Constructing Pairwise Disjoint Paths with Few Links

Himanshu Gupta* Rephael Wenger†

February 26, 1997

Abstract

Let P be a simple polygon and let $\{(u_i, u_i')\}$ be m pairs of distinct vertices of P where for every distinct $i, j \leq m$, there exist pairwise disjoint paths connecting u_i to u_i' and u_j to u_j'. We wish to construct m pairwise disjoint paths in the interior of P connecting u_i to u_i' for $i = 1, \ldots, m$, with minimal total number of line segments. We give an approximation algorithm which in $O(n \log m + M \log m)$ time constructs such a set of paths using $O(M)$ line segments where M is the number of line segments in the optimal solution.

1 Introduction

Let P be a simple polygon and let u and u' be two distinct vertices of P. The *link distance* from u to u' is the minimum number of line segments (also called *links*) required to connect u to u' by a polygonal path lying in P. A polygonal path which uses the minimum number of required line segments is called a *minimum link path*. Clearly, there may be more than one such path. Suri in [10] gave a linear time algorithm for determining the link distance and a minimal link path between two vertices.

A minimum link path from u to u' may intersect the boundary of P at many points other than u and u'. The *interior link distance* from u to u' is the minimum number of line segments required to connect u to u' by a polygonal path through the interior of P. Such a path which uses the minimum number of required line segments is called a *minimum link interior path*. The interior link distance from u to u' may differ greatly from the link distance between the two points. (See Figure 1.)

Let u_1, u_1', u_2, u_2' be four vertices lying in the given order around P. By virtue of the relative locations of these four vertices, there are nonintersecting paths, ζ_1 and ζ_2, connecting u_1 to u_1' and u_2 to u_2', respectively. However, it is

*Stanford University, Stanford CA 94305 (hgupta@cs.stanford.edu).

†Ohio State University, Columbus, OH 43210, U.S.A. (wenger@cis.ohio-state.edu). Supported by NSA grant MDA904-93-H-3026.

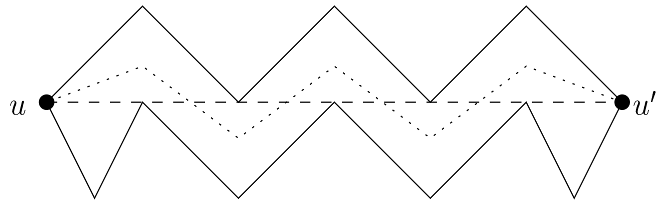


Figure 1: Minimum link and minimum interior link paths.

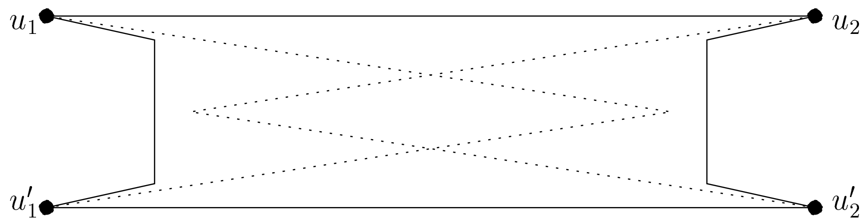


Figure 2: Intersecting Link Paths

possible that every minimum interior link path connecting u_1 to u'_1 intersects every minimum interior link path connecting u_2 to u'_2. (See Figure 2.) To simultaneously connect u_1 to u'_1 and u_2 to u'_2 by nonintersecting interior paths requires more line segments. In general, two additional line segments suffice to construct two such nonintersecting interior paths. (See [5].)

A set $\Pi = \{(u_i, u'_i)\}$, $i \leq m$, of m pairs of distinct vertices of P is *untangled* if some set of pairwise disjoint paths connects u_i to u'_i. Let $\Pi = \{(u_i, u'_i)\}$, $i \leq m$, be an untangled set of m pairs of distinct vertices of P. Let $l(u_i, u'_i)$ be the interior link distance from u to u' and let $L = \sum_{i=1..m} l(u_i, u'_i)$ be the sum of those distances. Clearly, L line segments are required to construct a set of pairwise disjoint interior paths connecting u_i to u'_i, for $i = 1, \ldots, m$. How many additional line segments are required? In [5] we proved that $O(m \log m)$ additional line segments suffice and claimed without proof that $\Omega(m \log m)$ additional line segments may be required. In Section 2 of this paper we prove the lower bound, presenting a construction which requires $\Omega(m \log m)$ line segments.

We define the *pairwise disjoint link paths problem* as: given an untangled set, $\{(u_i, u'_i)\}$, of m pairs of distinct vertices of P, find the minimum total number of line segments required by a set of pairwise disjoint interior paths connecting u_i to u'_i. We were unable to give a polynomial time algorithm for this problem or to determine if the problem is NP-complete. Instead we present an algorithm in Sections 3 and 4 which finds a solution within a constant factor of the optimal solution.

A triangulation T_P of P (possibly with interior vertices) is *isomorphic* to a triangulation T_Q of Q if there is a one-to-one, onto mapping f between the vertices of T_P and the vertices of T_Q such that p, p', p'' are vertices of a triangle in T_P if and only if $f(p), f(p'), f(p'')$ are vertices of a triangle in T_Q. An isomorphic triangulation of P and Q also defines a piecewise linear homeomorphism between P and Q. The size of a triangulation is the total number of vertices, edges and triangles in the triangulation.

Algorithms for constructing isomorphic triangulations between labelled point sets are described in [8] and [9]. Algorithms for isomorphic triangulations between simple polygons are given in [1], [7] and [5]. Our motivation for constructing pairwise disjoint paths using few line segments comes from our work on an approximation algorithm for constructing isomorphic triangulations between simple polygons in [5]. The main result in this paper improves the output size of that algorithm from $O(M_1 \log n + n \log^2 n)$ to $O(M_1 \log n)$ where n is the input size and M_1 is the size of the optimal solution. In Section 5 we outline this improvement.

2 Worst Case Bounds

In this section we construct polygons with n vertices, $(n+1)/8$ pairs of vertices, and interior link distance three, such that any set $\{\zeta_i\}$ of nonintersecting paths connecting the pairs requires a total of $\Omega(n \log n)$ links.

For each integer k, define functions f and g on the integers as follows. Each integer $a < 2^k$ can be represented in binary as the sum of powers of two. For each $a = \sum_{i=0..k-1} \alpha_i 2^i$, $\alpha_i \in \{0,1\}$, define $f(a) = \sum_{i=1..k-1} \alpha_i 8^i$ and $g(a) = 2^{k-1} \sum_{i=0..k-1} (1-\alpha_i) 4^i$. (Defining $g(a) = \sum_{i=0..k-1} (1-\alpha_i) 4^i$ will work just as well but the 2^{k-1} factor generates a more symmetric polygon.) Note that in the definition of f, the sum is from 1 to $k-1$, not from 0 to $k-1$. Thus $f(a) = f(a+1)$ when a is even. Let p_i be the point $(0, f(i))$ on the y-axis and let q_i be the point $(1, g(i))$ on the vertical line $x = 1$. Let l_i be the line segment from p_i to q_i. (See Figure 3.)

Let $L = \{l_i : i = 0, \ldots, 2^k - 1\}$. Let l^* be the line segment joining $(1, g(0))$ and $(1, g(2^k - 1))$. Let P be the polygon which is union of all the bounded regions in the arrangement of $L \cup \{l^*\}$. (See Figure 4.) Clearly every point in P can be connected to l^* by a line segment in the interior of P. Thus every pair of points in P can be connected by three line segments in the interior of P and P has interior link distance three.

Polygon P has 2^{k-1} vertices on the y-axis and two vertices on the vertical line $y = 1$. We will show that the total number of vertices n equals $2^{k+1} - 1$. The vertices of P on the y-axis are the points $\{p_0, p_2, \ldots, p_{2^k-2}\}$. (Remember $p_i = p_{i+1}$ for i even.) Pair each vertex p_{2i} with the vertex p_{2^k-2i}, constructing $(n+1)/8$ pairs. We claim that any set of non-intersecting paths connecting these pairs requires at least $(1/32) n \log_2 n$ links when k is even.

We first show that the relative position of the intersection points of two lines l_a and l_b can be determined by the most significant bit in which a and b differ. More specifically, if the most significant bit in which a and b differ is less than the most significant bit in which a' and b' differ then $l_a \cap l_b$ is to the left of

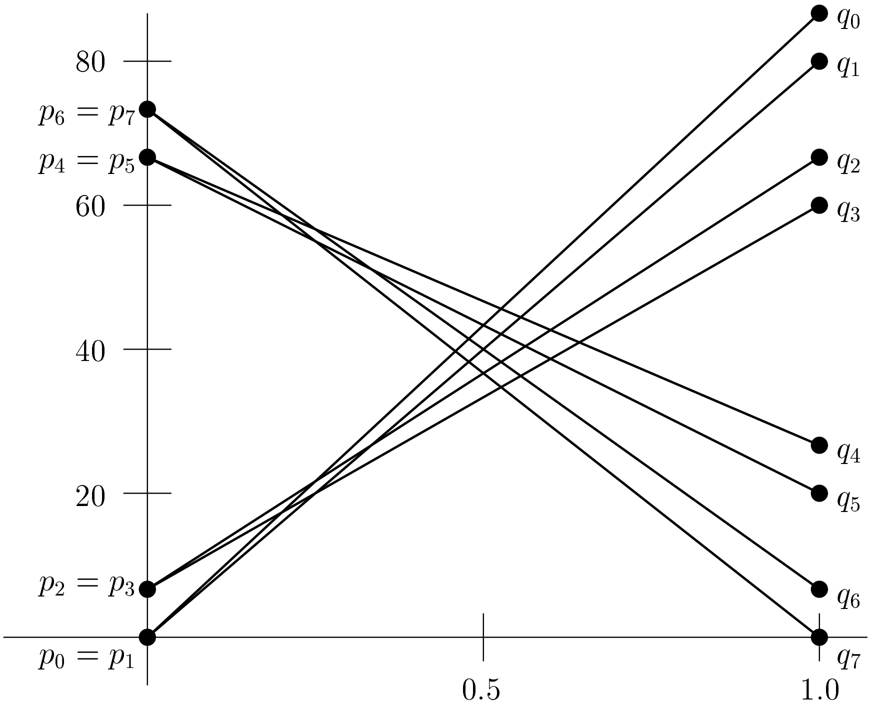


Figure 3: Line segments l_i, $i = 0, \ldots, 7$.

$l_{a'} \cap l_{b'}$.

