

Minimum weight pseudo-triangulations

Joachim Gudmundsson¹ and Christos Levcopoulos²

¹ National ICT Australia Ltd^{***}, Australia. joachim.gudmundsson@nicta.com.au

² Department of Computer Science, Lund University, Box 118, 221 00 Lund, Sweden.
christos@cs.lth.se

Abstract. We consider the problem of computing a minimum weight pseudo-triangulation of a set \mathcal{S} of n points in the plane. We first present an $\mathcal{O}(n \log n)$ -time algorithm that produces a pseudo-triangulation of weight $\mathcal{O}(\log n \cdot wt(\mathcal{M}(\mathcal{S})))$ which is shown to be asymptotically worst-case optimal, i.e., there exists a point set \mathcal{S} for which every pseudo-triangulation has weight $\Omega(\log n \cdot wt(\mathcal{M}(\mathcal{S})))$, where $wt(\mathcal{M}(\mathcal{S}))$ is the weight of a minimum spanning tree of \mathcal{S} . We also present a constant factor approximation algorithm running in cubic time. In the process we give an algorithm that produces a minimum weight pseudo-triangulation of a simple polygon.

1 Introduction

Pseudo-triangulations are planar partitions that recently received considerable attention [1–3] mainly due to their applications in visibility [22–24], ray-shooting [7, 11], kinetic collision detection [4, 16, 17], rigidity [27], and guarding [26].

A pseudo-triangle is a planar polygon that has exactly three convex vertices, called corners. A pseudo-triangulation of a set \mathcal{S} of n points in the plane is a partition of the convex hull of \mathcal{S} into pseudo-triangles whose vertex set is exactly \mathcal{S} .

A related problem is the problem of triangulating a point set. Minimizing the total length has been one of the main optimality criteria for triangulations and other kinds of partition. Indeed the minimum weight triangulation (MWT), i.e., minimizing the sum of the edge lengths, has frequently been referred to as the “optimal triangulation”. This triangulation has some good properties [8] and is e.g. useful in numerical approximation of bivariate data [28]. The complexity of computing a minimum weight triangulation is one of the most longstanding open problems in computational geometry and it is included in Garey and Johnson’s [9] list of problems from 1979 that neither are known to be NP-complete, nor known to be solvable in polynomial time. As a result approximation algorithms for the MWT-problem have been considered. The best known approximation is a constant factor approximation algorithm by Levcopoulos and Krzrnaric [19].

In this paper we consider the problem of computing a pseudo-triangulation of minimum weight (MWPT) which was posed as an open problem by Rote et al. in [25]. An interesting observation that makes the pseudo-triangulation very favorable compared to a standard triangulation is the fact that there exist point sets where any triangulation, and also any convex partition (without Steiner points), has weight $\Omega(n \cdot wt(\mathcal{M}(\mathcal{S})))$, while there always exists a pseudo-triangulation of weight $\mathcal{O}(\log n \cdot wt(\mathcal{M}(\mathcal{S})))$, where $wt(\mathcal{M}(\mathcal{S}))$ is the weight of a minimum spanning tree of the point set. We also present an approximation algorithm that produces a pseudo-triangulation whose weight is within a factor 12 times the weight of the MWPT. In comparison, the best constant approximation factor for the MWT-problem which is proved to be achievable by a polynomial-time algorithm [19] is so much larger that it has not been explicitly calculated.

Recently, there has been considerable research in the problem of computing a *pointed* pseudo-triangulation [27], i.e., a pseudo-triangulation with a minimum number of pseudo-triangles. The pseudo-triangulation produced in this paper will in general not be pointed, however, one of the algorithms (Theorem 3) can easily be extended to also hold for pointed pseudo-triangulations.

^{***} Australian Government’s Backing Australia’s Ability initiative, in part through the Australian Research Council

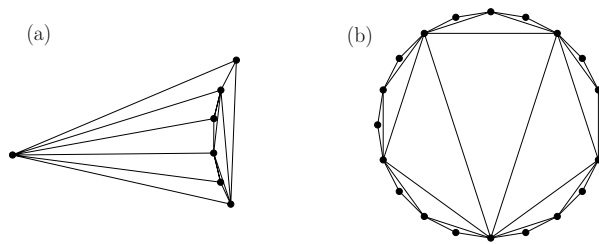


Fig. 1. (a) An example where any triangulation will have weight $\Omega(n \cdot wt(\mathcal{M}(\mathcal{S})))$. (b) An example where any pseudo-triangulation will have weight $\Omega(\log n \cdot wt(\mathcal{M}(\mathcal{S})))$.

This paper is organized as follows. In the next section we compare the worst-case weight of a triangulation with the worst-case weight of a pseudo-triangulation. We give an algorithm that produces a pseudo-triangulation that asymptotically meets this bound running in time $\mathcal{O}(n \log n)$. Even though this weight bound is asymptotically worst-case optimal it can be far from the optimal solution for many point sets. In sections 3 and 4 we focus on finding a constant factor approximation algorithm for the MWPT-problem. As a subroutine we use an algorithm that we believe is of independent interest since it computes an optimal solution of a simple polygon in cubic time.

An edge/segment with endpoints in two points u and v of \mathcal{S} will be denoted by (u, v) and its length $|uv|$ is equal to the Euclidean distance between u and v . Given a graph \mathcal{T} on \mathcal{S} we denote by $wt(\mathcal{T})$ the sum of all the edge lengths of \mathcal{T} . The minimum spanning tree of \mathcal{S} and the convex hull of \mathcal{S} , denoted $\mathcal{M}(\mathcal{S})$ and $\mathcal{CH}(\mathcal{S})$ respectively, will be used frequently throughout the paper. Both structures can be computed in $\mathcal{O}(n \log n)$ time, see [5].

2 A fast approximation algorithm

As mentioned in the introduction there exists a point set \mathcal{S} for which any triangulation will have weight $\Omega(n \cdot wt(\mathcal{M}(\mathcal{S})))$, an example is given in Fig. 1a. A natural question is whether there exist similar worst-case bounds for pseudo-triangulations. In this section we show that one can always construct a pseudo-triangulation of weight $\mathcal{O}(\log n \cdot wt(\mathcal{M}(\mathcal{S})))$, and this is asymptotically tight, i.e., there exists a point set \mathcal{S} for which every pseudo-triangulation has weight $\Omega(\log n \cdot wt(\mathcal{M}(\mathcal{S})))$. We start with the lower bound.

Observation 1 *There exists a point set \mathcal{S} in the plane such that any pseudo-triangulation has weight $\Omega(\log n \cdot wt(\mathcal{M}(\mathcal{S})))$.*

Proof. Let \mathcal{S} be a set of n equally spaced points on a circle with diameter 1, as illustrated in Fig. 1b. Any pseudo-triangulation of \mathcal{S} is also a triangulation since all points are on the convex hull. A triangulation can be seen as a number of rounds where one cut off a number of ears in each round. A minimum weight triangulation will use $\Omega(\log n)$ rounds [20]. Each round will use segments of total length at least equal to the radius of \mathcal{S} . The observation follows since the weight of a minimum spanning tree of \mathcal{S} is at most π , and the weight of the triangulation is $\Omega(\log n)$. For detailed arguments concerning the length and structure of the minimum weight triangulation of regular polygons, and pieces of regular polygons, we refer to Theorem 8 in [20]. \square

Next we present an algorithm that produces a pseudo-triangulation whose weight asymptotically meets the lower bound, that is:

Theorem 1. *Given a set \mathcal{S} of n points in the plane one can produce a pseudo-triangulation of \mathcal{S} of weight $\mathcal{O}(\log n \cdot wt(\mathcal{M}(\mathcal{S})))$ in $\mathcal{O}(n \log n)$ time.*

The algorithm has three main steps: first a partition of the convex hull of \mathcal{S} into simple polygons, secondly a partition of each polygon into *restricted weak visibility* polygons (to be defined) and finally, every restricted weak visibility polygon is pseudo-triangulated.

