in some previous phase. At the end of phase q, we add all white edges along $\pi(s_i, t_i)$ to $\mathcal{E}$ and color them red, for every pair (s_i, t_i) at level $(h - q + 1)$ of T_{sl}.

At every phase $q, 1 \leq q \leq h$, the geodesic decomposition of R^q is maintained. Geodesic polygons and links are kept as a doubly linked list of their vertices. In general, an apex is adjacent to two vertices in its geodesic polygon and one vertex in the incident geodesic link. An exception are apexes that are common to two geodesic polygons (e.g. apex a in figure 4), which are incident to two vertices in each of the incident geodesic polygons. Edges of geodesic polygons and geodesic links are marked as red or white, where all red edges have been included in $\mathcal{E}$. In order to have the ability to extract the white edges of $\pi(s_i, t_i)$ for some pair (s_i, t_i) at level $(h-q+1)$ of T_{sl} without visiting the red edges along $\pi(s_i, t_i)$, we also maintain a doubly linked list of the polygon vertices incident to white edges, referred to as the *white-list*. The white-list includes $\alpha(r_j)$ for every $r_j \in R^q$. Two vertices v, u in the white-list are joined by a link if and only if v and u belong to the shortest path from $\alpha(r_j)$ to $\alpha(n_{R^q}(r_j))$ and either $\overline{vu}$ is a white edge or $\pi(v, u)$ is a maximal red subpath of $\pi(\alpha(r_j), \alpha(n_{R^q}(r_j)))$. Every link of the white-list corresponds either to a white edge of the decomposition, referred to as a *white link*, or to a maximal sequence of red edges, referred to as a *red link*. Note that a white edge along a geodesic link appears twice in the white-list and thus each of the incident vertices also appears twice.

Let's consider the geodesic decomposition of phase 1, i.e., the geodesic decomposition of R. Pairs $(r_i, r_{i+1}), 1 \leq i \leq 2k$, are clearly in series and therefore the collection of $\pi(r_i, r_{i+1})$ can be computed in $O(n + k)$ time (lemma 2). Once $\pi(r_i, r_{i+1})$ for every $i, 1 \leq i \leq 2k$, have been computed, the geodesic decomposition of P can be easily obtained in linear time. For every pair of points (s_i, t_i) at the bottom level of T_{sl}, we color edges along $\pi(s_i, t_i)$ red and add them to $\mathcal{E}$. We also build the white list by a simple scan starting at $\alpha(r_1)$.

To compute the geodesic decomposition of $R^q, q > 1$, we use the geodesic decomposition of R^{q-1}. To facilitate updating we use the shortest path tree from s_1 to all points in R. Computing the shortest path tree from s_1 can be done in linear time using the algorithm of Guibas et al [5] or the simpler to implement algorithm of Hershberger and Snoeyink [6]. Given two points $r_i, r_j \in R$, the nearest common ancestor of r_i and r_j in the shortest path tree from s_1 is denoted as $nca(r_i, r_j)$. Suppose that the geodesic decomposition of P induced by R^{q-1} has been computed, and that we want to compute the geodesic decomposition induced by R^q. Let $R'^q = R^{q-1} - R^q$. For brevity, we will use $n(r_j)$ to denote $n_{R^q}(r_j)$, $p(r_j)$ to denote $p_{R^q}(r_j)$, and γ to denote the geodesic decomposition during phase q. At the end of phase q, $\gamma = \gamma(R^q)$.

Consider $r_j \in R^q$ such that $n(r_j) \neq n_{R^{q-1}}(r_j)$. Since r_j and $n(r_j)$ are neighboring points in R^q, we need to compute $\pi(\alpha(r_j), \alpha(n(r_j)))$ and update the geodesic decomposition. For this purpose we incrementally compute $\pi(\alpha(r_j), \alpha(r_i))$

for every $r_i \in R'^q \cup \{n(r_j)\}$ with $r_j < r_i \leq n(r_j)$, updating at the same time the geodesic decomposition. Suppose that $\pi(\alpha(r_j), \alpha(r_i))$ has been computed for some $r_i \in R'^q, r_j < r_i < n(r_j)$, and that the geodesic decomposition has been updated accordingly. i.e., $\gamma = \gamma(R^q \cup \{r \in R^{q-1}, r \geq r_i\})$. It is enough to update the geodesic polygon g that contains $\alpha(r_i)$ according to $\pi(\alpha(r_j, g), \alpha(r_l, g))$.

