The Minimum Weight Triangulation Problem with Few Inner Points

Michael Hoffmann and Yoshio Okamoto[⋆]

Institute of Theoretical Computer Science, ETH Zurich, CH-8092 Zurich, Switzerland
E-mail: {hoffmann,okamotoy}@inf.ethz.ch

Abstract. We propose to look at the computational complexity of 2-dimensional geometric optimization problems on a finite point set with respect to the number of inner points (that is, points in the interior of the convex hull). As a case study, we consider the minimum weight triangulation problem. Finding a minimum weight triangulation for a set of n points in the plane is not known to be NP-hard nor solvable in polynomial time, but when the points are in convex position, the problem can be solved in $O(n^3)$ time by dynamic programming. We extend the dynamic programming approach to the general problem and describe an exact algorithm which runs in $O(6^k n^5 \log n)$ time where n is the total number of input points and k is the number of inner points. If k is taken as a parameter, this is a fixed-parameter algorithm. It also shows that the problem can be solved in polynomial time if $k = O(\log n)$. In fact, the algorithm works not only for convex polygons, but also for simple polygons with k interior points.

1 Introduction

A lot of NP-hard optimization problems on graphs can be solved in polynomial time when the input is restricted to partial k-trees, that is, graphs with treewidth at most k, where k is fixed. In this sense, the treewidth is regarded as a natural parameter to measure the complexity of graphs. This is based on the observation that "some NP-hard optimization problems on graphs are easy when the class is restricted to trees."

We try to address the following question: What is a natural parameter that could play a similar role for geometric problems as the treewidth does for graph problems? One basic observation is that "some NP-hard optimization problems on a point set in the Euclidean plane are easy when the points are in convex position." Therefore, the number of inner points can be regarded as a natural parameter for the complexity of geometric problems. Here, an inner point is a point in the interior of the convex hull of the given point set.

In this paper, we concentrate on one specific problem: the minimum weight triangulation problem. The minimum weight triangulation is a triangulation with minimum total length of edges. Triangulations have numerous applications in finite element methods, interpolation and graphics, to name just a few. In applications one is usually interested in finding a triangulation that is optimal in a certain sense. Among several criteria, a minimum weight triangulation is one of the most natural ones.

[⋆] Supported by the Berlin-Zürich Joint Graduate Program "Combinatorics, Geometry, and Computation" (CGC), financed by ETH Zürich and the German Science Foundation (DFG).

The minimum weight triangulation problem is notorious as one of the problems which are not known to be NP-hard nor solvable in polynomial time for a long time [8]. However, when the points are in convex position, the problem can be solved in polynomial time by dynamic programming. The main result in this paper is an exact algorithm to compute the minimum weight triangulation in $O(6^k n^5 \log n)$ time, where n is the total number of input points and k is the number of inner points. From the viewpoint of parameterized complexity [7, 14] this is a fixed-parameter algorithm if k is taken as a parameter.[1] Furthermore, the algorithm implies that the problem can be solved in polynomial time if $k = O(\log n)$.

Actually, our algorithm also works for simple polygons with inner points. Or, rather we should say that the algorithm is designed for such objects, and as a special case, we can compute a minimum weight triangulation of a point set. This digression to simple polygons is essential because our strategy is based on recursion and in the recursion we cannot avoid dealing with simple polygons.

Related work Since the literature on the minimum weight triangulation problem is vast, we just mention some articles that are closely related to ours. As already mentioned, finding a minimum weight triangulation of a finite point set is not known to be NP-hard nor solvable in polynomial time [8]. For an n-vertex convex polygon, the problem can be solved in $O(n^3)$ using dynamic programming. For an n-vertex simple polygon, Gilbert [9] and Klincsek [12] independently gave a dynamic-programming algorithm running in $O(n^3)$ time. But with inner points the problem seems more difficult. Another polynomial-time solvable case was discussed by Anagnostou and Corneil [3]: they considered the case where a given point set lies on a constant number of nested convex hulls. As for exact algorithms for the general case, Kyoda, Imai, Takeuchi & Tajima [11] took an integer programming approach and devised a branch-and-cut algorithm. Aichholzer [1] introduced the concept of a "path of a triangulation," which can be used to solve any kinds of "decomposable" problems (in particular the minimum weight triangulation problem) by recursion. These algorithms were not analyzed in terms of worst-case time complexity. As for approximation of minimum weight triangulations, Levcopoulos & Krznaric [13] gave a constant-factor polynomial-time approximation algorithm, but with a huge constant. As for the parameterization with respect to the number of inner points, the two-dimensional Euclidean traveling salesman problem was recently shown to be fixed-parameter tractable [6].