The first step of the algorithm is trivial. As input we are given a set \mathcal{S} of n points in the plane, and as output we will produce a set of simple polygons. Construct the convex hull and the minimum spanning tree of \mathcal{S} . This is done in $\mathcal{O}(n \log n)$ time and it partitions $\mathcal{CH}(\mathcal{S})$ into simple (maybe degenerate) polygons.

Next, the polygons are partitioned into restricted weak visibility polygons. We believe that this step might be useful as a general tool thus we include a detailed description of this step in Section 2.1. Finally, each restricted weak visibility polygon is pseudo-triangulated. This step is described in Section 2.2.

2.1 Partition a simple polygon into simpler pieces

We start with some basic definitions. Two points p and q within a polygon P are said to *see* each other if there exists a straight-line segment within P with endpoints at p and q . A polygon P is said to be a *visibility polygon* with respect to a vertex q of P if every point within P can be seen from q . A polygon P is said to be a *weak visibility polygon* with respect to an edge (p, q) of P if every point within P can see at least one point on (p, q) . Note that this definition requires us to define if the segment is open or closed, since the resulting polygons will differ. The edge $e = (p, q)$ is called the (*open*) *visibility edge* of P .

In this section the aim is to partition a simple polygon into **rwv**-polygons. A *restricted weak visibility (rwv) polygon* of a polygon P with respect to an open segment e of P is a weak visibility polygon of P with respect to the open segment e such that every vertex of $\text{rwv}(P, e)$ also is a vertex of P , see Fig. 2b.

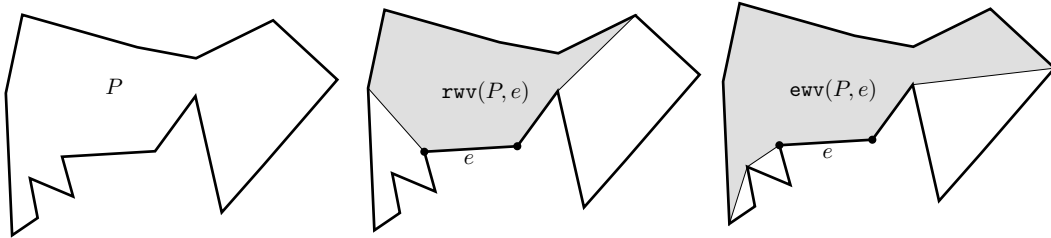


Fig. 2. (a) The input polygon P . (b) The restricted weak visibility polygon of P with respect to the open segment e . (c) The extended weak visibility polygon of P with respect to the open segment e .

The following observation shows one of the main properties of an **rwv**-polygon of an open segment. Note that an **rwv**-polygon is a weak visibility polygon whose visibility edge (p, q) has two interior convex vertices

Observation 2 *Let P be an **rwv**-polygon with respect to an open segment (p, q) of P , the geodesic shortest path between any pair of points u and v in P is a concave chain.*

Proof. The observation follows since there exists a path containing three edges within P from u to v , via the “visibility” edge (p, q) of P . This path may self-intersect but in that case the path can be shortened to two edges. \square

Step 1: Partition into **ewv-polygons** The partition of P will be done in two steps. In the first step P is partitioned into extended weak visibility polygons. An *extended weak visibility (ewv) polygon* of a polygon P with respect to an open segment (p, q) contains all edges of P that can be partially seen from (p, q) in order, inter-connected with the shortest paths within P , see Fig. 2c. Below we give a brief description on how an **ewv**-polygon can be computed.

Let P be a simple polygon with n vertices $p = v_1, \dots, v_n = q$ in clockwise order. Consider a weak visibility polygon Q of the open segment (p, q) of P and let $e = (q_1, q_2)$ be an edge of Q that does not coincide with an edge of P , as illustrated in Fig. 3.

Assume that q_1 is a vertex of P and that q_2 lies on an edge (v_i, v_{i+1}) of P . Rotate P such that e is vertical and q_2 is above q_1 . Consequently v_i lies to the left of e . Let $P'(e)$ be the polygon bounded by (q_1, q_2) , (q_2, v_i) and $\delta(v_i, q_1)$, where $\delta(v_i, q_1)$ is the shortest path within P between v_i and q_1 . Note that $\delta(v_i, q_1)$ is either a straight-line segment between v_i and q_1 or a concave chain, as shown in Fig. 3.

The ewv-polygon Q' of P with respect to the open segment (p, q) can now be computed from P and Q as follows. Let e_1, \dots, e_m be the set of edges of Q that do not coincide with any edge of P . The extended weak visibility polygon with respect to (p, q) is the union of Q and the polygons $P'(e_1), \dots, P'(e_m)$.

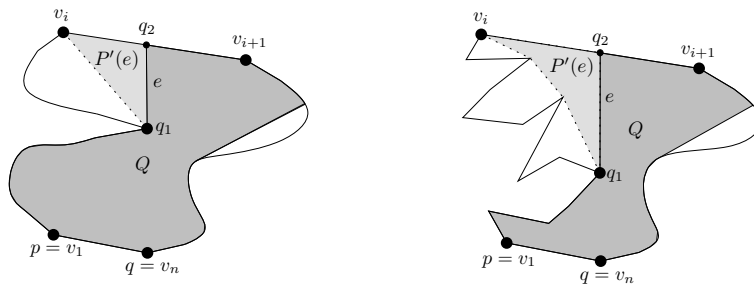


Fig. 3. The two cases that may occur when building an ewv-polygon.

The ewv-polygon Q of P partitions P into a set of simple polygons, denoted Q, P_1, \dots, P_k . Recursively continue the partition by computing the extended weak visibility polygon of P_i , $1 \leq i \leq k$, with respect to the open segment e_i where e_i is an edge of Q that does not coincide with an edge of P . This continues recursively until P has been entirely partitioned into extended weak visibility polygons.

Lemma 1. *A simple polygon P can be partitioned into ewv-polygons in $\mathcal{O}(n \log n)$ time such that the weight of the added segments is bounded by $wt(P)$.*

Proof. Computing the weak visibility polygon of a simple polygon can be done in linear time [6, 13]. Given the weak visibility polygon the ewv-polygon can be computed in $\mathcal{O}(m \log n)$ time in total where m is bounded by the number of edges in the weak visibility polygon. This follows from the result by Hershberger [14] who showed that shortest path queries within a simple polygon can be answered in $\mathcal{O}(\log n + k)$ time using $\mathcal{O}(n)$ preprocessing, where k is the complexity of the shortest path.

The time bound stated in the lemma is obtained by noting that the number of edges considered in each step together with the total complexity of all the extended weak visibility polygons is bounded by $\mathcal{O}(n)$.

Finally we prove the length bound. Consider an arbitrary visibility edge e of an ewv-polygon Q of P , and let $p(e)$ be the part of the perimeter of Q with an orthogonal projection onto e . Note that $p(e)$ is also the perimeter of P , obviously $p(e)$ can only belong to one ewv-polygon and thus the total weight of the added segments is bounded by $wt(P)$. \square

Step 2: Partition an ewv-polygon into rwv-polygons Let P be an extended weak visibility polygon with n vertices and let $e = (p, q)$ be the visibility edge of P . The aim is to partition P into a set of rwv-polygons.

An **ewv**-polygon P can be partitioned into **rwv**-polygons Q_1, \dots, Q_k by short-cutting the part of the perimeter of P going through vertices in the visibility polygon that are not vertices of P , as shown in Fig. 4. This can be done in $\mathcal{O}(n \log n)$ time as follows. Let $p = v_1, \dots, v_n = q$ be the vertices of P in counter-clockwise order along the perimeter of P . Traverse the vertices of P counter-clockwise starting at v_1 . The vertex v_1 is added to an empty list L . In a generic step vertex v_i is visited, if v_i can be seen from e then it is added to L otherwise it is discarded. When all the vertices of P have been traversed, the list is traversed and a segment is added between every consecutive pair of vertices that are not connected by an edge of P .