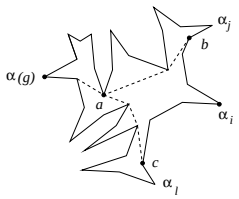

Fig. 5. $\gamma(\alpha(g), \alpha_j, \alpha_i, \alpha_l)$

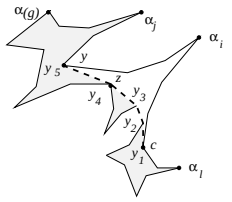

Fig. 6. The update of g in case 3.

Let α_j, α_i and α_l denote $\alpha(r_j, g), \alpha(r_i)$ and $\alpha(r_l, g)$ respectively. Note that α_j or α_l may coincide with the main apex of g. Consider the geodesic decomposition induced by $\{\alpha(g), \alpha_j, \alpha_i, \alpha_l\}$, denoted as $\gamma(\alpha(g), \alpha_j, \alpha_i, \alpha_l)$. If $\alpha(g) = \alpha_j$ or $\alpha(g) = \alpha_l$, $\gamma(\alpha(g), \alpha_j, \alpha_i, \alpha_l)$ must form a geodesic triangle (i.e., have three apexes). Otherwise, $\gamma(\alpha(g), \alpha_j, \alpha_i, \alpha_l)$ forms either a geodesic quadrilateral (four apexes, figure 5) or two geodesic triangles (figure 7). In any case, let $g(a_i)$ denote the geodesic polygon of $\gamma(\alpha(g), \alpha_j, \alpha_i, \alpha_l)$ containing α_i.

Let a, b and c denote the apexes of $\gamma(\alpha(g), \alpha_j, \alpha_i, \alpha_l)$, other than α_i, such that a is the last common vertex along $\pi(\alpha(g), \alpha_j)$ and $\pi(\alpha(g), \alpha_l)$, b is the last common vertex along $\pi(\alpha_j, \alpha(g))$ and $\pi(\alpha_j, \alpha_i)$, and c is the last common vertex along $\pi(\alpha_l, \alpha_i)$ and $\pi(\alpha_l, \alpha(g))$. If $\alpha(g) = a_j$ (resp. $\alpha(g) = a_i$) then $a = \alpha(g) = b$ (resp. $a = \alpha(g) = c$). If $g(\alpha_i)$ is a geodesic triangle (see figure 7), let a' denote the main apex of $g(\alpha_i)$. Note that $a' = nca(\alpha_j, \alpha_i)$ or $a' = nca(\alpha_i, \alpha_l)$. Furthermore, if $g(\alpha_i)$ is a geodesic quadrilateral then $a = nca(r_j, r_l)$. To update g, we need to compute $\pi(\alpha_j, \alpha_l)$ i.e., it is enough to compute $\pi(b, c)$.

If $g(\alpha_i)$ is a geodesic quadrilateral (see figure 5) we basically need to compute segment $\overline{yz}$ where $y \in \pi(\alpha_i, b) \cup \pi(b, a)$ and $z \in \pi(\alpha_i, c) \cup \pi(c, a)$ such that y and z are supporting and $\overline{yz}$ lies entirely in P (see figure 8). Then $\pi(b, c) = \pi(b, y) \cup \overline{yz} \cup \pi(z, c)$, where $\pi(b, y)$ and $\pi(c, z)$ are both known. There are four possible cases for segment $\overline{yz}$ (see figure 8). *Case 1:* $y \in \pi(\alpha_i, b)$ and $z \in \pi(\alpha_i, c)$ (figure 8(a)). *Case 2:* $y \in \pi(b, a)$ and $z \in \pi(\alpha_i, c)$, $y \neq b$ (figure 8(b)). *Case 3:* $y \in \pi(\alpha_i, b)$ and $z \in \pi(c, a)$, $z \neq c$ (figure 8(c)). *Case 4:* $y \in \pi(b, a)$ and $z \in \pi(c, a)$, $y \neq b$, $z \neq c$ (figure 8(d)).