The intersection of segment l_a and l_b has x-coordinate

$$\frac{1}{1 + \frac{g(b)-g(a)}{f(a)-f(b)}},$$

whenever $f(a) - f(b) \neq 0$. If $a = \sum_{i=0..k-1} \alpha_i 2^i$, $\alpha_i \in \{0,1\}$, and $b = \sum_{i=0..k-1} \beta_i 2^i$, $\beta_i \in \{0,1\}$, then

$$\begin{aligned} f(a) - f(b) &= \sum_{i=1}^{k-1} (\alpha_i - \beta_i) 8^i, \\ g(b) - g(a) &= 2^{k-1} \sum_{i=0}^{k-1} (\alpha_i - \beta_i) 4^i. \end{aligned}$$

If $j > 0$ is the most significant bit in which a and b differ, then

$$\frac{g(b)-g(a)}{f(a)-f(b)} \leq 2^{k-1} \frac{4^j + 4^{j-1} + 4^{j-2} + \cdots + 4 + 1}{8^j + 8^{j-1} + 8^{j-2} + \cdots + 8} = 2^{k-1} \frac{7}{3} \left(\frac{4^{j+1} - 1}{8^{j+1} - 8} \right)$$

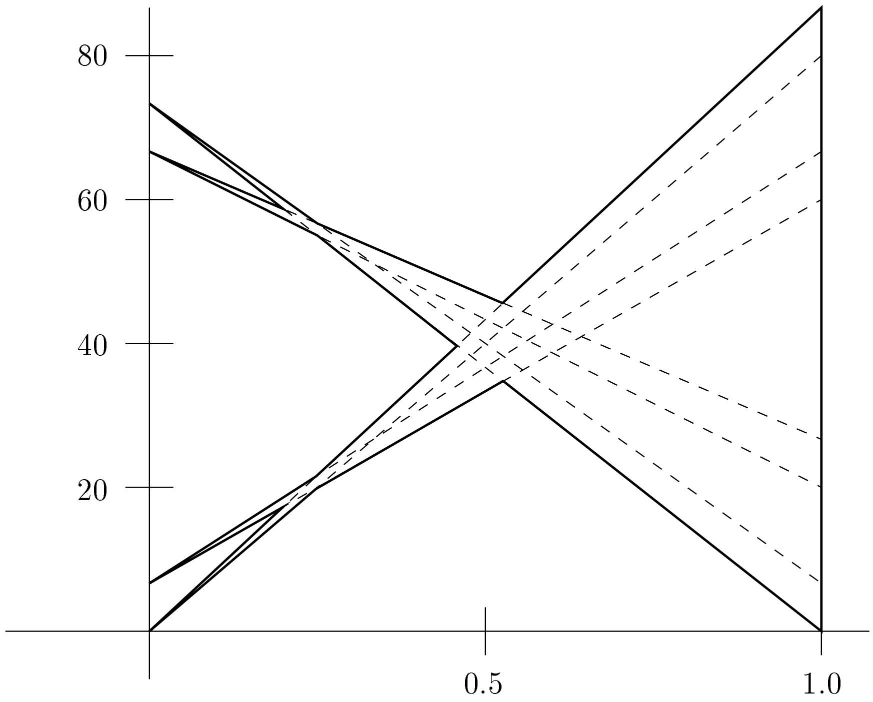


Figure 4: Weakly edge visible polygon.

and

$$\frac{g(b)-g(a)}{f(a)-f(b)} \geq 2^{k-1}\frac{4^j-4^{j-1}-4^{j-2}-\cdots-4-1}{8^j-8^{j-1}-8^{j-2}-\cdots-8} = 2^{k-1}\frac{7}{3}\left(\frac{2\times 4^j+1}{6\times 8^j+8}\right).$$

Thus,

$$2^{k-1}\frac{7}{3}\left(\frac{2\times 4^j+1}{6\times 8^j+8}\right) \leq \frac{g(b)-g(a)}{f(a)-f(b)} \leq 2^{k-1}\frac{7}{3}\left(\frac{4\times 4^j-1}{8\times 8^j-8}\right).$$

Assume $j>0$ is the most significant bit in which a and b differ, while $j+1$ is the most significant bit in which a' and b' differ.

$$\frac{g(b')-g(a')}{f(a')-f(b')} \leq 2^{k-1}\frac{7}{3}\left(\frac{4\times 4^{j+1}-1}{8\times 8^{j+1}-8}\right) < 2^{k-1}\frac{7}{3}\left(\frac{2\times 4^j+1}{6\times 8^j+8}\right) \leq \frac{g(b)-g(a)}{f(a)-f(b)}.$$

Thus

$$\frac{1}{1+\frac{g(b')-g(a')}{f(a')-f(b')}} > \frac{1}{1+\frac{g(b)-g(a)}{f(a)-f(b)}},$$

and the intersection point of l_a and l_b is to the left of the intersection point of $l_{a'}$ and $l_{b'}$. This is also true if a and b differ only in the zero'th bit, since then

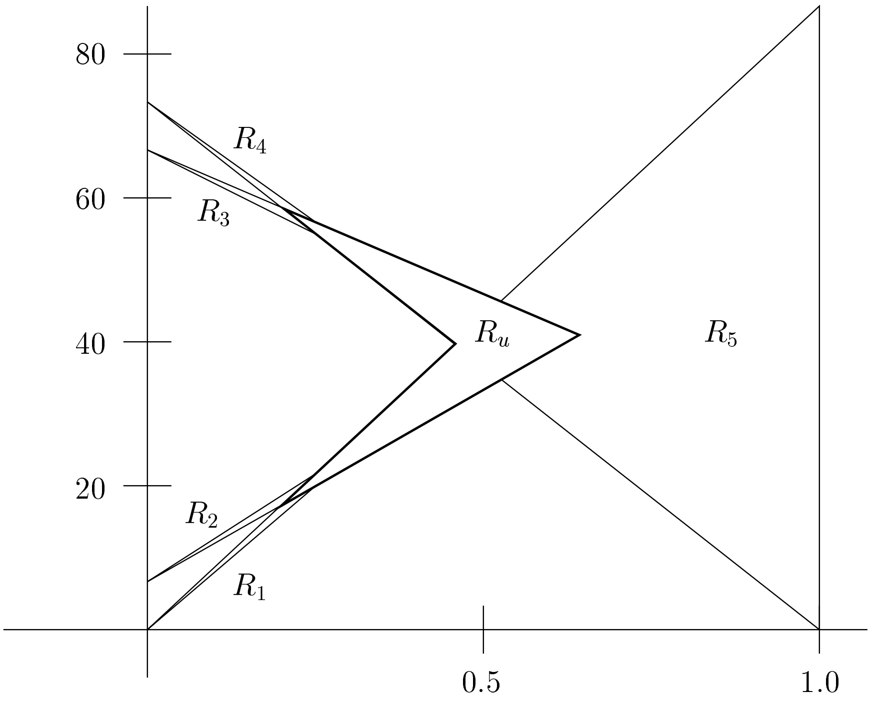


Figure 5: Polygon R_u.

their intersection lies on the x-axis. Thus the relative position of $l_a \cap l_b$ and $l_{a'} \cap l_{b'}$ can be determined by the most significant bits in which a and b and a' and b' differ.

The line segments $\{l_a\}$ can be represented as the leaves in a completely balanced binary tree $\mathcal{T}$, where the leaves are ordered by the values of a. Each internal node u in the tree represents the set $\mathcal{I}_u$ of intersections of the line segments in the left subtree with the line segments of the right subtree. For the 2^{k-1} nodes at height one, each $\mathcal{I}_u$ contains a single intersection point and these 2^{k-1} intersection points are the 2^{k-1} vertices of polygon P on the y-axis. For the $2^{k-1}-1$ nodes of height greater than one, note that the intersection points of all pairs of line segments in the left subtree and all pairs of segments in the right subtree lies to the left of any point in $\mathcal{I}_u$. Thus the "lowest" and "highest" line segments from the left subtree and the "lowest" and "highest" line segments from the right subtree bound all the points in $\mathcal{I}_u$. The intersection of these four segments gives four points in $\mathcal{I}_u$ whose convex hull is a diamond containing $\mathcal{I}_u$. The three leftmost vertices of this diamond are vertices of P. Thus the 2^{k-1} nodes at height greater than one generate $3 \times 2^{k-1} - 3$ polygon vertices. Polygon P has two more vertices on the line $y = 1$, so the total number n of vertices of P is $2^{k+1} - 1$.

For each internal node u in $\mathcal{T}$, again consider all the intersection points of the left and right subtrees of u. For each node u, let q_u^l and q_u^r be the leftmost and rightmost such intersection points. If u has height one, then $q_u^l = q_u^r$ is a vertex p_{2i} of polygon P. For each node u of height greater than one, let R_u be simple polygon with vertices $\{q_{u.left}^l, q_u^r, q_{u.right}^l, q_u^l\}$, where $u.left$ and $u.right$ are the left and right subtrees of u, respectively (Figure 5.) Let h be the height of u. Let $\{\zeta_i\}$ be a set of pairwise disjoint paths connecting p_{2i} to p_{2^k-2i} for $i = 0, \ldots, 2^{k-2} - 1$. We claim that R_u must contain at least 2^{h-3} vertices of $\{\zeta_i\}$.

Assume that the height of u is greater than two. Deleting polygon R_u from polygon P divides P into five pieces, four to the left and one to the right. Label the regions to the left R_1, R_2, R_3, R_4, from bottom to top, and the region to the right R_5. Any line segment in P with endpoints in R_1 and R_5 must intersect any line segment in P with endpoints in R_4 and R_5. Thus if ζ_i and ζ_j are pairwise disjoint paths connecting R_1 and R_4 to R_5, respectively, then either ζ_i or ζ_j has a vertex in R_u.

Assume u is not the root of $\mathcal{T}$. (Note that u is the root of $\mathcal{T}$ in Figure 5.) If ζ_i has an endpoint in R_1, R_2, R_3 or R_4, then the other endpoint of ζ_i must be in R_5. Since 2^{h-3} paths have endpoints in R_1 and 2^{h-3} paths have endpoints in R_4, polygon R_u contains at least 2^{h-3} vertices.

Now assume u is the root of $\mathcal{T}$. If ζ_i has an endpoint in R_1, then the other endpoint of ζ_i must be in R_4 and vice versa. Since no line segment in P connects R_1 to R_4, polygon R_u must contain a vertex of each of the 2^{k-3} paths connecting R_1 to R_4. Similarly, R_u must contain a vertex of each of the 2^{k-3} paths connecting R_2 to R_3. Thus R_u contains at least 2^{k-2} path vertices.

If u is an internal node of height two, then two paths ζ_i and ζ_{i+1} must cross R_u. By reasoning similar to that given above, R_u contains a vertex (other than the endpoints) of at least one of these paths. Thus if u has height two, then R_u contains at least one vertex.

Assume k is a positive even integer. The polygons R_u where u has even height (greater than one) are pairwise disjoint. There are 2^{k-h} polygons R_u where u has height h and each such polygon contains at least 2^{h-3} path vertices. Together they contain 2^{k-3} path vertices. There are 2^{k-2} polygons R_u where u has height two and each such polygon contains at least one path vertex. Finally the polygon R_u where u is the root contains at least 2^{k-2} path vertices. Summing these values, we find that the polygons R_u where u has even height together contain at least $(k+4)2^{k-4} = (\log_2(n+1)+3)(n+1)/32$ path vertices. Thus the pairwise disjoint paths $\{\zeta_i\}$ contain at least $(1/32)n\log_2 n$ line segments. For odd values of k, a similar argument gives an $\Omega(n \log n)$ lower bound with a somewhat lower constant.