Lemma 2. *The algorithm requires $\mathcal{O}(n \log n)$ time and partitions an **ewv**-polygon into **rwv**-polygons.*

Proof. Since ray-shooting queries in a simple polygon can be answered in logarithmic time using $\mathcal{O}(n)$ preprocessing and space, see Hershberger and Suri [15], it is straight-forward to see that the total time to perform the partition step is $\mathcal{O}(n \log n)$.

It remains to prove the correctness. Let P be the given **ewv**-polygon and let Q_1, \dots, Q_k denote the resulting polygons after the partition of P . From the fact that P is an **ewv**-polygon it follows that each added segment “cuts” off a subpolygon from P , as illustrated in Fig. 4a. Let Q_1 be the **rwv**-polygon of P that contains the visibility edge (p, q) . Obviously Q_1 is an **rwv**-polygon since every part of P that is not seen by (p, q) is cut off. Let Q_2, \dots, Q_k be the remaining subpolygons ordered in clockwise order along the perimeter of Q_1 and P . Every subpolygon Q_i , $2 \leq i \leq k$, will have exactly one edge $e_i = (p_i, q_i)$ that partly can be seen by (p, q) , and we will show that Q_i is an **rwv**-polygon with respect to e_i . There are two cases; either one of e_i ’s endpoints is visible from (p, q) , or none of e_i ’s endpoints are visible from (p, q) .

In the first case let (p_i, u) be the edge shared by Q_1 and Q_i , as seen in Fig. 4b. According to the definition of an **ewv**-polygon it follows that u and q_i are connected by a concave path, or a straight-line segments. Thus, Q_i is a pseudo-triangle with at most one concave chain and, hence, also a **rwv**-polygon with respect to e_i .

In the second case let (u, u') be the edge shared by Q_1 and Q_i , see Fig. 4c. According to the definition of an **ewv**-polygon it follows that u and q_i , and u' and p_i , are connected by a concave path, or a straight-line segment. Thus, Q_i is a polygon with at most two concave chains and since u and u' are the only vertices of Q_i that are visible from (p, q) it follows that Q_i is an **rwv**-polygon with respect to e_i . \square

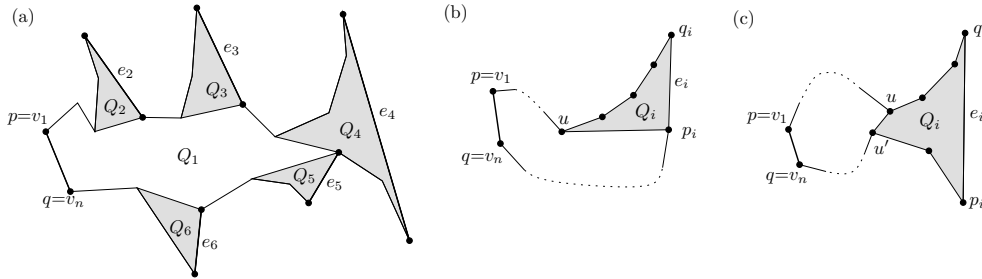


Fig. 4. (a) The partition of an **ewv**-polygon P into **rwv**-polygons. (b)-(c) Q_i is an **rwv**-polygon with respect to e_i .

Lemma 3. *Given an **ewv**-polygon with respect to e the algorithm adds segments of total length at most $wt(P) - wt(e)$.*

Proof. For each segment e' added by the algorithm, the part of the perimeter of P which is “cut off” by e' is longer than e' . Figure 4b illustrates the argument where the length of the added segment (u, p_i) is less than the length of e_i plus the length of $\delta(q_i, u)$, thus the lemma follows. \square

Putting together the above results gives us the following theorem.

Theorem 2. *Every simple polygon P can be partitioned into rwv -polygons in $\mathcal{O}(n \log n)$ time by adding segments of length $3 \cdot \text{wt}(P)$.*

Proof. When performing the second step of the algorithm we have to guarantee that the visibility edges defined in step 1 also are the visibility edges in step 2. This assures that the stated theorem follows by simply combining Lemmas 1-3, otherwise the length bound is increased to $4 \cdot \text{wt}(P)$. \square

2.2 Pseudo-triangulating an rwv -polygon

Next we show that an rwv -polygon can be pseudo-triangulated using segments of small total length.

Observation 3 *An rwv -polygon P can be pseudo-triangulated in $\mathcal{O}(n \log n)$ time using segments of total length $\mathcal{O}(\text{wt}(P) \cdot \log n)$.*

Proof. Let p_1, \dots, p_m be the convex vertices of P in counter-clockwise order and let (p_1, p_2) be the visibility edge of P . Since P is an rwv -polygon p_1 and p_2 are convex vertices. We will construct a pseudo-triangulation \mathcal{T} of P by adding a pseudo-triangle Δ within P . The segments of Δ partition P into smaller rwv -polygons that are recursively pseudo-triangulated.

Construct a pseudo-triangle Δ with corners at p_1, p_2 and $p_{\lceil m/2+1 \rceil}$, and add the edges of Δ to \mathcal{T} , see Fig. 5a. It holds that Δ is a pseudo-triangle since, according to Observation 2, there must be concave chains between p_1 and $p_{\lceil m/2+1 \rceil}$, and between p_2 and $p_{\lceil m/2+1 \rceil}$. The weight of the segments added to \mathcal{T} in this step is bounded by $(\text{wt}(P) - \text{wt}(p_1, p_2))$.

The segments of Δ partition P into a number of subpolygons, each with at most $\lceil m/2 + 1 \rceil$ convex vertices. Also, since p_1, p_2 and $p_{\lceil m/2+1 \rceil}$ are convex vertices in P it holds that each subpolygon is an rwv -polygon, i.e., a weak visibility polygon whose visibility edge has two convex vertices. This process continues recursively until every polygon is pseudo-triangulated, see Fig. 5b. It remains to bound the weight of the segments in \mathcal{T} . Consider the computation tree obtained from the recursion, in each internal node a pseudo-triangle is added to \mathcal{T} . Since a balanced partition is performed in each internal node the tree has height $\mathcal{O}(\log n)$. Finally, consider one level of the tree, the total weight of all edges added on one level is bounded by $\text{wt}(P)$, hence the observation follows. \square

This concludes the proof of Theorem 1.

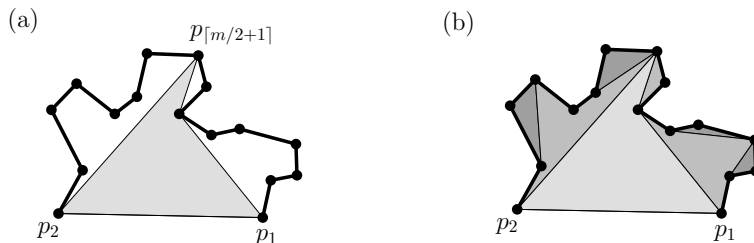


Fig. 5. (a) A pseudo-triangle can be found in an rwv -polygon. (b) A pseudo-triangle partitions an rwv -polygon into smaller rwv -polygons that can be pseudo-triangulated recursively.

3 An MWPT of a simple polygon

Even though the above algorithm is asymptotically worst-case optimal with respect to the weight of the minimum spanning tree it can be very far from the optimal solution. For example, often an optimal solution will have weight which is within a constant factor times the weight of a minimum weight spanning tree, which implies that the above algorithm will produce a solution which is a factor $\Theta(\log n)$ of the optimal. In the rest of this paper we will focus on developing a constant factor approximation algorithm for the MWPT-problem. As a subroutine we will also develop an algorithm that finds an optimal pseudo-triangulation of a simple polygon.