If $g(\alpha_i)$ is a geodesic triangle then $\pi(\alpha_j, \alpha_l)$ is already known. (Recall that $\pi(\alpha(g), a_j)$ and $\pi(\alpha(g), a_l)$ belong to the shortest path tree from s_1). If $a' = nca(\alpha_j, \alpha_i)$, let $z = a'$ and let y be the vertex following z along $\pi(a', b)$. Segment $\overline{yz}$ is supporting to both $\pi(a', b)$ and $\pi(a', \alpha_i)$. Since $y \in \pi(a, b)$ and $z \in \pi(\alpha_i, c)$, this can be regarded as case (2) for $y \neq b$ or case (1) for $y = b$. If $a' = nca(\alpha_l, \alpha_i)$, let $y = a'$ and z be the following vertex along $\pi(a', c)$. Since $y \in \pi(b, \alpha_i)$ and $z \in \pi(c, a)$ this can be regarded as case (3) for $z \neq c$ or case (1) for $z = c$.

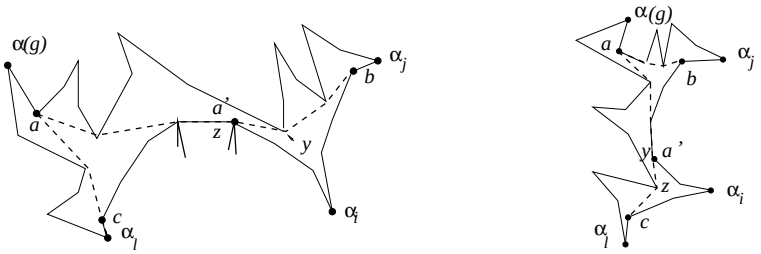

Fig. 7. $g(\alpha_i)$ is a geodesic triangle.

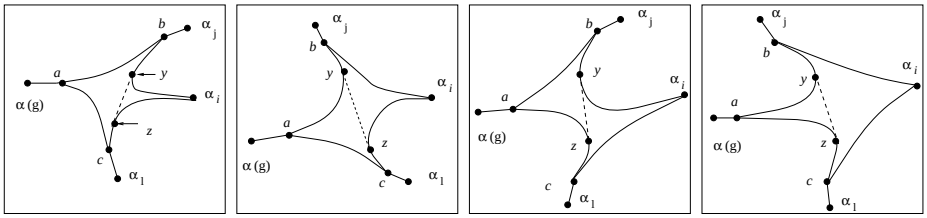

Fig. 8. The four cases for segment $\overline{yz}$.

To determine which of the four cases is the one occurring and determine vertices y and z, we advance a variable vertex v along $\pi(\alpha_i, b) \cup \pi(b, a)$ and a variable vertex u along $\pi(\alpha_i, c) \cup \pi(c, a)$ until $v = y$ and $u = z$ (v and u start at α_i). The problem is that vertices a, b and c as well as apexes α_j and α_l and $\alpha(g)$ are not known in advance. However, useful information can be obtained while advancing v and u using the shortest path tree from s_1. In particular, we can determine that $v = b$ (resp. $u = c$) due to the following property: $v = b$, for $v \in \pi(\alpha_i, \alpha_j)$, (resp. $u = c$, $u \in \pi(\alpha_i, \alpha_l)$) if and only if the predecessor of v (resp. u) on the shortest path from s_1, denoted as $pred(v)$, is supporting with respect to $\overline{pred(v)v}$ (resp. $\overline{pred(u)u}$) and $pred(v) \notin \pi(\alpha_i, \alpha_j)$ (resp. $pred(u) \notin \pi(\alpha_i, \alpha_l)$). Vertex a needs to be known only in case 4 in which case $a = nca(r_j, r_l)$. Segment $\overline{vu}$ is advanced so that it always lies entirely in the interior of $g(\alpha_i)$. This can be easily enforced due to the following property: $\overline{vu}$ intersects $\pi(a, c)$ (resp. $\pi(a, b)$) if and only if $pred(u)$ (resp. $pred(v)$) lie at the same side of $\overline{vu}$ as α_i. If at any point during the advancement it is determined that $\overline{vu}$ intersects $\pi(a, c)$ (resp. $\pi(a, b)$), we have case 3 or case 4 (resp. case 2 or 4).