3 Approximation Algorithm

Point $p \in P$ is *visible* from point $p' \in P$ if P contains the open line segment (p, p'). Point p is *clearly visible* from point $p' \in P$ if the interior of P contains

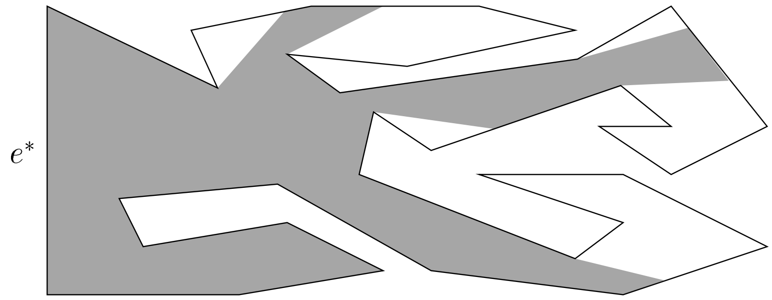


Figure 6: $\hat{\mathrm{V}}\mathrm{is}(e)$.

the open line segment (p, p'). The *(clear) visibility polygon* from a point $p \in P$ is the set of points in P (clearly) visible from p.

Point $p \in P$ is *(clearly) visible* from edge $e \in P$ if there is some point $p' \in e$ such that p is (clearly) visible from p'. (This definition of visibility is sometimes called *weak visibility* as opposed to *strong visibility* where p must be visible from every point $p' \in e$. Throughout this paper, visibility refers to weak visibility.) Edge e or triangle t is *(clearly) visible* from edge e' or triangle t' if there are points $p \in e$ or $p \in t$ and $p' \in e'$ or $p' \in t'$ such that p is (clearly) visible from p'.

The *(clear) visibility polygon* from a line segment $e \subseteq P$ is the set of points in P (clearly) visible from e. (See Figure 6.) We denote the visibility polygons as $\mathrm{Vis}(p)$ and $\mathrm{Vis}(e)$ and the clear visibility polygons as $\hat{\mathrm{V}}\mathrm{is}(p)$ and $\hat{\mathrm{V}}\mathrm{is}(e)$. Note that $\mathrm{Vis}(p)$ and $\mathrm{Vis}(e)$ are closed sets while $\hat{\mathrm{V}}\mathrm{is}(p)$ and $\hat{\mathrm{V}}\mathrm{is}(e)$ are usually not closed.

In [3], Chazelle gives a linear time algorithm for constructing a triangulation T_P of P (using no vertices other than those of P.) Using this triangulation, Guibas et al. in [4] give a linear time algorithm for constructing $\mathrm{Vis}(e)$. Their algorithm can be easily modified to construct $\hat{\mathrm{V}}\mathrm{is}(e)$.

The algorithm by Guibas et al. can also be modified to run in time $O(n_e)$ (after the $O(n)$ triangulation construction) where n_e is the number of triangles of T_P intersected by $\mathrm{Vis}(e)$. A brief outline of the necessary modifications follows. Edge or diagonal (w_2, w_3) intersects the visibility polygon $\mathrm{Vis}(e)$ of $e = (w_0, w_1)$ if and only if the shortest paths inside P from w_0 to w_2 and from w_1 to w_3 are outwardly convex or vice versa. (See [4] for the definition of outwardly convex and a complete discussion of the algorithm.) The algorithm by Guibas et al. constructs the shortest paths from w_0 and w_1 to each vertex of the polygon, checks for outward convexity, and then constructs $\mathrm{Vis}(e)$ from outward convex paths. Instead of initially constructing all shortest paths to all vertices, the modified algorithm checks for outward convexity and visible diagonals as the shortest paths are constructed. Whenever a diagonal (w_2, w_3) is found which is not visible to e, the entire region which is separated from e by (w_2, w_3) is not visible to e and does not need to be processed. Details are left to the reader. A

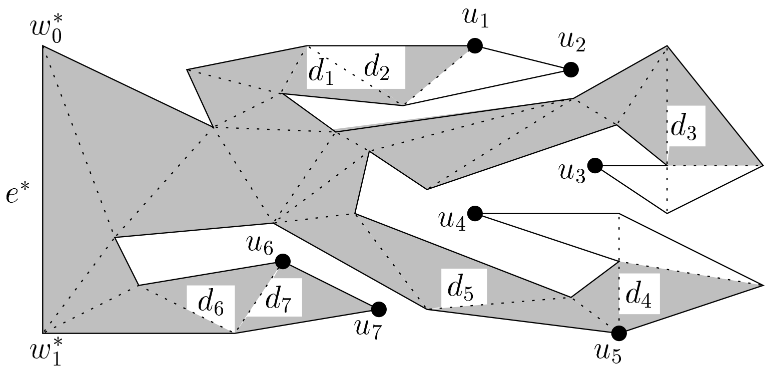


Figure 7: Γ_{e^*}.

similar modification will construct $\hat{\mathrm{V}}\mathrm{is}(e)$ in $O(n_e)$ time.

Again let T_P be a triangulation of polygon P using no vertices other than those of P. Let u and e be a vertex and an edge of P, respectively. Edge d of triangulation T_P *separates* u from e if every interior path from u to the interior of e must intersect the interior of d. Triangle t of triangulation T_P *separates* u from e if every interior path from u to the interior of e must intersect the interior of t.

Lemma 1 *Let P be a simple polygon on n vertices with distinguished edge $e^* = \{w_0^*, w_1^*\}$ and let $\mathcal{U}$ be a subset of $\mathrm{Vert}(P) \setminus \{w_0^*, w_1^*\}$ of size m. A set of m pairwise disjoint interior paths connecting the vertices in $\mathcal{U}$ to the interior of e^* can be constructed in $O(n \log m + M \log m)$ time using a total of at most $240M$ line segments where M is the minimum total number of line segments necessary to connect $\mathcal{U}$ to e^* by m pairwise disjoint paths.*

Proof: Let $u_1, u_2, \ldots, u_m$ be the points in $\mathcal{U}$ labeled in clockwise order around P starting at e^*. Construct a triangulation T_P of P. Let Γ_{e^*} be union of the triangles of T_P which are clearly visible from edge e^*. Γ_{e^*} is a simple polygon in P. (See Figure 7.)

For any point $p \in P$, let $d(p)$ be the diagonal of T_P lying on a triangle containing p and separating p from e^*. Similarly, if c is an edge of triangulation T_P, let $d(c)$ be the diagonal of T_P lying on a triangle containing c and separating c from e^*. If p or c lie on the triangle containing e^*, then let $d(p)$ or $d(c)$ equal e^*.

The boundary of Γ_{e^*} is composed of edges and chords of the original polygon P. Each $u_i \in \mathcal{U}$ is either in Γ_{e^*} or is separated from Γ_{e^*} by some chord c of P on the boundary of Γ_{e^*}. If u_i is in Γ_{e^*}, let d_i be $d(u_i)$. If u_i is separated from Γ_{e^*} by chord c on the boundary of Γ_{e^*}, let d_i equal $d(c)$.

Since each diagonal d_i is clearly visible from edge e^*, the edge e^* is clearly visible from each diagonal d_i. Let s_i be $\mathrm{int}(e^*) \cap \hat{\mathrm{V}}\mathrm{is}(d_i)$. ($\mathrm{int}(e^*)$ is the interior

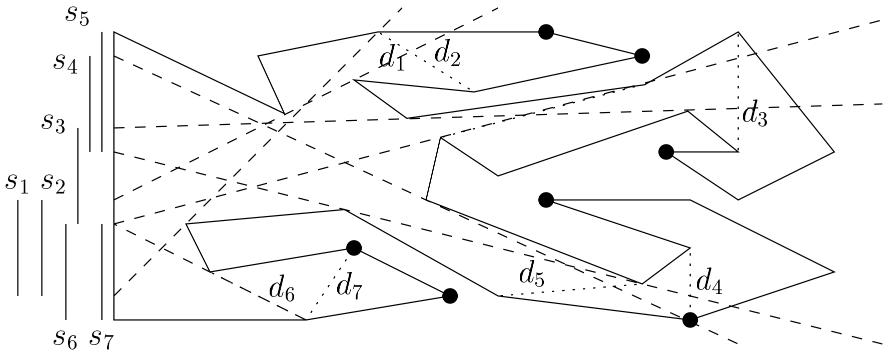
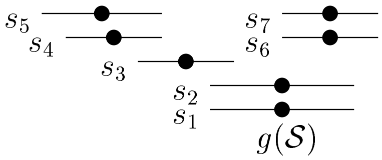


Figure 8: Line segments $s_i = \mathrm{int}(e^*) \cap \hat{\mathrm{V}}\mathrm{is}(d_i)$.



$$f(g(\mathcal{S}), \mathcal{S}) = 2,$$
$$f^-(g(\mathcal{S}), \mathcal{S}) = 3,$$
$$f^+(g(\mathcal{S}), \mathcal{S}) = 2.$$

Figure 9: $\mathcal{S}$, $g(\mathcal{S})$ and $f(g(\mathcal{S}), \mathcal{S})$.

of e^*.) Set s_i is an open line segment lying on e^*. Note that s_i may equal s_j (and d_i may equal d_j) for many distinct points $u_i, u_j \in \mathcal{U}$. (See Figure 8.)

For each u_i we wish to choose a point p_i on e^* to be the endpoint of the path from u_i to e^*. Obviously, a point in s_i is a good candidate since it can reach d_i with a single line segment. However, we also need to choose the p_i such that their order on e^* is consistent with the order of $\mathcal{U}$ around P. In other words, u_i, u_j, p_j, p_i should lie clockwise or counter-clockwise around P in the given order.

Let $\mathcal{S}$ be any set of open line segments in $\mathbf{R}^1$, not necessarily distinct. For each point $q \in \mathbf{R}^1$, let $f(q, \mathcal{S})$ be the number of line segments of $\mathcal{S}$ which contain the point q. Note that the line segments of $\mathcal{S}$ are open and do not contain their endpoints. Let $f^-(q, \mathcal{S})$ and $f^+(q, \mathcal{S})$ be the number of line segments of $\mathcal{S}$ contained in the open intervals $(-\infty, q)$ and (q, ∞), respectively. Note that $f(q, \mathcal{S}) + f^-(q, \mathcal{S}) + f^+(q, \mathcal{S})$ equals $|\mathcal{S}|$.

Let $\mathcal{R}$ be the set of midpoints of line segments of $\mathcal{S}$, again not necessarily distinct. The median point of $\mathcal{R}$ is the $\lceil |\mathcal{R}|/2 \rceil$'th point in $\mathcal{R}$, ordered from $-\infty$ to ∞. Let $g(\mathcal{S})$ be this median point of $\mathcal{R}$. At least $|\mathcal{R}|/2 = |\mathcal{S}|/2$ points of $\mathcal{R}$

lie in each of the closed intervals $(-\infty, g(\mathcal{S})]$ and $[g(\mathcal{S}), \infty)$. If the midpoint of segment $s \in \mathcal{S}$ lies in $(-\infty, g(\mathcal{S})]$, then segment s either contains $g(\mathcal{S})$ or lies in the open interval $(-\infty, g(\mathcal{S}))$. Thus $f(g(\mathcal{S}), \mathcal{S}) + f^-(g(\mathcal{S}), \mathcal{S})$ is greater than or equal to $\lceil |\mathcal{S}|/2 \rceil$. Similarly $f(g(\mathcal{S}), \mathcal{S}) + f^+(g(\mathcal{S}), \mathcal{S})$ is greater than or equal to $\lceil |\mathcal{S}|/2 \rceil$. (See Figure 9.)