Theorem 3. *Given a simple polygon P one can compute the minimum weight pseudo-triangulation of P in $\mathcal{O}(n^3)$ time using $\mathcal{O}(n^2)$ space.*

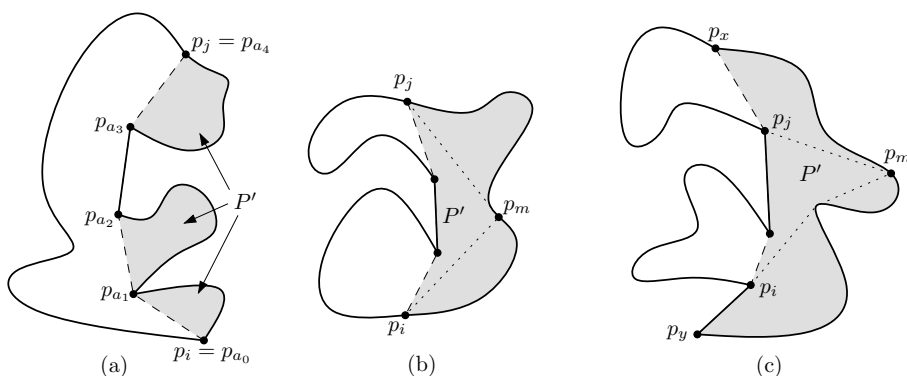


Fig. 6. (a) $\delta(p_i, p_j)$ is a convex path and P' is a degenerate polygon. (b) and (c) $\delta(p_i, p_j)$ is a concave path.

We will use a similar dynamic programming method as proposed by Gilbert [10] and Klincsek [18] for finding a minimum weight triangulation of a simple polygon. The basic observation used is that once some (pseudo-)triangle of the (pseudo-)triangulation has been fixed the problem splits into subproblems whose solutions can be found recursively, hence avoiding recomputation of common subproblems.

Let p_1, \dots, p_n be the vertices of P in counter-clockwise order. Let $\delta(p_i, p_j)$ be the shortest (directed) geodesic path from p_i to p_j within P . For each pair of vertices p_i and p_j of P we will in most cases compute three values; namely $L[i, j]$, $C[i, j]$ and $wt(\delta(p_i, p_j))$. Let P' be the (maybe degenerate) subpolygon of P bounded by the counter-clockwise path p_i, p_{i+1}, \dots, p_j along the boundary of P and the path $\delta(p_i, p_j)$, see Fig. 6. The value $L[i, j]$ is the total edge length of an optimal pseudo-triangulation of P' , while the value $C[i, j]$ is the total edge length of an optimal pseudo-triangulation of P' containing a pseudo-triangle with convex corners at p_i and p_j , as illustrated in Fig. 6b-c.

Define the *order* of a pair of points p_i, p_j to be the value $((n + j - i) \bmod n)$, i.e., the number of edges on the path from p_i to p_j along the perimeter of P in counter-clockwise order. Sort the pairs with respect to their order, ties are broken arbitrarily. Note that every pair of points p_i and p_j will occur twice; once as (p_i, p_j) and once as (p_j, p_i) . Process each pair in sorted order as follows.

Assume we are about to process (p_i, p_j) and that the path $\delta(p_i, p_j)$ goes through the vertices $\langle p_{a_0}, p_{a_1}, \dots, p_{a_{k-1}}, p_{a_k} \rangle$, where $p_i = p_{a_0}$ and $p_j = p_{a_k}$. As above we define P' to be the (maybe degenerate) subpolygon of P bounded by the counter-clockwise path p_i, p_{i+1}, \dots, p_j along the boundary of P and the path $\delta(p_i, p_j)$.

The path $\delta(p_i, p_j)$ is said to be convex if the open straight line segment between p_i and p_j lies to the right of $\delta(p_i, p_j)$. If the straight line segment lies to the left, or on, $\delta(p_i, p_j)$ then $\delta(p_i, p_j)$

is said to be concave, see Fig. 6. Compute $\delta(p_i, p_j)$ within P . This can be done in linear time according to Hershberger [14]. The weight of the reported path is stored.

1. If $\delta(p_i, p_j)$ is neither a convex path nor a concave path, i.e., the path zick-zacks, then set $L[i, j] = \infty$ and $C[i, j] = \infty$.
2. If $\delta(p_i, p_j) = (p_i, p_j)$ then $C[i, j]$ and $L[i, j]$ can be computed as follows. We start with $C[i, j]$. Any optimal pseudo-triangulation of P' that contains a pseudo-triangle τ with convex corners at p_i and p_j must have a vertex p_m , $i < m < j$, as the third convex corner. Testing a pseudo-triangle with corners at p_i, p_j and p_m takes constant time since all the $L[* , *]$ -values of the paths between p_i and p_m , and p_m and p_j have already been computed. This follows since the paths $\delta(p_i, p_m)$ and $\delta(p_j, p_m)$ are either single edges or convex paths. In both cases $L[i, m]$ and $L[j, m]$ have already been computed, see cases 2 and 3.

Next we compute $L[i, j]$. Consider an optimal pseudo-triangulation \mathcal{T} of P' and let τ be the pseudo-triangle in \mathcal{T} that includes (p_i, p_j) . The pseudo-triangle τ must have a concave chain with endpoints (convex corners) at p_x and p_y that includes the edge (p_i, p_j) , as a result $L[i, j] = C[x, y]$, as illustrated in Fig. 6c. Obviously x and y lie on the counter-clockwise path from p_i to p_j along P and therefore $C[x, y]$ has already been computed, see case 4. Testing all of the $\mathcal{O}(n^2)$ concave chains within P' that (p_i, p_j) may belong to requires $\mathcal{O}(n^2)$. Note however, that the total number of tests performed in this step can be bounded by $\mathcal{O}(n^3)$ by observing that there are $\mathcal{O}(n^2)$ concave chains and each chain may contain $\mathcal{O}(n)$ edges, thus the total bound is $\mathcal{O}(n^3)$.

3. If $\delta(p_i, p_j)$ is a convex path containing more than one edge then $L[p_i, p_j]$ and $C[i, j]$ can be computed in time $\mathcal{O}(n)$ as follows.

For every pair p_{a_l} and $p_{a_{l+1}}$, $i \leq a_l \leq j$, the value $L[p_{a_l}, p_{a_{l+1}}]$ has already been computed, see case 2 and Fig. 6a. The values can be added up in linear time, i.e., calculating $\sum_{\alpha=0}^{k-1} L[p_{a_\alpha}, p_{a_{\alpha+1}}]$. Since $\delta(p_i, p_j)$ is a convex path and it consists of several edges there is no pseudo-triangle within P' with convex corners at p_i and p_j , thus $C[i, j] = \infty$.

4. If $\delta(p_i, p_j)$ is a concave path containing more than one edge then $C[p_i, p_j]$ can be computed in linear time in the same way as described in case 2. For completeness we repeat the approach. Any optimal pseudo-triangulation of P' that contains a pseudo-triangle τ with convex corners at p_i and p_j must have a vertex p_m , $i < m < j$ as the third convex corner. Testing a pseudo-triangle with corners at p_i, p_j and p_m takes constant time since all the $L[* , *]$ -values of the paths between p_i and p_m , and p_m and p_j have already been computed, as illustrated in Fig. 6b-c. Computing the $L[i, j]$ value is too time consuming and it is not needed.

Note that case 2 and 4 use the L -values of case 2 and 3, and the C -value of case 4. Case 3 uses the L -values of case 3. Thus, all the needed values are computed.

By adding up the running times of each of the above cases the total running time is bounded by $\mathcal{O}(n^3)$. The space bound follows from the fact that for every pair of points p_i and p_j we store $L[i, j]$ and $C[i, j]$.

When all the $L[* , *]$ and the $C[* , *]$ have been computed we can easily test every possible pseudo-triangle in constant time as follows. Let τ be an arbitrary pseudo-triangle within P , and denote its three convex corners by p_i, p_j and p_k . The length of an optimal pseudo-triangulation of P that includes τ can be obtained by adding up the values:

$$L[i, j] + L[j, k] + L[k, i] + wt(\delta(p_i, p_j)) + wt(\delta(p_j, p_k)) + wt(\delta(p_k, p_i)).$$

Since all the six values are pre-computed Theorem 3 follows.

4 A constant factor approximation algorithm

In this section we will give an approximation algorithm for the MWPT-problem. It is similar to the approximation algorithm presented in Section 2 in the sense that the two main steps are the same; first a partition of the convex hull of the point set into simple polygons followed by

a pseudo-triangulation of each polygon. In the pseudo-triangulation step we will use the optimal algorithm presented in the previous section. As input we are given a set \mathcal{S} of n points in the plane, and as output we will produce a pseudo-triangulation \mathcal{T} of \mathcal{S} .