2 Preliminaries and description of the result

We start our discussion with introduction of some notations and definitions used in the paper. Then we state our result in a precise manner. From now on, we assume that input points are in general position, that is, no three points are on a single line and no two points have the same x-coordinate.

[1] A *fixed-parameter algorithm* has running time $O(f(k)\text{poly}(n))$, where n is the input size, k is a parameter and $f : \mathbb{N} \to \mathbb{N}$ is an arbitrary function. For example, an algorithm with running time $O(440^k n)$ is a fixed-parameter algorithm whereas one with $O(n^k)$ is not.

The line segment connecting two points $p, q \in \mathbb{R}^2$ is denoted by $\overline{pq}$. The length of a line segment $\overline{pq}$ is denoted by $\mathsf{length}(\overline{pq})$, which is measured by the Euclidean distance. A *polygonal chain* is a planar shape described as $\gamma = \bigcup_{i=0}^{\ell-1} \overline{p_i p_{i+1}}$ where $p_0, \ldots, p_\ell \in \mathbb{R}^2$ are distinct points except that p_0 and p_ℓ can be identical (in such a case, the chain is *closed*). For a closed polygonal chain we assume in the following that all indices are taken modulo ℓ.

The length of γ is the sum of the lengths of the line segments, that is, $\mathsf{length}(\gamma) = \sum_{i=0}^{\ell-1} \mathsf{length}(\overline{p_i p_{i+1}})$. We say γ is *selfintersecting* if there exists $i, j \in \{0, \ldots, \ell-1\}$, $i \neq j$, such that $(\overline{p_i p_{i+1}} \cap \overline{p_j p_{j+1}}) \setminus \{p_i, p_{i+1}, p_j, p_{j+1}\} \neq \emptyset$. Otherwise, we say γ is *non-selfintersecting*. The points $p_0, \ldots, p_\ell$ are the *vertices* of γ. When γ is not closed, p_0 and p_ℓ are called the *endpoints* of γ. In this case, we say γ *starts from* p_0 (or p_ℓ).

A *simple polygon* P is a simply connected compact region in the plane bounded by a closed non-selfintersecting polygonal chain. A *vertex* of P is a vertex of the polygonal chain which is the boundary of P. We denote the set of vertices of P by $\mathsf{Vert}(P)$. A *neighbor* of a vertex $p \in \mathsf{Vert}(P)$ is a vertex $q \in \mathsf{Vert}(P)$ such that the line segment $\overline{pq}$ lies on the boundary of P.

Following Aichholzer, Rote, Speckmann & Streinu [2], we call a pair $\Pi = (S, P)$ a *pointgon* when P is a simple polygon and S is a finite point set in the interior of P. We call S the set of *inner points* of Π. The *vertex set* of Π is $\mathsf{Vert}(P) \cup S$, and denoted by $\mathsf{Vert}(\Pi)$. Fig. 1 shows an example of pointgons.

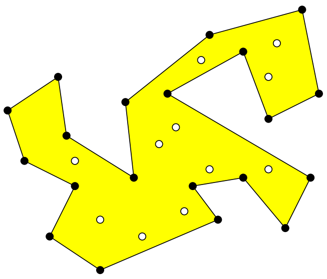


Fig. 1: A pointgon $\Pi = (S, P)$. In this paper, the points of S are drawn by empty circles and the points of $\mathsf{Vert}(P)$ are drawn by solid circles.

Let $\Pi = (S, P)$ be a pointgon. A *triangulation* $\mathcal{T}$ of a pointgon $\Pi = (S, P)$ is a subdivision of P into triangles whose edges are straight line segments connecting two points from $\mathsf{Vert}(\Pi)$ and which have no point from $\mathsf{Vert}(\Pi)$ in the interiors. The *weight* of $\mathcal{T}$ is the sum of the edge lengths used in $\mathcal{T}$. (Especially, all segments on the boundary of P are used in any triangulation and counted in the weight.) A *minimum weight triangulation* of a pointgon Π is a triangulation of Π which has minimum weight among all triangulations.

In this paper, we study the problem of computing a minimum weight triangulation of a given pointgon $\Pi = (S, P)$. The input size is proportional to $|\mathsf{Vert}(\Pi)|$. For a simple polygon, it is known that a minimum weight triangulation can be found in polynomial time [9, 12]. However, in spite of the simplicity of the problem, the minimum weight triangulation problem for general pointgons is not known to be solvable in polynomial time nor to be NP-hard [8]. Our goal is to find an exact algorithm for a pointgon $\Pi = (S, P)$ where $|S|$ is small. The main theorem of this work is as follows.