Now consider two sets of line segments $\mathcal{S}_0$ and $\mathcal{S}_1$ on $\mathbf{R}^1$ and let $\mathcal{S} = \mathcal{S}_0 \cup \mathcal{S}_1$. Define

$$\mathcal{F}(\mathcal{S}_0, \mathcal{S}_1) = f(g(\mathcal{S}), \mathcal{S}) + f^+(g(\mathcal{S}), \mathcal{S}_0) + f^-(g(\mathcal{S}), \mathcal{S}_1).$$

Without loss of generality, assume that w_0^*, e^*, w_1^* appear in counter-clockwise order around P. Let $\mathcal{S}_\mathcal{U}$ be the sequence $(s_1, s_2, \ldots, s_m)$. Embed e^* and the line segments $s_i \in \mathcal{S}$ in the real line $\mathbf{R}^1$, mapping w_0^* to zero and w_1^* to one. In the next section, we describe an algorithm to partition $\mathcal{S}_\mathcal{U}$ into contiguous subsequences $\sigma_1 = (s_1, s_2, \ldots, s_{i_1})$, $\sigma_2 = (s_{i_1+1}, s_{i_1+2}, \ldots, s_{i_2}), \ldots$, $\sigma_{2h} = (s_{i_{2h-1}+1}, s_{i_{2h-1}+2}, \ldots, s_m)$, such that:

1. $g(\sigma_1 \cup \sigma_2) \leq g(\sigma_3 \cup \sigma_4) \leq \cdots \ldots \leq g(\sigma_{2h-1} \cup \sigma_{2h})$;
2. $|\sigma_{2j-1}| = |\sigma_{2j}|\ \ (+1)$ for $1 \leq j \leq h$;
3. $\sum_{j=1..h} \mathcal{F}(\sigma_{2j-1}, \sigma_{2j}) \geq m/40$.

(One possible partition of the segments in Figure 8 is $\sigma_1 = \{s_1, s_2, s_3\}$, $\sigma_2 = \{s_4, s_5\}$, $\sigma_3 = \{s_6\}$, $\sigma_4 = \{s_7\}$.)

Let g_j equal $g(\sigma_{2j-1} \cup \sigma_{2j})$ for $j = 1, \ldots, h$. Note that $g_1, g_2, \ldots, g_h$ lie in counter-clockwise order around P. Let $\mathcal{U}_j = \{u_i : s_i \in \sigma_j\}$ be the points in $\mathcal{U}$ corresponding to the line segments in σ_j for $j = 1, \ldots, h$. For each g_j, let Γ_j be the union of the triangles of T_P which intersect $\hat{\text{Vis}}(g_j)$ and separate some $u \in \mathcal{U}_{2j-1} \cup \mathcal{U}_{2j}$ from e^*. (See Figure 10.) Let Γ be the union of all the Γ_j. Similar to Γ_{e^*}, Γ is also a simple polygon in P, its boundary is composed of edges and chords of P, and it has a triangulation T_Γ induced by the triangulation T_P of P.

Let $\mathcal{C}$ be the set of chords of P bounding Γ. Each chord $c \in \mathcal{C}$ separates P into two subpolygons. Let P_c be the subpolygon not containing Γ. Let w_0^c and w_1^c be the endpoints of c. For each chord $c \in \mathcal{C}$, let $\mathcal{U}_c$ be the points of $\mathcal{U} \setminus \{w_0^c, w_1^c\}$ in P_c. Recursively, construct pairwise disjoint paths connecting the points in $\mathcal{U}_c$ to c. (See Figure 12.)

Let $\mathcal{D}$ be the diagonals of T_Γ which separate a single triangle of T_Γ from e^*. Label these diagonals $\tilde{d}_1, \tilde{d}_2, \ldots, \tilde{d}_{\tilde{n}}$ in counter-clockwise order around Γ. For each diagonal $\tilde{d}_k$ where $k < \tilde{n}$, there is a unique triangle $\tilde{t}_k$ such that $\tilde{t}_k$ separates $\tilde{d}_k$ from $\tilde{d}_{k+1}$ and $\tilde{d}_k$ from e^* and $\tilde{d}_{k+1}$ from e^*. Let $\tilde{w}_k$ be the vertex of $\tilde{t}_k$ which $\tilde{t}_k$ separates from e^*. Vertex $\tilde{w}_k$ lies between the endpoints of $\tilde{d}_k$ and $\tilde{d}_{k+1}$. For convenience, label vertices w_0^* and w_1^* as $\tilde{w}_0$ and $\tilde{w}_k$, respectively. (See Figure 13.)

Let ψ_k be the set of vertices of $\mathcal{U}$ between $\tilde{w}_{k-1}$ and $\tilde{w}_k$, possibly including $\tilde{w}_{k-1}$ but not $\tilde{w}_k$. For each diagonal $\tilde{d}_k \in \mathcal{D}$, choose the minimum j such that $\tilde{d}_k$ is an internal chord of Γ_j. Note that $\tilde{d}_k$ is clearly visible from g_j. Construct $|\psi_k|$ distinct line segments in the interior of P connecting g_j to the interior

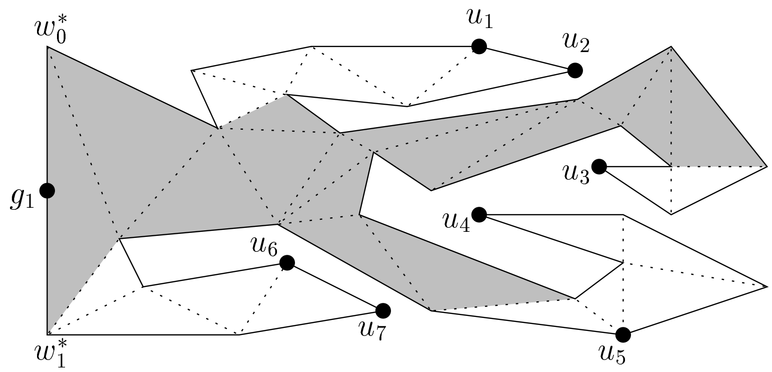


Figure 10: Γ_1.

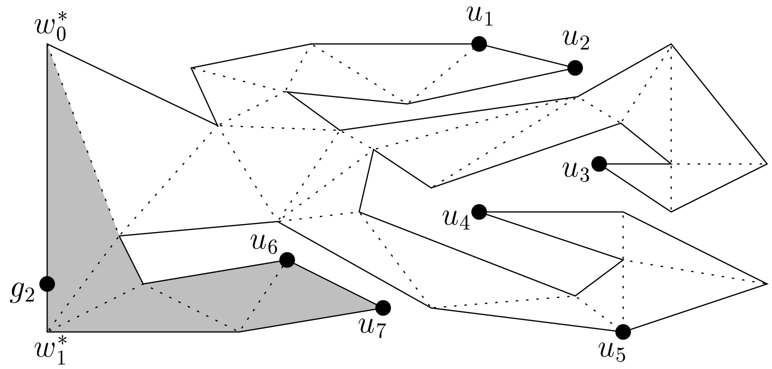


Figure 11: Γ_2.

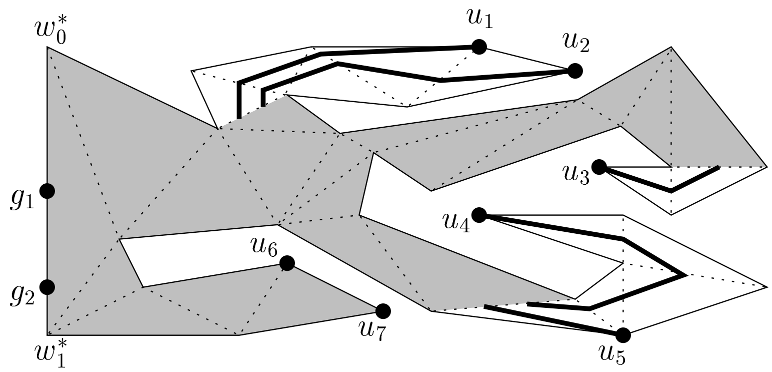


Figure 12: Γ, triangulation T_Γ and paths to the boundary of Γ.

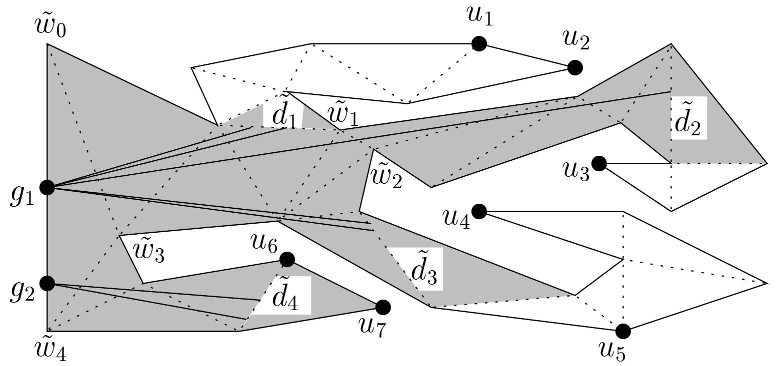


Figure 13: Diagonals $\tilde{d}_k$, vertices $\tilde{w}_k$ and line segments λ_i.

of $\tilde{d}_k$. A total of m line segments are constructed. Label these line segments $\lambda_1, \lambda_2, \ldots, \lambda_m$, as their endpoints appear in clockwise order around P starting at w_0^*. Note that if u_i is an element of ψ_k, then λ_i has an endpoint on $\tilde{d}_k$.

We claim that the line segments λ_i intersect only at their endpoints. Let λ_i and $\lambda_{i'}$ be line segments connecting distinct g_j and $g_{j'}$ to diagonals $\tilde{d}_k$ and $\tilde{d}_{k'}$, respectively. Without loss of generality, assume that j is less than j'. Diagonal $\tilde{d}_k$ is an internal chord of Γ_j, so diagonal $\tilde{d}_k$ separates some point of $\mathcal{U}_j$ from e^*. Similarly, diagonal $\tilde{d}_{k'}$ separates some point of $\mathcal{U}_{j'}$ from e^*. Since j is less than j', points w_0^*, g_j, $g_{j'}$, w_1^* lie in clockwise order around P while w_0^*, $\mathcal{U}_j$, $\mathcal{U}_{j'}$, w_1^* lie in clockwise order around P. Thus λ_i and $\lambda_{i'}$ don't intersect.

For each point $u_i \in \mathcal{U}(c)$, let $\pi_0(u_i)$ be the endpoint on c of the path connecting u_i to c. For each point $u_i \in \mathcal{U}$ which lies in Γ, let $\pi_0(u_i)$ equal u_i. Let $\pi_1(u_i)$ be $d(\pi_0(u_i)) \cap \lambda_i$. Let w_2^* be the third vertex of the triangle containing $e^* = (w_0^*, w_1^*)$. Let $\pi_2(u_i)$ be the first intersection point of λ_i and the polygonal line from w_0^* to w_2^* to w_1^*. Place m points equally spaced on e^*. Let $\pi_3(u_i)$ be the i'th point, ordered counter-clockwise from w_0^*. Connect $\pi_0(u_i)$ to e^* with a polygonal line through $\pi_0(u_i), \pi_1(u_i), \pi_2(u_i), \pi_3(u_i)$. (See Figure 14.)