After segment $\overline{yz}$ has been determined the update of g can be briefly stated as follows: *Case 1*: Remove $\pi(y, \alpha_i) \cup \pi(\alpha_i, z)$ and add $\pi = \overline{yz}$. *Case 2*: Remove $\pi(b, \alpha_i) \cup \pi(\alpha_i, z)$ and add $\pi = \pi(b, y) \cup \overline{yz}$. *Case 3*: Remove $\pi(c, \alpha_i)\pi(\alpha_i, y)$ and add $\pi = \pi(c, z) \cup \overline{yz}$. *Case 4*: Remove $\pi(b, \alpha_i)\pi(\alpha_i, c)$ and add $\pi = \pi(b, y) \cup \overline{yz} \cup \pi(z, c)$. To update g in any of the four cases we walk along the corresponding $\pi \subseteq \pi(b, c)$ starting at vertices b or c. (In case 4 we walk along $\pi_1 = \pi(b, y)$ and $\pi_2 = \pi(c, z)$ and then consider segment $\overline{yz}$.) Figure 6 illustrates the update of g for case 3 i.e., $\pi = \pi(c, y)$. Note that edge $\overline{y_2 y_3}$ and vertex y_4 become geodesic links.

The edges visited by the algorithms to determine $\overline{yz}$ and update g are either deleted from g or they lie along $\pi \subseteq \pi(b, c)$. For each edge visited only constant time is spent. Edges along π are either new white edges to be added to the geodesic decomposition or they become part of a geodesic link. Once an edge joins a geodesic link at some phase, it is never visited again until it gets output or deleted from the geodesic decomposition. Thus over all phases, the time spent for updates of the geodesic decomposition is proportional to the total number of edges appearing in the geodesic decomposition throughout the algorithm. Since no two such edges intersect and since they are always incident to vertices or points in R, their number is bounded by $O(n + k)$.

To update the white list, let π' be the path obtained from π by substituting any maximal sequence of red edges by a single red link. Let h_1 and h_2 be the vertices in the white-list preceding and following (or coinciding) the endpoints of π respectively. (All vertices in the white-list between h_1 and h_2 got deleted from γ during the update.) Delete the part of the white list between h_1 and h_2 and merge π'. The time spent is proportional to the number vertices deleted plus the size of π. Thus,

Lemma 3. *The total time spent for updating the geodesic decomposition and the white-list in every phase of the algorithm is $O(n + k)$.*

To complete the update, we need to compute $nca(r_j, r_l)$ since r_j and r_l become neighbors after the deletion of r_i. Clearly $nca(r_j, r_l)$ is the one between $nca(r_j, r_i)$ and $nca(r_i, r_l)$ that occurs first along $\pi(s_1, r_i)$ i.e., the one nearest to the root. Assigning a level number to all nodes of the shortest path tree from s_1 allows to compute $nca(r_j, r_l)$ from $nca(r_j, r_i)$ and $nca(r_i, r_l)$ in constant time.

After the geodesic decomposition induced by R^q has been computed, we need to extract white edges of $\pi(s_i, t_i)$ for every pair (s_i, t_i) at level $(h - q + 1)$ of T_{sl}, add them to $\mathcal{E}$, color them red, and update the white list. Starting at $\alpha(s_i)$ we follow the white list until $\alpha(t_i)$, adding the encountered white edges to $\mathcal{E}$. Note that white edges along $\pi(s_i, t_i)$ that are part of geodesic links appear twice along the white-list and that both occurrences must become red. Finally, substitute $\pi(\alpha(s_i), \alpha(t_i))$ by a single red link between $\alpha(s_i)$ and $\alpha(t_i)$. The time spent is proportional to the number of white edges that get output to $\mathcal{E}$ i.e., become red.

We therefore conclude that $\mathcal{E}$ can be computed in $O(n + k)$ time.