Algorithm PSEUDOTRIANGULATE(\mathcal{S})

1. Construct the convex hull and the minimum weight spanning tree of \mathcal{S} . This partitions $\mathcal{CH}(\mathcal{S})$ into simple (maybe degenerate) polygons denoted P_1, \dots, P_k .
2. Apply Theorem 3 to each of the k polygons. The pseudo-triangulation obtained together with the convex hull and the minimum spanning tree of \mathcal{S} is reported.

The aim of this section is to prove the following theorem.

Theorem 4. *Given a set of n points \mathcal{S} in \mathbb{R}^2 algorithm PSEUDOTRIANGULATE computes a pseudo-triangulation \mathcal{T} of \mathcal{S} in time $\mathcal{O}(n^3)$ using $\mathcal{O}(n^2)$ space such that $wt(\mathcal{T}) \leq 12 \cdot wt(\mathcal{T}_{opt})$, where \mathcal{T}_{opt} is a minimum weight pseudo-triangulation of \mathcal{S} .*

The running time of the algorithm is $\mathcal{O}(n^3)$ since the time-complexity is dominated by computing a MWPT of each polygon. Note that the algorithm produces a MWPT that includes $\mathcal{M}(\mathcal{S})$, thus it suffices to prove that there exists a pseudo-triangulation of \mathcal{S} that includes the edges in a minimum weight spanning tree of \mathcal{S} and whose weight is at most $12 \cdot wt(\mathcal{T}_{opt})$.

4.1 The weight of a pseudo-triangulation that includes a minimum spanning tree

In this section we will prove the following lemma, which completes the proof of Theorem 4.

Lemma 4. *Let \mathcal{S} be a set of n points in \mathbb{R}^2 and let \mathcal{T}_{opt} denote a MWPT of \mathcal{S} . There exists a pseudo-triangulation \mathcal{T} of \mathcal{S} that includes the edges of $\mathcal{M}(\mathcal{S})$ and whose weight is at most $12 \cdot wt(\mathcal{T}_{opt})$.*

We will need the following generalization of a pseudo-triangle.

Definition 1. *A simple polygon P is said to be a pseudo- k -gon if P includes exactly k convex vertices.*

The proof of Lemma 4 is performed in two steps. First it will be shown that one can construct a planar graph \mathcal{G} of the vertices of \mathcal{S} such that no edge of \mathcal{G} intersects an edge of $\mathcal{M}(\mathcal{S})$ (Lemma 5), every face of \mathcal{G} is a pseudo- k -gon for $3 \leq k \leq 6$ (Lemma 6), and the weight of \mathcal{G} is bounded by $4 \cdot wt(\mathcal{T}_{opt})$ (Lemma 7).

The second step shows how a pseudo- k -gon Q , $4 \leq k \leq 6$, can be partitioned into pseudo-triangles by adding $k - 3$ edges to Q of total weight at most $wt(Q)$ (Lemma 8). Since every edge may belong to one or two pseudo- k -gons the upper bound on the final weight is $(4 + 2 \cdot 4)wt(\mathcal{T}_{opt}) = 12 \cdot wt(\mathcal{T}_{opt})$.

Constructing \mathcal{G} . Initially \mathcal{G} contains the edges in $\mathcal{M}(\mathcal{S})$ and $\mathcal{CH}(\mathcal{S})$. Process every edge $e = (u, v)$ in \mathcal{T}_{opt} as follows. If e does not intersect any edge of $\mathcal{M}(\mathcal{S})$ then add e to \mathcal{G} . Otherwise assume for simplicity that e is vertical and that u lies above v . Let $f_1 = (p_1, q_1), \dots, f_m = (p_m, q_m)$ be the edges of $\mathcal{M}(\mathcal{S})$ that intersect e ordered with respect to their intersection with e from top to bottom. Furthermore, assume that p_i lies to the left of q_i and denote by x_i the intersection point between e and f_i . The following edges are now added to \mathcal{G} , as illustrated in Fig. 7.

- (1) If $|p_1x_1| < |q_1x_1|$ then the concave path $\lambda_0(e) = \delta(u, p_1)$ between u and p_1 , for which the region bounded by (u, v) , $\delta(u, p_1)$ and f_1 is empty, is added to \mathcal{G} , as shown in Fig. 7a. Otherwise, if $|q_1x_1| \leq |p_1x_1|$, the corresponding path between u and q_1 is added to \mathcal{G} .
- (2) If $|p_mx_m| < |q_mx_m|$ then the concave path $\lambda_m(e) = \delta(v, p_m)$ between v and p_m , for which the region bounded by (u, v) , $\delta(v, p_m)$ and f_m is empty, is added to \mathcal{G} . Otherwise, if $|q_mx_m| \leq |p_mx_m|$, the corresponding path between v and q_m is added to \mathcal{G} .

- (3) If $m \geq 2$ then for each $1 \leq i < m$ we will have four cases. Note that case (b) and (d) below are symmetric to (a) and (c) respectively. Let a_i be the endpoint of (p_i, q_i) closest to x_i .
- a. If $a_i = p_i$ and $a_{i+1} = p_{i+1}$ then the concave path $\lambda_i(e) = \delta(p_i, p_{i+1})$ between u_i and u_{i+1} for which the region bounded by (u, v) , $\delta(p_i, p_{i+1})$, f_i and f_{i+1} is empty is added to \mathcal{G} , as illustrated in Fig. 7b.
 - b. If $a_i = q_i$ and $a_{i+1} = q_{i+1}$ then the concave path $\lambda_i(e) = \delta(q_i, q_{i+1})$ between q_i and q_{i+1} for which the region bounded by (u, v) , $\delta(q_i, q_{i+1})$, f_i and f_{i+1} is empty is added to \mathcal{G} .
 - c. If $a_i = p_i$ and $a_{i+1} = q_{i+1}$ then the shortest path $\lambda_i(e) = \delta(p_i, q_{i+1})$ between p_i and q_{i+1} for which it holds that the two regions bounded by (u, v) , $\delta(p_i, q_{i+1})$, f_i and f_{i+1} are empty is added to \mathcal{G} , see Fig. 7c.
 - d. If $a_i = q_i$ and $a_{i+1} = p_{i+1}$ then the shortest path $\lambda_i(e) = \delta(q_i, p_{i+1})$ between q_i and p_{i+1} for which it holds that the two regions bounded by (u, v) , $\delta(q_i, p_{i+1})$, f_i and f_{i+1} are empty is added to \mathcal{G} .

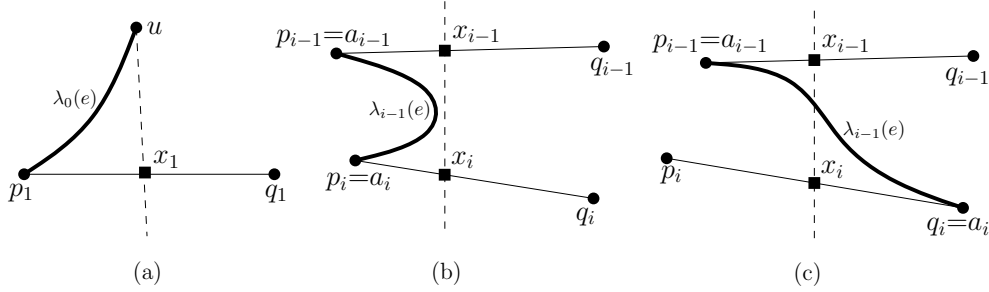


Fig. 7. The three cases that may occur when the partial minimum weight spanning tree edge (dashed) is replaced by a chain (fat).

Properties of \mathcal{G} . It remains to prove that \mathcal{G} is a planar spanning graph of \mathcal{S} (Lemma 5) and that every internal face of \mathcal{G} is a pseudo- k -gon (Lemma 6) before we can bound the weight of \mathcal{G} .