We claim that this algorithm connects $\mathcal{U}$ to e^* using $O(M)$ links where M is the number of links in some optimal solution. For each $u_i \in \mathcal{U}$, let ζ_i be the path constructed from u_i to e^* by our algorithm while η_i is the path from u_i to e^* in the optimal solution. Path ζ_i has at most three line segments in Γ. Line segment s_i is in $\sigma_{2j-1} \cup \sigma_{2j}$ for some j. If s_i contains g_j, then some point on diagonal d_i is clearly visible from g_j and d_i is a diagonal of $\Gamma_j \subseteq \Gamma$. Since d_i is the farthest diagonal visible from e^* which separates u_i from e^*, any path from u_i to e^* must have at least one line segment contained in $\Gamma_j \subseteq \Gamma$. Thus if s_i contains g_j, then we can charge the three links of ζ_i in Γ to a line segment of η_i in Γ. However, s_i may not contain g_j.

Consider the case where $s_i \in \sigma_{2j-1}$ lies between g_j and w_1^* while $s_{i'} \in \sigma_{2j}$ lies between w_0^* and g_j. Any two paths from u_i to s_i and $u_{i'}$ to $s_{i'}$ must intersect.

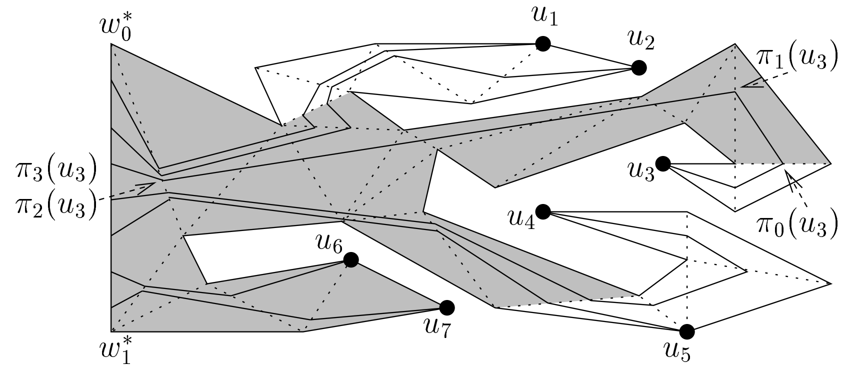


Figure 14: Pairwise disjoint paths connecting $\mathcal{U}$ to e^*.

Since paths η_i and $\eta_{i'}$ are pairwise disjoint, either the endpoint of η_i must lie between w_0^* and g_j or the endpoint of $\eta_{i'}$ must lie between g_j and w_1^*. Without loss of generality, assume that the endpoint ρ of η_i lies between w_0^* and g_j. In that case, g_j lies between ρ and s_i. Let d be the farthest diagonal of P visible from ρ and separating ρ from u_i. By the construction of d_i and s_i, diagonal d separates d_i from e^* and hence is visible to s_i. Since g_j lies between ρ and s_i, diagonal d is also visible to g_j and is contained in $\Gamma_j \subseteq \Gamma$. Thus if ρ lies between w_0^* and g_j, path η_i must have at least one line segment contained in $\Gamma_j \subseteq \Gamma$. Similarly, if the endpoint of $\eta_{i'}$ lies between g_j and w_1^*, path $\eta_{i'}$ must have at least one line segment contained in $\Gamma_j \subseteq \Gamma$. It follows that either η_i or $\eta_{i'}$ must have a line segment contained in $\Gamma_j \subseteq \Gamma$.

Let m_0, m_-, m_+ equal $f(g_j, \sigma_{2j-1})$, $f^-(g_j, \sigma_{2j-1})$ and $f^+(g_j, \sigma_{2j-1})$, respectively, while m'_0, m'_-, m'_+ equal $f(g_j, \sigma_{2j})$, $f^-(g_j, \sigma_{2j})$ and $f^+(g_j, \sigma_{2j})$, respectively. By the arguments above, the paths connecting the points in $\mathcal{U}_{2j-1} \cup \mathcal{U}_{2j}$ to e^* in the optimal solution must have at least $m_0 + m'_0 + \min(m_+, m'_-)$ line segments contained in Γ. By the choice of point g_j, $m_0 + m_- + m'_0 + m'_- \geq |\sigma_{2j-1} \cup \sigma_{2j}|/2$. Since $|\sigma_{2j-1}|$ equals $|\sigma_{2j}|$ or $|\sigma_{2j-1}| + 1$, $m_0 + m_- + m'_0 + m'_- \geq |\sigma_{2j-1}|$. On the other hand, $m_0 + m_- + m_+ = |\sigma_{2j}|$. Subtracting the second equation from the first gives $m'_0 + m'_- \geq m_+$. Thus

$$\begin{aligned} m_0 + m'_0 + \min(m_+, m'_-) &= \min(m_0 + m'_0 + m_+, m_0 + m'_0 + m'_-) \\ &\geq \min(m_0 + m'_0 + m_+, m_0 + m_+) \\ &= m_0 + m_+ \end{aligned}$$

Similarly, $m_0 + m_+ \geq m'_-$ and $m_0 + m'_0 + \min(m_+, m'_-) \geq m'_0 + m'_-$. Thus

$$m_0 + m'_0 + \min(m_+, m'_-) \geq \max(m_0 + m_+, m'_0 + m'_-) \geq \mathcal{F}(g_j, \sigma_{2j-1}, \sigma_{2j})/2.$$

The paths connecting the points in $\mathcal{U}_{2j-1} \cup \mathcal{U}_{2j}$ to e^* in the optimal solution must have at least $\mathcal{F}(g_j, \sigma_{2j-1}, \sigma_{2j})/2$ line segments in Γ. Since

$\sum_{j=1..h} \mathcal{F}(\sigma_{2j-1}, \sigma_{2j}) \geq m/40$, any pairwise disjoint paths connecting the points in $\mathcal{U}$ to e^* must have at least $m/80$ line segments contained in Γ. The construction produces at most $3m$ line segments in Γ, so the solution is at most 240 times the optimal.

Finally, we analyze the running time of our algorithm. Constructing the initial triangulation T_P takes $O(n)$ time [3, 4]. Constructing the visibility region $\hat{\mathrm{V}}\mathrm{is}(e^*)$ and the subpolygon Γ_{e^*} takes $O(n^*)$ time where n^* is the number of triangles of T_P intersected by $\hat{\mathrm{V}}\mathrm{is}(e^*)$.

The segments $s_i = \mathrm{int}(e^*) \cap \hat{\mathrm{V}}\mathrm{is}(d_i)$ for $i = 1, \ldots, m$, can also all be constructed in $O(n^*)$ time. Instead of constructing $\hat{\mathrm{V}}\mathrm{is}(d_i)$ for each d_i, construct $\hat{\mathrm{V}}\mathrm{is}(e^*)$ in Γ_{e^*} using the algorithm in [4] modified to run in $O(n^*)$ time. For each edge e visible to e^*, this algorithm explicitly constructs the points on e visible to e^* and the points on e^* visible to e. Similarly, for each diagonal d_i visible to e^*, we can construct the points on d_i visible to e^* and the points on e^* visible to d_i. This set of points on e^* is just the line segment s_i.

As discussed in the next section, partitioning $\mathcal{S}_\mathcal{U}$ into the subsequences σ_j takes $O(m \log m)$ time. Finding $g_j = g(\sigma_{2j-1} \cup \sigma_{2j})$ for $j = 1, \ldots, h$, takes $O(m)$ time. Constructing each $\hat{\mathrm{V}}\mathrm{is}(g_j)$ and Γ_j individually and merging the result to form Γ would take an unacceptable $O(n^* m)$ time. Instead we construct Γ directly. As mentioned above, the algorithm in [4] can be used to determine the points on e^* visible to each diagonal d of T_P and thus the points $g_{j_0}, g_{j+1}, \ldots, g_{j_1}$ visible to d. Let u_{i_0} be the point with lowest index i_0 in $\mathcal{U}_{2j_0-1}$. Let u_{i_1} be the point with greatest index i_1 in $\mathcal{U}_{2j_1}$. Diagonal d of T_P is an internal diagonal of Γ if and only if d separates e^* from some point in $\{u_{i_0}, u_{i_0+1}, \ldots, u_{i_1}\}$. This can be determined in constant time per diagonal so Γ can be constructed in $O(n^*)$ time.

Construction of $\mathcal{D} = \{\tilde{d}_1, \tilde{d}_2, \ldots, \tilde{d}_{\tilde{n}}\}$ and $\{\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_{\tilde{n}}\}$, the partition of $\mathcal{U}$ into sets ψ_k, the construction of the line segments λ_i and of the polygonal paths $(\pi_0(u_i), \pi_1(u_i), \pi_2(u_i), \pi_3(u_i))$ can all be done in $O(n^* + m)$ time. Thus the non-recursive steps in this algorithm take $k_1 n^* + k_2 m \log m$ time for some constants k_1, k_2.

We wish to show that the running time of the algorithm is less than $k_0(n \log m + M \log m)$ for some constant k_0. For each subpolygon P_c, $c \in \mathcal{C}$, let n_c be the size of the triangulation of P_c, let m_c be the size of $\mathcal{U}_c$ and let M_c be the size of the optimal solution connecting $\mathcal{U}_c$ to (w_0^c, w_1^c). By induction, the running time of our algorithm is less than or equal to

$$k_1 n^* + k_2 m \log m + \sum_{c \in \mathcal{C}} k_0(n_c \log m_c + M_c \log m_c).$$

We partition this expression into two parts, $k_1 n^* + \sum_{c \in \mathcal{C}} k_0(n_c \log m_c)$ and $k_2 m \log m + \sum_{c \in \mathcal{C}} k_0(M_c \log m_c)$, and bound each expression separately.

We start by bounding $k_2 m \log m + \sum_{c \in \mathcal{C}} k_0(M_c \log m_c)$. Since any optimal solution must have at least $m/80$ line segments contained in Γ,

$$m/80 + \sum_{c \in \mathcal{C}} M_c \leq M.$$

Assuming that k_0 is greater than $80k_2$,

$$k_2 m \log m + \sum_{c\in\mathcal{C}} k_0 M_c \log m_c \leq (80k_2 m/80 + \sum_{c\in\mathcal{C}} k_0 M_c)\log m \leq k_0 M \log m.$$

To bound $k_1 n^* + \sum_{c\in\mathcal{C}} k_0(n_c \log m_c)$, we divide the n^* triangles in Γ^* into those that are also in Γ and those that are outside Γ. Let n_0^* be the number of the former while n_1^* is the number of the latter. The n_0^* triangles in Γ are never encountered again in the algorithm but the the n_1^* triangles outside Γ may be processed many more times and their contribution to the running time must be carefully evaluated.

The chords $c \in \mathcal{C}$ which bound Γ can also be divided into those that also bound Γ^* and those that lie in the interior of Γ^*. Let $\mathcal{C}_0$ be the former set while $\mathcal{C}_1$ is the latter set.