Theorem 1. *Let* $\Pi = (S, P)$ *be a pointgon. Let* $n := |\mathsf{Vert}(\Pi)|$ *and* $k := |S|$*. Then we can find a minimum weight triangulation of* Π *in* $O(6^k n^5 \log n)$ *time. In particular, if* $k = O(\log n)$ *then a minimum weight triangulation can be found in polynomial time.*

This theorem shows that, in the terminology of parameterized complexity, the problem is fixed-parameter tractable, when the size of S is taken as a parameter.

In the next section, we prove this theorem by providing an algorithm.

3 A fixed-parameter algorithm for minimum weight triangulations

First, we describe a basic strategy for our algorithm. The details are then discussed in Sections 3.2 and 3.3.

3.1 Basic strategy

In the sequel, for a given pointgon $\Pi = (S, P)$, we set $n := |\mathsf{Vert}(\Pi)|$ and $k := |S|$.

An *inner path* of a pointgon $\Pi = (S, P)$ is a polygonal chain $\gamma = \bigcup_{i=0}^{\ell-1} \overline{p_i p_{i+1}}$ such that $p_0, \dots, p_\ell$ are all different, $p_i \in S$ for each $i \in \{1, \dots, \ell-1\}$, $p_0, p_\ell \in \mathsf{Vert}(P)$, and $\gamma \setminus \{p_0, p_\ell\} \subseteq P$. An inner path $\bigcup_{i=0}^{\ell-1} \overline{p_i p_{i+1}}$ is called x-*monotone* if the x-coordinates of $p_0, \dots, p_\ell$ are either increasing or decreasing.

The basic fact we are going to use is the following.

Observation 2. *Let* $\Pi = (S, P)$ *be a pointgon and* p *be a vertex of* Π *with the smallest* x*-coordinate. Denote by* p', p'' *the neighbors of* p *in* P*. Then, for every triangulation* $\mathcal{T}$ *of* Π*, either*

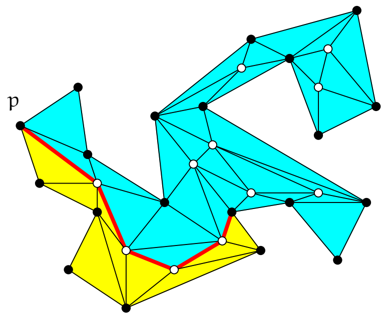
(1) there exists a non-selfintersecting x*-monotone inner path starting from* p *and consisting of edges of* $\mathcal{T}$*, or*
(2) the three points p, p', p'' *form a triangle of* $\mathcal{T}$*.*

The situation in Observation 2 is illustrated in Fig. 2.

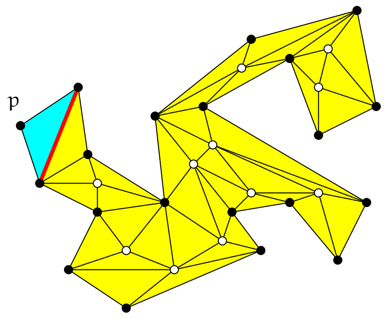
We would like to invoke Observation 2 for our algorithm.

Let $\Pi = (S, P)$ be a pointgon, and $p \in \mathsf{Vert}(P)$ a vertex with the smallest x-coordinate. A non-selfintersecting x-monotone inner path divides a pointgon into two smaller pointgons. (See Fig. 2(a) and recall the general position assumption.) Hence, by looking at all non-selfintersecting x-monotone inner paths, we can recursively solve the minimum weight triangulation problem. To establish an appropriate recursive formula, denote by $\mathcal{D}(p)$ the set that consists of the line segment $\overline{p'p''}$ and of all non-selfintersecting x-monotone inner paths starting from p. Each non-selfintersecting inner path $\gamma \in \mathcal{D}(p)$ divides our pointgon Π into two smaller pointgons, say Π'_γ and Π''_γ. Then, we can see that

$$\mathsf{mwt}(\Pi) = \min_{\gamma \in \mathcal{D}(p)} (\mathsf{mwt}(\Pi'_\gamma) + \mathsf{mwt}(\Pi''_\gamma) - \mathsf{length}(\gamma)). \tag{1}$$



(a) Case (1).