Lemma 5. \mathcal{G} is a planar spanning graph of \mathcal{S} .

Proof. We will use the same notation as in the construction of \mathcal{G} above. The graph is obviously a spanning graph since the edges in $\mathcal{M}(\mathcal{S})$ are added to \mathcal{G} in the first step of the algorithm.

It remains to prove that \mathcal{G} is planar. Note that \mathcal{G} contains three sets of edges that we distinguish between; the convex hull of \mathcal{S} , the edges in $\mathcal{M}(\mathcal{S})$ and the remaining edges of \mathcal{G} denoted E' . No edge on the convex hull of \mathcal{S} can intersect any other edge of \mathcal{G} , thus we may ignore the edges in the convex hull of \mathcal{S} . Furthermore, no edge of $\mathcal{M}(\mathcal{S})$ can intersect any other edge of $\mathcal{M}(\mathcal{S})$. It remains to prove that no edge in E' can intersect (1) any edge in $\mathcal{M}(\mathcal{S})$, or (2) any other edge in E' .

An edge $e = (u, v)$ in \mathcal{T}_{opt} is removed and replaced by $k + 1$ paths, $\lambda_0(e), \dots, \lambda_k(e)$, if and only if e intersects $k > 0$ edges of $\mathcal{M}(\mathcal{S})$.

To prove (1) it suffices to prove that no edge of $\mathcal{M}(\mathcal{S})$ can intersect $\lambda_i(e)$, $0 \leq i \leq k$. (Here we say that two edges intersect if and only if their open segments intersect in exactly one point.) The path $\lambda_i(e)$ connects the two edges f_i and f_{i+1} of $\mathcal{M}(\mathcal{S})$, where f_0 is the degenerate edge (u, u) and f_{k+1} is the degenerate edge (v, v) . If any edge f of $\mathcal{M}(\mathcal{S})$ would intersect $\lambda_i(e)$ then f would also intersect f_i , f_{i+1} or (u, v) , see Fig. 7. Obviously it cannot intersect f_i or f_{i+1} since they also are edges of $\mathcal{M}(\mathcal{S})$, and it cannot intersect (u, v) since then f would be the edge f_{i+1} which is a contradiction.

It remains to prove (2) by arguing that the path $\lambda_i(e)$ cannot intersect any edge $\mu \in E'$. Let $\delta(p_i, p_{i+1})$ denote the concave path between p_i and p_{i+1} such that the region bounded by f_i, f_{i+1}, e and $\delta(p_i, p_{i+1})$ is empty, and define $\delta(q_i, q_{i+1})$ symmetrically. The empty region between $f_i, f_{i+1}, \delta(p_i, p_{i+1})$ and $\delta(q_i, q_{i+1})$ is denoted $\mathcal{A}_i(e)$. Note that the set of edges that $\lambda_i(e)$ may contain is the edges on the path $\delta(p_i, p_{i+1})$, the edges on the path $\delta(q_i, q_{i+1})$, and two edges between the two paths, as illustrated in Fig. 8a.

It has already been shown that no edge of \mathcal{G} can intersect f_i or f_{i+1} hence if μ intersects $\mathcal{A}_i(e)$ then it must intersect $\delta(p_i, p_{i+1})$ and $\delta(q_i, q_{i+1})$ and, as a consequence, μ must intersect (u, v) .

In the case when μ is an edge of \mathcal{T}_{opt} then μ cannot intersect (u, v) since they are both edges of the planar graph \mathcal{T}_{opt} .

The only remaining case is when μ is part of a path $\lambda_j(e')$ in \mathcal{G} between two edges f'_j and f'_{j+1} of $\mathcal{M}(\mathcal{S})$. Using the corresponding notation as above it holds that there is an empty region $\mathcal{A}_j(e')$ that is bounded by $f'_j, f'_{j+1}, \delta(p'_j, p'_{j+1})$ and $\delta(q'_j, q'_{j+1})$, as illustrated in Fig. 8b. Furthermore, f'_j and f'_{j+1} cannot intersect e between f_i and f_{i+1} since then f_i and f_{i+1} would not be consecutive along e which is a contradiction. Using the above arguments it follows that e and e' must intersect which is again a contradiction since both e and e' belongs to \mathcal{T}_{opt} which is a planar graph.

This completes the proof of the lemma. \square

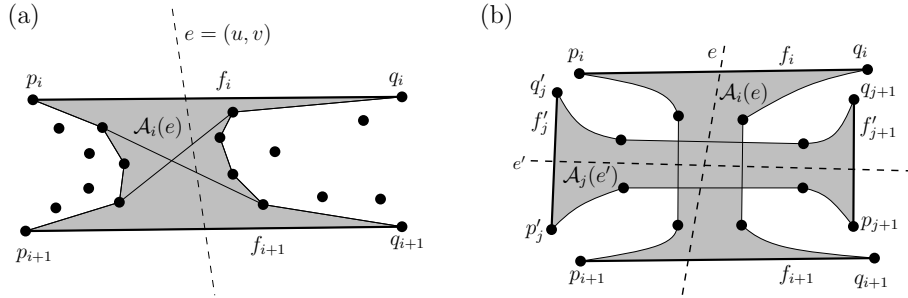


Fig. 8. Illustrating the proof of Lemma 5.

Lemma 6. *Each internal face of \mathcal{G} is a pseudo- k -gon, for $3 \leq k \leq 6$.*

Proof. Consider the arrangement \mathcal{A} of the edges in \mathcal{T}_{opt} and $\mathcal{M}(\mathcal{S})$, and an internal face f of \mathcal{A} . Note that every edge of $\mathcal{M}(\mathcal{S})$ “cuts” off a convex vertex from the pseudo-triangle τ in \mathcal{T}_{opt} that f belongs to, hence f is a pseudo- k -gon, where $3 \leq k \leq 6$, as illustrated in Fig. 9a.

More precisely, every convex corner of f induces at most one convex corner of \mathcal{G} . Consider an edge $e = (u, v)$ of $\mathcal{M}(\mathcal{S})$ that intersects τ and let $e_1 = (x_1, y_1)$ and $e_2 = (x_2, y_2)$ be two edges of τ , adjacent along e , that intersect e . If u is a vertex of τ then set $u = y_1$ and if v is a vertex on τ then set $v = y_2$. Let u' and v' be the convex corners of f at e closest to u and v respectively, as shown in Fig. 9b-c. Assume that u' lies on e_1 and v' on e_2 .

We will have two cases as shown in Fig. 9b and 9c.

1. If u' lies closer to u than to v and v' lies closer to v than to u , then x_1 is connected to u by a concave path and x_2 is connected to v by a concave path, according to the algorithm. Hence f has a convex corner at u and one at v . No other convex corners can arise when adding the two paths.
2. In the case when v' lies closer to u than to v , then x_1 is connected to u by a concave path and x_2 is connected to u by a concave path. These paths may partly coincide, thus assume they meet at a point x (which may be u). Now, f has a convex corner at x and possibly one at x_2 . No other convex corners can arise when adding the two paths. Note that the case when u' lies closer to v than to u is symmetrical.

Hence, there is at most one convex corner in \mathcal{G} for every convex corner in f . Note that a face f may, in some cases, collapse when an edge is replaced by a path in \mathcal{G} . \square

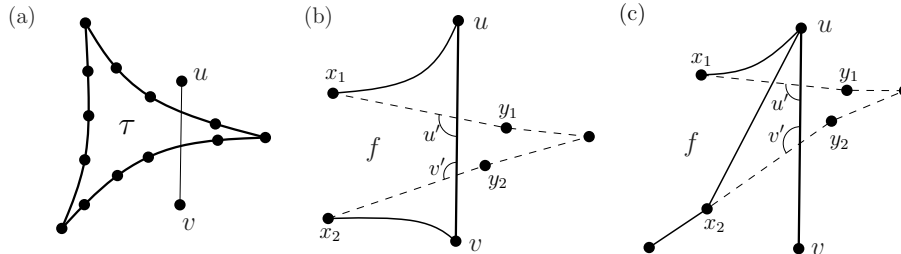


Fig. 9. (a) An edge of a minimum spanning tree that intersects a pseudo-triangle τ cuts off a convex corner of τ . (b) f includes both u and v . (c) f only includes v .