We claim that m_c is at most $2/3m$ for each $c \in \mathcal{C}_1$. Let s_c be the open line segment $\mathrm{int}(e^*) \cap \hat{\mathrm{V}}\mathrm{is}(c)$ lying on e^*. Note that s_i is a subset of s_c for every $u_i \in \mathcal{U}_c$. The point g_j is a midpoint of one of the line segments in $\sigma_{2j-1} \cup \sigma_{2j}$. If $\mathcal{U}_{2j-1} \cup \mathcal{U}_{2j}$ was a subset of $\mathcal{U}_c$, then g_j would lie in s_c. In that case, c would be visible to g_j and would lie in the interior of Γ_j and Γ. Since c is on the boundary of Γ, we conclude that no set $\mathcal{U}_{2j-1} \cup \mathcal{U}_{2j}$ is a subset of $\mathcal{U}_c$,

The points in $\mathcal{U}_c$ must be split between two sets $\mathcal{U}_{2j-1} \cup \mathcal{U}_{2j}$ and $\mathcal{U}_{2j+1} \cup \mathcal{U}_{2j+2}$. Without loss of generality, assume that $\mathcal{U}_{2j-1} \cup \mathcal{U}_{2j}$ contains at least half the points in $\mathcal{U}_c$. If the size of $\mathcal{U}_c$ was more than $2m/3$, then $\mathcal{U}_{2j-1} \cup \mathcal{U}_{2j}$ would contain more than $m/3$ points of $\mathcal{U}_c$ and less than $m/3$ other points. Since g_j does not lie in s_c, the more than $m/3$ midpoints of $\sigma_{2j-1} \cup \sigma_{2j}$ which lie in s_c all lie to one side of g_j. There are less than $m/3$ other midpoints of $\sigma_{2j-1} \cup \sigma_{2j}$, so g_j would not evenly split the midpoints of $\sigma_{2j-1} \cup \sigma_{2j}$, contrary to the definition of g_j. We conclude that the size of m_c is at most $2m/3$ for every $c \in \mathcal{C}_1$.

Since a triangle that is in Γ is not in any of the polygons P_c, $c \in \mathcal{C}$,

$$n_0^* + \sum_{c\in\mathcal{C}} n_c \leq n.$$

On the other hand, since every triangle that is in Γ_e^* but not in Γ is in some polygon P_c, $c \in \mathcal{C}_1$,

$$n_1^* \leq \sum_{c\in\mathcal{C}_1} n_c.$$

Assuming k_0 is chosen to be greater than $k_1/(\log(3/2))$,

$$\begin{aligned}
k_1 n^* + \sum_{c\in\mathcal{C}}(k_0 n_c \log m_c) &\leq k_1 n_0^* + k_1 n_1^* + \sum_{c\in\mathcal{C}_0}(k_0 n_c \log m_c) \\
&\quad + \sum_{c\in\mathcal{C}_1}(k_0 n_c \log m_c) \\
&\leq k_1 n_0^* + k_1 n_1^* + \sum_{c\in\mathcal{C}_0}(k_0 n_c \log m)
\end{aligned}$$

$$
\begin{aligned}
& \quad + \sum_{c \in \mathcal{C}_1} (k_0 n_c \log(2/3m)) \\
\leq \quad & k_1 n_0^* + \sum_{c \in \mathcal{C}_0} (k_0 n_c \log m) + \sum_{c \in \mathcal{C}_1} (k_0 n_c \log m) \\
& \quad + k_1 n_1^* + \sum_{c \in \mathcal{C}_1} k_0 (n_c \log(2/3)) \\
\leq \quad & k_0 n \log m + k_1 n_1^* - \sum_{c \in \mathcal{C}_1} k_0 (n_c \log(3/2)) \\
\leq \quad & k_0 n \log m.
\end{aligned}
$$

□

We employ arguments similar to the ones in [5] to turn Lemma 1 into a theorem about connecting pairs of vertices.

Theorem 1 *Let P be a simple polygon on n vertices let $\Pi = \{(u, u')\}$ be an untangled set of m pairs of distinct vertices of P. A set of m pairwise disjoint interior paths connecting u to u' for each $(u, u') \in \Pi$ can be constructed in $O(n \log m + M \log m)$ time using $O(M)$ line segments where M is the minimum total number of line segments necessary to connect all pairs $(s, s') \in \Pi$ by pairwise disjoint paths.*

Proof: As in [5], we start by considering the problem of connecting vertices to edges in a triangulation of P. Let P be a simple polygon on n vertices with triangulation T_P and distinguished edge e^*. Let $\hat{\Pi}(u, e_u)$ be a set of pairs of vertices and edges of T_P where either $e_u = e^*$ or e_u is a diagonal of P which separates e^* from u. The vertices, but not the edges, are all distinct and are not endpoints of e^*. Let M' be the minimum total number of line segments necessary to connect the vertex-edge pairs in $\hat{\Pi}$ by pairwise disjoint paths. In $O(n + M' + m' \log m')$ time, we can construct m' pairwise disjoint interior paths ζ_u connecting u to the interior of e_u for each $(u, e_u) \in \hat{\Pi}$ where ζ_u has at most $240M' + 3m'$ line segments.

The algorithm follows the algorithm in Lemma 1. Let $\mathcal{U}'$ be the set of points for which e_u is the distinguished edge e^*. Apply the algorithm in Lemma 1 using only the points in $\mathcal{U}'$, construct the region Γ. Removing Γ subdivides P into subpolygons. Recursively solve the problem of connecting pairs of vertices and edges in each of these subpolygons. Connect each path with endpoint on the boundary of Γ to the appropriate edge in Γ. The only difference is that the connection may not be to e^* but to some edge e_u on the path to e^*.

As argued in Lemma 1, the constructed paths connecting $\mathcal{U}'$ to e^* use $3|\mathcal{U}'|$ line segments in Γ while the optimal solution requires at least $|\mathcal{U}'|/80$ line segments in Γ. Each path whose endpoint is not on e^* but on some other edge of T_P lying in Γ may also require three line segments. Since this can happen only once per path, the paths have a total of at most $240M' + 3m'$ line segments.

Now consider the original problem of connecting m pairs of vertices $(u, u') \in \Pi$. Arbitrarily choose an edge e^* of P. For each pair $(u, u') \in \Pi$, there is a

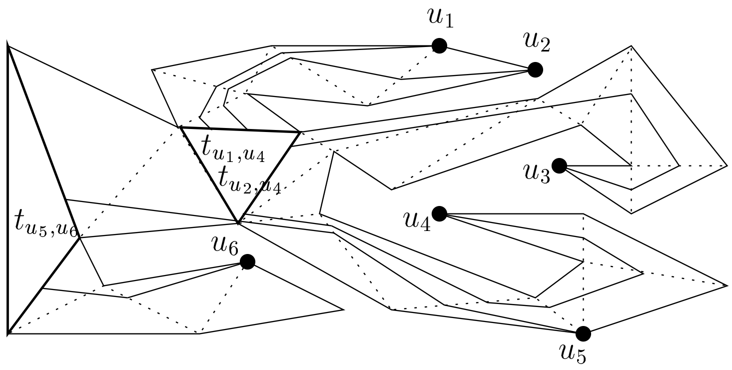


Figure 15: Triangles $t_{u,u'}$.

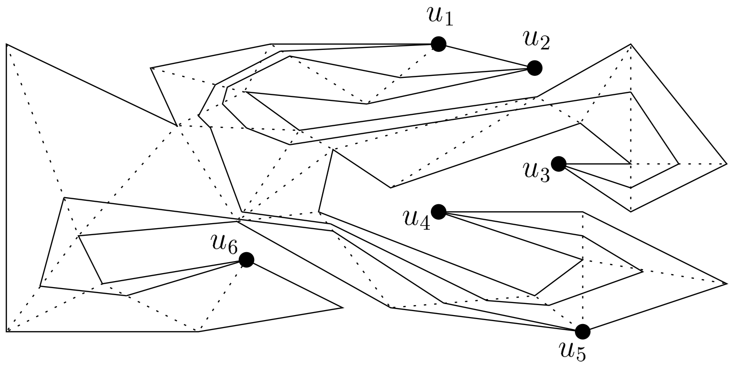


Figure 16: Paths connecting vertices.

unique triangle $t_{u,u'}$ such that any path from u to u' or any path from u to e^* or any path from u' to e^* must pass through $t_{u,u'}$. If triangle $t_{u,u'}$ does not contain u, let e_u be the edge of triangle $t_{u,u'}$ which separate $t_{u,u'}$ from u. Otherwise, let e_u be an edge of triangle $t_{u,u'}$ containing u. Similarly, let $e_{u'}$ be the edge of triangle $t_{u,u'}$ either separating $t_{u,u'}$ from u' or containing u'. (See Figure 15.) Let $\hat{\Pi}$ be the set of $m' = 2m$ pairs (u, e_u) where $(u, u') \in \Pi$ (or $(u', u) \in \Pi$.) Run the above algorithm to construct $2m$ pairwise disjoint interior paths, ζ_u, from u to e_u for each pair $(u, e_u) \in \hat{\Pi}$. Finally, for each $(u, u') \in \Pi$, connect the paths ζ_u and $\zeta_{u'}$ by a line segment from e_u to $e_{u'}$. (See Figure 16.)

Let M and M' be the minimum total number of line segments needed to connect the vertex-vertex and vertex-edge pairs in Π and $\hat{\Pi}$, respectively. The m paths connecting pairs of vertices in Π can be cut to form $m' = 2m$ pairwise disjoint paths connecting vertex-edge pairs in $\hat{\Pi}$ using at most $M + m$ line segments. Thus M' is less than or equal to $M + m$. Our algorithm uses at most

$240M' + 3m'$ line segments to connect the vertex-edge pairs in $\hat{\Pi}$ and at most $240M' + 3m' + m$ line segments to connect the vertex-vertex pairs in Π. Thus our solution uses at most $240M + 247m$ line segments. $\square$

4 Partition Algorithm

In this section, we describe and analyze the algorithm for partitioning a sequence of line segments. The functions f, f^+, f^- and $\mathcal{F}$ were defined in the previous section.

Lemma 2 *Let $\mathcal{S}$ be a sequence of line segments$(s_1, s_2, \ldots, s_m)$ on the real line $\mathbf{R}^1$. In $O(m \log m)$ time, $\mathcal{S}$ can be partitioned into contiguous subsequences $\sigma_1 = (s_1, s_2, \ldots, s_{i_1})$, $\sigma_2 = (s_{i_1+1}, s_{i_1+2}, \ldots, s_{i_2}), \ldots$, $\sigma_{2h} = (s_{i_{2h-1}+1}, s_{i_{2h-1}+2}, \ldots, s_m)$, such that:*

1. *$g(\sigma_1 \cup \sigma_2) \leq g(\sigma_3 \cup \sigma_4) \leq \cdots\cdots \leq g(\sigma_{2h-1} \cup \sigma_{2h})$;*
2. *$|\sigma_{2j-1}| = |\sigma_{2j}|\ (+1)$ for $1 \leq j \leq h$;*
3. *$\sum_{j=1..h} \mathcal{F}(\sigma_{2j-1}, \sigma_{2j}) \geq m/40$.*

Proof: Split $\mathcal{S}$ into m distinct subsequences, (s_i), consisting of one element each. Store the m sets in a linked list $\mathcal{A}$ in the order they appear in $\mathcal{S}$. Each set is stored in a separate node a in the linked list where $a.seq$ contains the subsequence stored at a, $a.size = |a|$ is the number of elements in the $a.seq$, $a.next$ is the next node in the linked list, and $a.prev$ is the previous node in the linked list. Let $\mathcal{A}.first$ and $\mathcal{A}.last$ be the first and last nodes in $\mathcal{A}$. We call $\mathcal{A}$ *balanced* if $3|a| \geq |a.next|$ and $3|a| \geq |a.prev|$ for all nodes $a \in \mathcal{A} - \{\mathcal{A}.first, \mathcal{A}.last\}$. Clearly, $\mathcal{A}$ is initially balanced.