4 Concluding Remarks

We have given a simple linear time algorithm for the k-pairs non-crossing shortest path problem in a simple polygon P. Similarly, to [12, 11, 10], this result can be extended to a simple polygon P with one hole Q where source-destination pairs may appear on both ∂P and ∂Q. The main observation is that once a path π_i between $s_i \in \partial P$ and $t_i \in \partial Q$ has been routed there is only one way (if any) to route the rest of the pairs. Furthermore, π_i can be routed in only two ways: clockwise and counterclockwise around Q.

If P is a set of rectangles enclosed by an outer rectangle and points in $\mathcal{ST}$ appear on the outer rectangle and one inner rectangle the shortest non-crossing

path problem can be solved in $O((n+k)\log(n+k))$ time as shown in [11, 10] where n is the number or rectangles and k is the number of pairs. Note however that k can be arbitrarily large compared to n. Our observation is that k need not play a role in the main part of the algorithm. The idea is as follows: Given $\mathcal{ST}$ and the set of rectangles, derive a set $\mathcal{ST}'$ of $O(\min\{n,k\})$ pairs, solve the problem for $\mathcal{ST}'$ using the algorithm of [11] in $O(n\log n)$ time, and modify the solution in additional $O(k)$ time to derive the set of non-crossing paths for $\mathcal{ST}$. In particular, consider the partition of the outer rectangle into intervals derived by the vertical and horizontal *valid* projections of the inner rectangles. (Valid projection lines do not intersect obstacles.) For $(s_i,t_i) \in \mathcal{ST}$, let I_{s_i} and I_{t_i} denote the intervals on the outer rectangle containing s_i and t_i respectively. Let a,b and c,d be the endpoints of I_{s_i} and I_{t_i}. Then $\mathcal{ST}' = \{(I_{s_i}, I_{t_i}), (s_i,t_i) \in \mathcal{ST}\}$, where $(I_{s_i}, I_{t_i}) = \{(a,c),(a,d),(b,c),(b,d)\}$. Computing $\pi(s_i,t_i)$ for $(s_i,t_i) \in \mathcal{ST}$ in [11] can now be substituted by computing $\pi(a,c), \pi(a,d), \pi(b,c)$, and $\pi(b,d)$. The details are left for the full paper.

Acknowledgment. I would like to thank professor D.T. Lee for many helpful discussions and comments.

References

1. B. Chazelle, "A theorem on polygon cutting with applications," *Proc. 23rd Annu. IEEE Sympos. Found. Comput. Sci.*, 1982, pp. 339–349.
2. B. Chazelle, H. Eddelsbrunner, M. Grigni, L. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink, "Ray shooting in polygons using geodesic triangualtions", *Algorithmica*, 12, 1994, 54-68.
3. M. T. Goodrich and R. Tamassia, "Dynamic Ray Shooting and Shortest Paths via Balanced Geodesic Triangualtions", *In Proc. 9th Annu. ACM Sympos. Comput. Geom*, 1993, 318-327.
4. L.J. Guibas and J. Hershberger, "Optimal shortest path queries in a simple polygon", *J. Comput. Syst. Sci.*, 39, 1989, 126-152.
5. 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.
6. J. Hershberger and J. Snoeyink, "Computing Minimum Length Paths of a given homotopy class", *Comput. Geometry: Theory and Applications*, 4, 1994, 63-97.
7. D.T. Lee, "Non-crossing paths problems", Manuscript, Dept. of EECS, Northwestern University, 1991.
8. D. T. Lee and F. P. Preparata, "Euclidean Shortest Paths in the Presence of Rectilinear Barriers", *Networks*, 14 1984, 393-410.
9. F.P. Preparata and M. I. Shamos, *Computational Geometry: an Introduction*, Springer-Verlag, New York, NY 1985.
10. J. Takahashi, H. Suzuki, and T. Nishizeki, "Finding shortest non-crossing rectilinear paths in plane regions", *Proc. of ISAAC'93, Lect. Notes in Computer Science, Spinger-Verlag*, 762, 1993, 98-107.
11. J. Takahashi, H. Suzuki, and T. Nishizeki, "Shortest non-crossing rectilinear paths in plane regions", *Algorithmica*, to appear.
12. J. Takahashi, H. Suzuki, and T. Nishizeki, "Shortest non-crossing paths in plane graphs", *Algorithmica*, to appear.

This article was processed using the LaTeX macro package with LLNCS style