Lemma 7. $wt(\mathcal{G}) \leq 4 \cdot wt(\mathcal{T}_{opt})$.

Proof. The idea of the proof is to show that the weight of every path in \mathcal{G} corresponding to an edge e in \mathcal{T}_{opt} can be charged to the weight of e .

Consider any edge $e = (u, v)$ of \mathcal{T}_{opt} . If e does not intersect any edges of $\mathcal{M}(\mathcal{S})$ then we are done since e also is an edge in \mathcal{T}_{opt} . Otherwise consider the edges added to \mathcal{G} when e is processed in the construction of \mathcal{G} . Using the same notation as in the construction algorithm, it holds that the path added between u and v can be seen as $m + 1$ subpaths, $\lambda_0(e), \dots, \lambda_m(e)$, as shown in Fig. 7.

Let e_i denote the part of e between f_i and f_{i+1} . Consider an edge (x, y) of $\mathcal{M}(\mathcal{S})$, and let $\mathcal{D}(x, |xy|)$ and $\mathcal{D}(y, |xy|)$ be the discs with radius $|xy|$ and with center at x and y respectively. From the geometric properties of an edge of $\mathcal{M}(\mathcal{S})$ it holds that the intersection of $\mathcal{D}(x, |xy|)$ and $\mathcal{D}(y, |xy|)$ must be empty of points which implies that $|ux_1| > \min(|x_1p_1|, |x_1q_1|)$ and that $|v, x_m| > \min(|x_m, u_m|, |x_m, v_m|)$, as shown in Fig. 10a. It holds

$$wt(\lambda_0(e)) < |ux_1| + |x_1a_1| < 2|ux_1| = 2|e_0|,$$

and

$$wt(\lambda_m(e)) < |ux_m| + |x_ma_m| < 2|ux_m| = 2|e_m|.$$

Next consider $\lambda_i(e)$ where $1 \leq i < m$. Since each $\lambda_i(e)$ is a concave chain it holds that

$$\sum_{1 \leq i < m} wt(\lambda_i(e)) < |e| - (|e_0| + |e_m|) + \sum_{1 \leq i < m} (|a_i x_i| + |a_{i+1} x_{i+1}|).$$

It will be shown that the last term can be bounded by $2(|e| - |e_0| - |e_m|)$. Again we are going to use an emptiness property of the minimum spanning tree. There will be four cases; the path from f_i to f_{i+1} passes through p_i and p_{i+1} , q_i and q_{i+1} , p_i and q_{i+1} , or q_i and p_{i+1} . Assume that the shortest path in $\mathcal{M}(\mathcal{S})$ between p_i and p_{i+1} traverse both f_i and f_{i+1} , as illustrated in Fig. 10b. Similar arguments can be applied to each of the three other cases. Since f_i and f_{i+1} are edges of $\mathcal{M}(\mathcal{S})$ it holds that p_i and q_i cannot lie in $\mathcal{D}(p_{i+1}, |p_{i+1}q_{i+1}|)$, and p_{i+1} and q_{i+1} cannot lie within $\mathcal{D}(p_i, |p_iq_i|)$. Without loss of generality we can assume that f_i is at least as long as f_{i+1} .

Straight-forward geometry shows that the ratio $\frac{|e_i|}{|a_i x_i| + |a_{i+1} x_{i+1}|}$ is minimized when $q_i = q_{i+1}$ and the angle between f_i and f_{i+1} is minimized, which is the case when the length of f_{i+1} is equal to the length of f_i and q_i lies on the perimeter of $\mathcal{D}(p_i, |p_iq_i|)$. Furthermore, the ratio is minimized when the distance between x_i and q_i is equal to the distance between q_{i+1} and x_{i+1} , which in turn

is equal to the distance between x_i and x_{i+1} . As a result it follows that $|x_i a_i| + |a_{i+1} x_{i+1}| \leq 2|e_i|$, and thus

$$\begin{aligned}
\sum_{0 \leq i \leq m} wt(\lambda_i(e)) &< 2(|e_0| + |e_m|) + \sum_{1 \leq i < m} wt(\lambda_i(e)) \\
&\leq 2(|e_0| + |e_m|) + (|e| - |e_0| - |e_m|) + \sum_{1 \leq i < m} (|a_i x_i| + |a_{i+1} x_{i+1}|) \\
&\leq (|e| + |e_0| + |e_m|) + 2 \sum_{1 \leq i < m} |e_i| \\
&< 3|e|
\end{aligned}$$

The lemma is obtained adding up the above bound with $wt(\mathcal{M}(\mathcal{S}))$. \square

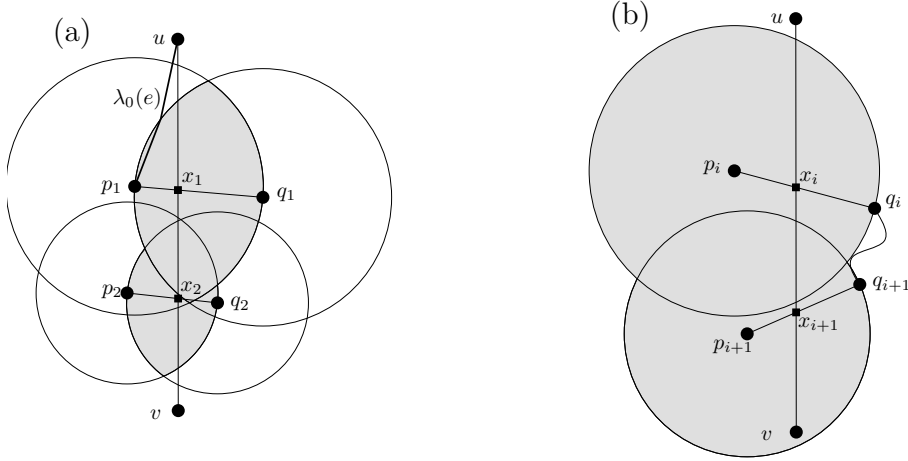


Fig. 10. (a) Illustrating that $|ux_1| > \min(|x_1 p_1|, |x_1 q_1|)$. (b) The shortest path in $\mathcal{M}(\mathcal{S})$ between p_i and p_{i+1} traverses both f_i and f_{i+1} .

It remains to show how the resulting pseudo- k -gons, $3 < k \leq 6$, can be pseudo-triangulated. Note that the pseudo- k -gons in \mathcal{G} are very special in the sense that $k - 3$ of the convex chains are straight-line segments and they are connected to concave chains that may or may not be straight-line segments. We call these *restricted* pseudo- k -gons, see Fig. 11. To complete the proof of Lemma 4 we end this section with the following lemma, which also completes the proof of Theorem 4.

Lemma 8. *For any $3 < k \leq 6$ it holds that a restricted pseudo- k -gon P can be pseudo-triangulated in $\mathcal{O}(n)$ time by adding $k - 3$ edges of total weight at most $wt(P)$.*

Proof. Consider the concave chains C_1, \dots, C_k of P in clockwise order, where at least $k - 3$ of the chains are straight-line segments. Denote the convex corners of P by v_1, \dots, v_k . Consider the different values of k .

$k = 4$: There is always a diagonal of P (actually at least two) that partitions P into two pseudo-triangles, as shown in Fig. 11a. If an edge (u, v) is added then the weight is bounded by the shortest path along the perimeter of P between u and v , hence at most $1/2 \cdot wt(P)$.

$k = 5$: There is always a diagonal e of P that partitions P into one pseudo-triangle P_1 and one pseudo-4-gon P_2 , as shown in Fig. 11b. Then P_2 is partitioned as in the case $k = 4$. The weight of the added diagonals is $wt(e) + 1/2 \cdot wt(P_2) < wt(P)$.