While $\mathcal{A}$ contains some node a such that $g(a.seq) > g(a.next.seq)$, repeat the following. First, merge node a with $a.next$, storing the result in a. Next, call procedures BALANCE-NEXT and BALANCE-PREV to rebalance the linked list by merging and splitting nodes near a. Repeat this while loop until $g(a.seq) \leq g(a.next.seq)$ for each a in $\mathcal{A}$. (See Figure 17.)

Procedure BALANCE-NEXT rebalances nodes after a as follows. Let a' equal a. Note that a' was formed by merging nodes from a balanced list, so $3|a'|$ is automatically greater than or equal to $|a'.next|$. We need to ensure that $3|a'.next| \geq |a'|$. If $|a'|$ is greater than $3|a'.next|$, then merge nodes $a'.next$ and $a'.next.next$ storing the result in $a'.next$. Continue merging $a'.next$ and $a'.next.next$ until $|a'|$ is less than or equal to $3|a'.next|$. If $|a'.next|$ is less than or equal to $3|a'.next.next|$, then stop. If $|a'.next|$ is greater than $\max(3|a'.next.next|, (2/3)|a'|)$, then split $a'.next$ into two nodes of approximately equal size and stop. Otherwise, let a' equal $a'.next$ and start again. In both cases, the condition $3|a'| \geq |a'.next|$ is satisfied. In a similar manner, BALANCE-PREV rebalances nodes before a. (See Figure 18.)

Steps 10 and 11 in BALANCE-NEXT and BALANCE-PREV require some explanation. Let b_1, b_2, b_3 be the three nodes after a' at the completion of

PARTITION($\mathcal{S}$)
/* $\mathcal{S}$ = a sequence of line segments $(s_1, s_2, \ldots, s_m)$ */
/* Returns a linked list of contiguous subsequences of $\mathcal{S}$ */
1. Initialize linked list $\mathcal{A}$ to $\emptyset$;
2. FOR $i = 1$ TO m DO
3. Create new node a where $a.seq = (s_i)$ and $a.size = 1$;
4. Add a to the end of linked list $\mathcal{A}$;
5. WHILE $\exists a \in \mathcal{A}$ such that $g(a.seq) > g(a.next.seq)$ DO
6. Merge a and $a.next$ to form a new node a' in $\mathcal{A}$;
7. BALANCE-NEXT(a');
8. BALANCE-PREV(a');
9. Return($\mathcal{A}$).

Figure 17: Algorithm Partition-Segments.

BALANCE-NEXT, where $b_1 = a'.next$, $b_2 = b_1.next$ and $b_3 = b_2.next$. Nodes b_1 and b_2 were created by splitting a node whose size is greater than $(2/3)|a'|$ and $3|b_3|$, so $3|b_1| \geq |a'|$ and $3|b_2| \geq |b_3|$. Nodes b_1 and b_2 have approximately the same size, so $3|b_2| \geq |b_1|$ and $3|b_1| \geq |b_2|$. The remaining condition we need to check is $3|b_3| \geq |b_2|$.

Let b' be the node split to create b_1 and b_2. Node b' was created by merging a sequence of nodes $b'_1, b'_2, \ldots, b'_k$, where $a'.next = b'_1$ and $b'_i.next = b'_{i+1}$. The total number of elements in $b'_1, b'_2, \ldots, b'_{k-1}$ is less than $|a|/3$ or b'_k would not have been added to b'. The value of $|b'_k|$ must be at least $|a'|/3$ or else $|b'|$ would not be greater than $(2/3)|a'|$. Thus

$$|b'_k| > |a'|/3 > \sum_{i=1..k-1} |b'_i|.$$

Since the list was initially balanced,

$$|b_3| \geq |b'_k|/3 \geq \left(\sum_{i=1..k-1} |b'_i| \right) /3.$$

Thus,

$$|b_3| \geq (|b'_k| + \sum_{i=1..k-1} |b'_i|)/6 \geq |b_2|/3,$$

and the new list is balanced.

Let a_j be the j'th node in $\mathcal{A}$ when the algorithm is completed. Partition $a_j.seq = (s_i, \ldots, s_{i'})$ into two approximately equal sized sequences $\sigma_{2j-1} = (s_i, \ldots, s_{\lceil (i+i')/2 \rceil})$ and $\sigma_{2j} = (s_{\lceil (i+i')/2 \rceil + 1}, \ldots, s_{i'})$. We claim that this is a partitioning of $\mathcal{S}$ with the desired properties. Initially, the s_j are stored in $\mathcal{A}$ in sorted order. The merging and splitting steps in the main algorithm and in the subroutines BALANCE-NEXT and BALANCE-PREV preserve the order of the s_i, so the σ_j properly partition $\mathcal{S}$ into contiguous subsequences.

```
BALANCE-NEXT(a)
/* a = node in linked list */
/* Ensure 3|a'| ≥ |a'.prev| for all a' ∈ A − {A.first, A.last} */
If a = A.last THEN RETURN;
a' ← a;
WHILE a'.next ≠ A.first AND |a'| > 3|a'.next| DO
    REPEAT
        Merge a'.next and a'.next.next;
    UNTIL |a'| ≤ 3|a'.next| OR a'.next = A.last;
    IF a'.next = A.last THEN RETURN;
    IF |a'.next| ≤ 3|a'.next.next| THEN RETURN;
    IF |a'.next| > (2/3)|a'| THEN
        Split a'.next into two equal sized sets;
        RETURN;
    a' ← a'.next;
```

```
BALANCE-PREV(a)
/* a = node in linked list */
/* Ensure 3|a'| ≥ |a'.next| for all a' ∈ A − {A.first, A.last} */
If a = A.first THEN RETURN;
a' ← a;
WHILE a'.prev ≠ A.first AND |a'| > 3|a'.prev| DO
    REPEAT
        Merge a'.prev and a'.prev.prev;
    UNTIL |a'| ≤ 3|a'.prev| OR a'.prev = A.first;
    IF a'.prev = A.first THEN RETURN;
    IF |a'.prev| ≤ 3|a'.prev.prev| THEN RETURN;
    IF |a'.prev| > (2/3)|a'| THEN
        Split a'.prev into two equal sized sets;
        RETURN;
    a' ← a'.prev;
```

Figure 18: Algorithms BALANCE-NEXT and BALANCE-PREV.

Let g_j be $g(a_j.seq) = g(\sigma_{2j-1} \cup \sigma_{2j})$. The while loop only terminates when $g_1 \leq g_2 \leq \cdots \leq g_h$, so property 1 is clearly satisfied. Sets σ_{2j-1} and σ_{2j} are created by partitioning $a_j.seq$ into two equal sized sequences, so property 2 is satisfied.

To show property 3 holds, note that a_j could be an initial node or it could be created when $g(a.seq) > g(a.next.seq)$ or it could be created in the rebalancing procedure. We first show that at most $(4/5)m$ line segments lie in nodes a_j created in the balance step.

Let $b_1, b_2, \ldots, b_k$ be nodes created in a call BALANCE-NEXT(a) where $a.next = b_1$ and $b_j.next = b_{j+1}$ for $j \leq k$. Since BALANCE-NEXT stops whenever $|a'.next| > (2/3)|a'|$, $|b_1| \leq (2/3)|a|$ and $|b_j| \leq (2/3)|b_{j-1}|$ for all $2 \leq j \leq k-1$. We claim that $|b_k| \leq (4/3)|b_{k-1}|$. Let $b'_1, b'_2, \ldots, b'_h$ be the nodes merged to form b_k, where $b'_j.next = b'_{j+1}$, $j \leq h$. The total number of elements in $b'_1, b'_2, \ldots, b'_{h-1}$ is less than $|b_{k-1}|/3$ or b'_h would not have been added to b_k. In particular, $|b'_{h-1}|$ must be less than $|b_{k-1}|/3$. Since the list was initially balanced,

$$|b'_h| \leq 3|b'_{h-1}| \leq 3(|b_{k-1}|/3) = |b_{k-1}|.$$

Thus $\sum_{j=1..h} |b'_j| \leq (4/3)|b_{k-1}|$. Summing the values of $|b_j|$, we get:

$$\sum_{i=1..k} |b_j| \leq \left(\sum_{i=1..k-1} (2/3)^i |a| \right) + (4/3)(2/3)^{k-1}|a| \leq 2|a|.$$

Similarly, the total size of the nodes created in a call BALANCE-PREV(a) is at most $2|a|$. Since each line segment in a corresponds to at most 4 line segments in nodes created in BALANCE-NEXT(a) and BALANCE-PREV(a), there are at most $(4/5)m$ line segments in nodes created in the balance step.

If a_j is an initial node, then $a_j.seq = \{s\}$ for some $s \in \mathcal{S}$ and $\sigma_{2j-1} = \{s\}$ and $\sigma_{2j} = \emptyset$. Point g_j is the midpoint of s and $\mathcal{F}(\sigma_{2j-1}, \sigma_{2j}) \geq 1 \geq (1/8)|a_j|$.

Assume a_j is created when $g(a.seq)$ is greater than $g(a.next.seq)$ and that $|a.next| \geq |a|$. The sequence $a.seq$ is a subsequence of σ_{2j-1}, so

$$\begin{aligned} f(g_j, \sigma_{2j-1}) &\geq f(g_j, a.seq) \text{ and} \\ f^+(g_j, \sigma_{2j-1}) &\geq f^+(g_j, a.seq). \end{aligned}$$

The point $g_j = g(a_j.seq)$ must lie between $a.g$ and $a.next.g$, so

$$\begin{aligned} f(g_j, a.seq) + f^+(g_j, a.seq) &\geq f(g(a.seq), a.seq) + f^+(g(a.seq), a.seq) \\ &\geq |a|/2. \end{aligned}$$

Since $|a.next| \leq 3|a|$, we have $|a_j| \leq 4|a|$. Thus,

$$\mathcal{F}(\sigma_{2j-1}, \sigma_{2j}) \geq f(g_j, \sigma_{2j-1}) + f^+(g_j, \sigma_{2j-1}) \geq (1/8)|a_j|.$$

In the case that $|a.next| < |a|$, similar reasoning gives

$$\mathcal{F}(\sigma_{2j-1}, \sigma_{2j}) \geq f(g_j, \sigma_{2j}) + f^-(g_j, \sigma_{2j}) \geq (1/8)|a_j|.$$

Since at least $m/5$ line segments are in nodes a_j with the property that $\mathcal{F}(\sigma_{2j-1}, \sigma_{2j}) \geq (1/4)|a_j|$, $\sum_{i=1..k} \mathcal{F}(\sigma_{2j-1}, \sigma_{2j}) \geq m/40$.