$k = 6$: There are at least three concave chains of P that are straight-line segments, as shown in Fig. 11c. Assume w.l.o.g. that it is $C_1 = (v_1, v_2)$, $C_3 = (v_3, v_4)$ and $C_5 = (v_5, v_6)$. Add the shortest geodesic path within P between v_1 and v_3 , between v_3 and v_5 and, between v_5 and v_1 . Obviously the weight of the three paths is bounded by the weight of P . We claim that the added edges partition P into four pseudo-triangles. Let P_1 be the face in the partition of P containing v_2 , let P_2 be the face containing v_4 , let P_3 be the face containing v_6 , and finally, let P_4 be the remaining face, as shown in Fig. 11c.

Note that P_1 contains a path from v_1 to a point x_1 on C_2 , and that this path must be concave, otherwise it can be shortened. The path from v_2 to x_1 follows C_2 and is therefore also concave. It follows that P_1 is a pseudo-triangle. Similar arguments can be used to show that P_2 and P_3 also are pseudo-triangles. It remains to consider P_4 . Every chain of P_4 is a shortest path, and must be concave, otherwise the chain could be shortened. □

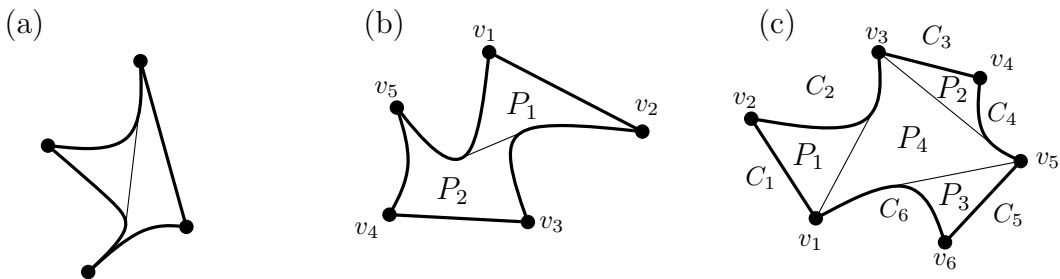


Fig. 11. For any $3 < k \leq 6$ it holds that a pseudo- k -gon can be pseudo-triangulated by adding $k - 3$ edges of total weight $wt(P)$.

5 Open problems

An obvious question is whether the minimum weight pseudo-triangulation problem is NP-hard. Is it as hard as finding the minimum weight triangulation? Computing the minimum weight triangulation is one of the few open problems listed in Garey and Johnson's 1979 book on NP-completeness [9] that remain open today.

A second open problem concerning the weight of a pseudo-triangulation is if there exists a minimum pseudo-triangulation of low weight. It was shown by Streinu [27] that every point set allows a minimum planar pseudo-triangulation that has $2n - 3$ edges. Neither of the two algorithms presented in this paper produces minimum pseudo-triangulations, although the dynamic programming algorithm for simple polygons can be modified to compute a minimum weight minimum pseudo-triangulation.

6 Acknowledgements

The authors would like to thank Mattias Andersson, Mark de Berg and Bettina Speckmann for valuable discussions during the work of this paper. We would also like to thank the two referees of this journal that found several technical problems in an earlier version.

References

1. O. Aichholzer, M. Hoffmann, B. Speckmann, and C. D. Tóth. Degree Bounds for Constrained Pseudo-Triangulations. Proc. 15th Canadian Conference on Computational Geometry, pp. 155-158, 2003.

2. O. Aichholzer, D. Orden, F. Santos, and B. Speckmann. On the Number of Pseudo-Triangulations of Certain Point Sets. Proc. 15th Canadian Conference on Computational Geometry, pp. 141-144, 2003.
3. O. Aichholzer, G. Rote, B. Speckmann, and I. Streinu. The Zigzag Path of a Pseudo-Triangulation. Proc. 8th International Workshop on Algorithms and Data Structures, pp. 377-388, Lecture Notes in Computer Science 2748, Springer Verlag, 2003.
4. J. Basch, L. J. Guibas, J. Hershberger, and L. Zhang. Deformable free space tiling for kinetic collision detection. Proc. 4th Workshop on Algorithmic Foundations of Robotics, 2000.
5. M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf. Computational Geometry: Algorithms and Applications, 2nd edition. Springer-Verlag, Berlin, Germany, 2000.
6. B. Chazelle. Triangulating a Simple Polygon in Linear Time. Discrete & Computational Geometry, 6:485-524, 1991.
7. B. Chazelle, H. Edelsbrunner, M. Grigni, L. J. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. Algorithmica, 12:54-68, 1994.
8. G. Das and D. Joseph. Which triangulations approximate the complete graph? In Proc. International Symposium on Optimal Algorithms, pp. 168-192, Lecture Notes in Computer Science 401, Springer Verlag, 1989.
9. M. Garey and D. Johnson. Computers and Intractability. W. H. Freeman and Company, 1979.
10. P. D. Gilbert. New results in planar triangulations. Report R-850, Univ. Illinois Coordinated Science Lab, 1979.
11. M. T. Goodrich and R. Tamassia. Dynamic Ray Shooting and Shortest Paths in Planar Subdivisions via Balanced Geodesic Triangulations. Journal of Algorithms, 23(1):51-73, 1997.
12. L. Guibas and J. Hershberger. Optimal shortest path queries in a simple polygon. Journal of Computer System Science, 39:126-152, 1989.
13. L. 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.
14. J. Hershberger. A new data structure for shortest path queries in a simple polygon. Information Processing Letters, 38(5):231 - 235, 1991.
15. J. Hershberger and S. Suri. A Pedestrian Approach to Ray Shooting: Shoot a Ray, Take a Walk. Journal of Algorithms, 18(3): 403-431, 1995.
16. D. Kirkpatrick, J. Snoeyink, and B. Speckmann. Kinetic Collision Detection for Simple Polygons. International Journal of Computational Geometry and Applications, 12(1&2):3-27, 2002.
17. D. Kirkpatrick and B. Speckmann. Kinetic Maintenance of Context-Sensitive Hierarchical Representations for Disjoint Simple Polygons. Proc. 18th ACM Symposium on Computational Geometry, pp. 179-188, 2002.
18. G. Klincsek. Minimal triangulations of polygonal domains. Annals of Discrete Mathematics, 9:121-123, 1980.
19. D. Krzrnaric and C. Levkopoulos. Quasi-Greedy Triangulations Approximating the Minimum Weight Triangulation. Journal of Algorithms 27(2): 303-338. 1998.
20. W. Lenhart and G. Liotta. Drawing Outerplanar Minimum Weight Triangulations. Information Processing Letters, 57(5):253-260, 1996.
21. D. A. Plaisted and J. Hong. A heuristic triangulation algorithm. Journal of Algorithms 8:405-437, 1987.
22. M. Pocchiola and G. Vegter. Pseudo-triangulations: Theory and applications. In Proc. 12th ACM Symposium on Computational Geometry, pp. 291-300, 1996.
23. M. Pocchiola and G. Vegter. Topologically sweeping visibility complexes via pseudo-triangulations. Discrete Computational Geometry, 16(4):419-453, 1996.
24. M. Pocchiola and G. Vegter. Minimal tangent visibility graphs. Computational Geometry Theory & Applications, 6(5):303-314, 1996.
25. G. Rote, C. A. Wang, L. Wang, and Y. Xu. On constrained minimum pseudotriangulations. Proc. 9th Symposium on Computing an Combinatorics, pp. 445-454, Lecture Notes in Computer Science 2697 Springer Verlag, 2003.
26. B. Speckmann and C. D. Tóth. Allocating Vertex pi-guards in Simple Polygons via Pseudo-Triangulations. Proc. 14th ACM-SIAM Symposium on Discrete Algorithms, pp. 109-118, 2003.
27. I. Streinu. A Combinatorial Approach to Planar Non-Colliding Robot Arm Motion Planning. Proc. 41st ACM Annual Symposium on Foundations of Computer Science, pp. 443-453, 2000.
28. P. Yoeli. Compilation of data for computer-assisted relief cartography. In J. Davis, and M. McCullagh, eds., Display and Analysis of Spatial Data. John Wiley & Sons, New York, 1975.