Finally, we show the algorithm PARTITION runs in $O(m \log m)$ time. Steps 1-4 take $O(m)$ time. If the sequences $a.seq$ are stored as linked lists with pointers to the beginnings and ends of the lists, merging two nodes in step 6 takes constant time. Merging two nodes decreases the number of nodes in $\mathcal{A}$ by one. Procedures BALANCE-NEXT and BALANCE-PREV never increase the number of such nodes. (Node $a'.next$ is only split in step 10 if it is the product of merges which increased its size too drastically.) Since list $\mathcal{A}$ starts with m nodes, step 6 is executed at most $m-1$ times. Similarly, procedures BALANCE-NEXT and BALANCE-PREV are called at most $m-1$ times.

The running time of BALANCE-NEXT is dominated by the inner loop in step 5 and the time to execute step 10. Each execution of step 5 reduces the number of nodes in $\mathcal{A}$ by one. Step 10 increases the number of nodes in $\mathcal{A}$ by one, but it is only executed once per call to BALANCE-NEXT. Since BALANCE-NEXT is only called $m-1$ times and there are at most m nodes in $\mathcal{A}$, step 5 is executed at most $2m-2$ times.

The time to split node $a'.next$ in step 10 of BALANCE-NEXT is proportional to $|a'.next|$, the size of $a'.next.seq$. As argued above, the total size of the nodes created in the call BALANCE-NEXT(a') is at most $2|a'|$. Thus the total running time of step 10 is proportional to the sum of the sizes of nodes created in step 6 of PARTITION. Step 6 merges nodes a and $a.next$ to create a new node a'. The size of a' is at least $4/3$ times the size of a and $4/3$ times the size of $a.next$. Since each time a node is merged in step 10, its size increases by $4/3$, any given line segment s is in $O(\log m)$ merge steps. Thus the sum of the sizes of nodes created in step 6 of PARTITION and the total running time of step 10 of BALANCE-NEXT is $O(m \log m)$. A similar analysis applies to BALANCE-PREV.

We are left with analyzing the running time of step 5 of PARTITION. Step 5 checks if $a.next.g < a.g$ for some $a \in \mathcal{A}$. We can accomplish this by keeping a set $\mathcal{A}^*$ of nodes in $\mathcal{A}$ to be checked. For each node $a \in \mathcal{A}^*$, we check that $g(a.seq) \leq g(a.next.seq)$ and that $g(a.seq) \geq g(a.prev.seq)$. The points $g(a.seq)$, $g(a.next.seq)$ and $g(a.prev.seq)$ can be found in time proportional to $|a|$, $|a.next|$ and $|a.prev|$, respectively, using a linear median find algorithm. Since $|a.next| \leq 3|a|$ and $|a.prev| \leq 3|a|$, comparing $g(a.seq)$, $g(a.next.seq)$ and $g(a.prev.seq)$ takes $O(|a|)$ time. Node a can be created as a merge of two nodes in step 5 of PARTITION or in BALANCE-NEXT or BALANCE-PREV. As previously argued, the sum of the sizes of the nodes created in step 5 of PARTITION is $O(m \log m)$. Each node a' created in step 5 can cause the creation of nodes in BALANCE-NEXT(a') and BALANCE-PREV(a') with a total of $4|a|$ elements. Thus the sum of the sizes of created nodes is $O(m \log m)$ and the running time of step 5 is $O(m \log m)$.

$\square$

5 Constructing Isomorphic Triangulations

A triangulation T_P of P (possibly with interior vertices) is *isomorphic* to a triangulation T_Q of Q if there is a one-to-one, onto mapping f between the vertices of T_P and the vertices of T_Q such that p, p', p'' are vertices of a triangle in T_P if and only if $f(p), f(p'), f(p'')$ are vertices of a triangle in T_Q. An isomorphic triangulation of P and Q also defines a piecewise linear homeomorphism between P and Q. The size of a triangulation is the total number of vertices, edges and triangles in the triangulation.

The result in Theorem 1 leads directly to an improvement in the output size of the algorithm for constructing isomorphic triangulations in [5]. We outline the improved algorithm here. Readers are referred to [5] for more complete details and figures of the original algorithm.

Theorem 2 *Let P be a simple polygon with vertices $\{p_1, p_2, \ldots, p_n\}$ and Q be a simple polygon with vertices $\{q_1, q_2, \ldots, q_n\}$. Let M_1 be the minimum number of triangles in any isomorphic triangulation of P and Q mapping p_i to q_i. Isomorphic triangulations mapping p_i to q_i using $O(M_1 \log n)$ edges can be constructed in $O(M_1 \log n)$ time. Moreover, no new vertices are added to the boundary of P and Q in these triangulations.*

Proof: As in [5], construct a triangulation T_P of P using no vertices other than those of P and choose a set of edges $\mathcal{C}$ from T_P as follows. Arbitrarily, pick an edge e of P. Let Ψ be the union of the set of triangles of T_P which intersect $\hat{\mathrm{V}}\mathrm{is}(e)$. The boundary of Ψ is composed of edges and chords of the original polygon P. Add all the chords bounding Ψ to $\mathcal{C}$. Each such chord c divides P into two polygons. Let $P(c)$ be one not containing Ψ. Recursively apply this procedure to each such polygon $P(c)$ starting at its edge c, to choose a set of edges from $P(c)$ and add them to $\mathcal{C}$.

For each diagonal (p_i, p_j) in P there is a corresponding pair of vertices (q_i, q_j) in Q. Let k be the size of $\mathcal{C}$ and Π be the set of k pairs of distinct vertices of Q corresponding to the diagonals in $\mathcal{C}$. Note that the pairs of vertices in Π are untangled. By Theorem 1, a set of k non-intersecting interior paths connecting each pair $(q_i, q_j) \in \Pi$ can be constructed in $O(n + M_2 + k \log k)$ time using a total of at most $240M_2 + 247k$ links, where M_2 is the minimum total number of line segments necessary to connect all pairs $(q_i, q_j) \in \Pi$ by pairwise disjoint paths. For each new vertex q' on the path from q_i to q_j create a corresponding new vertex p' on the diagonal (p_i, p_j).

The k diagonals in $\mathcal{C}$ split polygon P into $k+1$ subpolygons $P_1, P_2, \ldots, P_{k+1}$. The interior paths split polygon Q into $k+1$ corresponding subpolygons Q_1, $Q_2, \ldots, Q_{k+1}$. Each subpolygon P_i has an edge e which lies within link distance two of every point in P_i. Thus each P_i has link diameter at most five. By Lemma 3.4 in [5], isomorphic triangulations of P_i and Q_i can be constructed in $O(n_i \log n_i)$ time using $40 n_i \lfloor \log_2 n_i \rfloor + O(n_i)$ edges, where n_i is the number of vertices of P_i and Q_i. Since these isomorphic triangulations do not contain any new boundary vertices, their union is an isomorphic triangulation of P and Q.

The total number of edges in the isomorphic triangulations of P and Q is at most $\sum_{i=1}^{k+1}(40n_i\lfloor\log_2 n_i\rfloor + O(n_i))$. Since each edge of Q_i is either an edge of Q or an edge on one of the k non-intersecting interior paths, $\sum_{i=1}^{k+1} n_i < n + 2(240M_2 + 247k)$. Bounding $\log_2 n_i$ by $\log_2 n$ and k by n gives a bound of

$$40(480M_2 + 495n)\lfloor\log_2 n\rfloor + O(M_2 + n) = O(M_2 \log n + n \log n).$$

Let T_P^* and T_Q^* be an optimal pair of isomorphic triangulations of P and Q mapping p_i to q_i and let M_1 be the number of triangles in T_P^* (which equals the number of triangles in T_Q^*.) We claim that $M_2 \leq 3M_1$. As isomorphic triangulations define a piecewise linear mapping between P and Q [5], each diagonal $(p_i, p_j) \in \mathcal{C}$ has a mapping, induced by the isomorphic triangulations T_P^* and T_Q^*, in Q. The set of diagonals $(p_i, p_j) \in \mathcal{C}$ get mapped to a set of disjoint interior link paths, $\{\zeta_{i,j}\}$, between corresponding pairs of vertices $(q_i, q_j) \in \Pi$ of Q. By the choice of $\mathcal{C}$, no triangle of T_P^* intersects more than three diagonals in $\mathcal{C}$. Hence, no triangle of T_Q^* intersects more than three link paths in $\{\zeta_{i,j}\}$ and contains more than three line segments of $\{\zeta_{i,j}\}$. Therefore, the paths $\{\zeta_{i,j}\}$ use at most $3M_1$ line segments. As M_2 is the number of line segments in an optimal construction of link paths, $M_2 \leq 3M_1$. Since $M_1 \geq n - 2$, we conclude that $M_2 \log n + n \log n = O(M_1 \log n)$.

As in [5], it can be shown that the running time of the above algorithm is $O(M_1 \log n)$. □

6 Conclusions

Let P be a simple polygon on n vertices, let Π be an untangled set of m pairs of distinct vertices of P, and let L be the sum of the interior link distances between pairs in Π. In [5] we proved that pairwise disjoint interior paths connecting the pairs can be constructed using $L + O(m \log m)$ line segments. In this paper we showed that $L + \Omega(m \log m)$ line segments are sometimes required. We gave an approximation algorithm for constructing pairwise disjoint paths in $O(n + M + m \log m)$ time using $O(M)$ line segments where M is the minimum number required. We applied this approximation algorithm to construct isomorphic triangulations of two polygons in $O(M_1 \log n)$ time using $O(M_1 \log n)$ edges where M_1 is the size of the optimal solution. This improved our result in [5].

We still do not know if constructing pairwise disjoint paths using the fewest line segments is an NP-complete problem or whether it can be solved in polynomial time. We also do not know of an ϵ-approximation scheme for constructing such paths. Possibly the techniques of Arora in [2] for the Euclidean Travelling Salesman Problem could apply.

In [6], Kahan and Snoeyink show that a polygon on n vertices with bit complexity $n \log n$ may have a minimal link path with $\Theta(n^2 \log n)$ bit complexity. Similar problems plague pairwise disjoint paths with the minimum number of line segments. Because the algorithms in this paper are approximation algorithms, the bit complexity of coordinates in the output is a constant factor times the bit complexity of input coordinates.

References

[1] Aronov, B., Seidel, R., and Souvaine, D. On compatible triangulations of simple polygons. *Comput. Geom. Theory Appl. 3*, 1 (1993), 27–35.

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

[3] Chazelle, B. Triangulating a simple polygon in linear time. *Discrete Comput. Geom. 6* (1991), 485–524.

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

[5] Gupta, H., and Wenger, R. Constructing piecewise linear homeomorphisms of simple polygons. *J. Algorithms 22* (1997), 142–157.

[6] Kahan, S., and Snoeyink, J. On the bit complexity of minimum link paths: Superquadratic algorithms for problems solvable in linear time. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.* (1996), pp. 151–158.

[7] Kranakis, E., and Urrutia, J. Isomorphic triangulations with small number of Steiner points. In *Proc. 7th Canad. Conf. Comput. Geom.* (1995), pp. 291–296.

[8] Saalfeld, A. Joint triangulations and triangulation maps. In *Proc. 3rd Annu. ACM Sympos. Comput. Geom.* (1987), pp. 195–204.

[9] Souvaine, D., and Wenger, R. Constructing piecewise linear homeomorphisms. Technical Report 94–52, DIMACS, New Brunswick, New Jersey, 1994.

[10] Suri, S. A linear time algorithm for minimum link paths inside a simple polygon. *Comput. Vision Graph. Image Process. 35* (1986), 99–110.