(b) Case (2).

Fig. 2: Situations in Observation 2.

To see that Eq. (1) is really true, the following observation should be explicitly mentioned, although the proof is straightforward thus omitted.

Observation 3. *Let* $\Pi = (S, P)$ *be a pointgon and* $\mathcal{T}$ *be a minimum weight triangulation of* Π*. Choose an inner path* γ *which uses edges of* $\mathcal{T}$ *only, and let* Π' *and* Π'' *be two smaller pointgons obtained by subdividing* Π *with respect to* γ*. Then, the restriction of* $\mathcal{T}$ *to* Π' *is a minimum weight triangulation of* Π'*. The same holds for* Π'' *as well.*

Therefore, by solving Recursion (1) with an appropriate boundary (or initial) condition, we can obtain a minimum weight triangulation of Π. Note that even if Π is a convex pointgon, the pointgons Π'_γ and Π''_γ encountered in the recursion might not be convex. Thus, our digression to simple polygons is essential also for the minimum weight triangulation problem for a finite point set, i.e., a convex pointgon.

3.2 Outline of the algorithm

Now, we describe how to solve Recursion (1) with the dynamic-programming technique.

First, let us label the elements of $\mathsf{Vert}(P)$ in a cyclic order, i.e., the order following the appearance along the boundary of P. According to this order, let us denote $\mathsf{Vert}(P) = \{p_0, p_1, \dots, p_{n-k-1}\}$. Then, pick a vertex $p_i \in \mathsf{Vert}(P)$, and consider a non-selfintersecting x-monotone inner path γ starting from p_i. Let $p_j \in \mathsf{Vert}(P)$ be the other endpoint of γ. Note that $\mathsf{Vert}(\gamma) \setminus \{p_i, p_j\}$ consists of inner points of Π only. Therefore, such a path can be uniquely specified by a subset $T \subseteq S$. That is, we associate a triple (i, j, T) with an x-monotone inner path $\overline{p_i q_1} \cup \overline{q_1 q_2} \cup \cdots \cup \overline{q_{t-1} q_t} \cup \overline{q_t p_j}$ where $T = \{q_1, q_2, \dots, q_t\}$. For the sake of brevity we write $\gamma(T)$ to denote the inner path associated with T when the endpoints p_i, p_j are clear from the context.

For two vertices $p_i, p_j \in \mathsf{Vert}(P)$ on the boundary of Π, and a set $T \subseteq S$ of inner points, let $\Pi(i, j, T)$ be the pointgon obtained from Π as follows: the boundary polygon is the union of the polygonal chains $\bigcup_{\ell=i}^{j-1} \overline{p_\ell p_{\ell+1}}$ and $\gamma(T)$. (Note that we only consider the case where $\gamma(T)$ is well-defined, that is, it does not intersect the boundary polygon.) The inner points of $\Pi(i, j, T)$ consist of the inner points of Π contained in the boundary polygon specified above. Furthermore, denote by $\mathsf{mwt}(i, j, T)$ the weight of a minimum weight triangulation of the pointgon $\Pi(i, j, T)$. Then, Eq. (1) can be rewritten in the following way if we take p_0 for the role of p:

$$\begin{aligned}&\mathsf{mwt}(\Pi) = \\ &\min\Big\{ \min_{1\le i<n-k,\ T\subseteq S} \{\mathsf{mwt}(0, i, T) + \mathsf{mwt}(i, 0, T) - \mathsf{length}(\gamma(T))\}, \\ &\mathsf{mwt}(1, n-k-1, \emptyset, \mathsf{null}) + \mathsf{mwt}(n-k-1, 1, \emptyset, \mathsf{null}) - \mathsf{length}(\overline{p_1 p_{n-k-1}})\Big\} . \quad (2)\end{aligned}$$

The number of values considered in the right hand side of Eq. (2) is $O((n-k)2^k) = O(2^k n)$. Hence, for the computation of $\mathsf{mwt}(\Pi)$ it is sufficient to know $\mathsf{mwt}(i, j, T)$ for every triple (i, j, T) of two indices $i, j \in \{0, \dots, n-k-1\}$ and a subset $T \subseteq S$. Since the number of such triples is $O(2^k n^2)$, the efficient computation of each value results in fixed parameter tractability of the minimum weight triangulation problem.

Nevertheless, to compute these values, we have to generalize the class of pointgons under consideration. That is because pointgons we encounter in the recursion might not be of the form $\Pi(i, j, T)$. Therefore we introduce two additional types of pointgons.

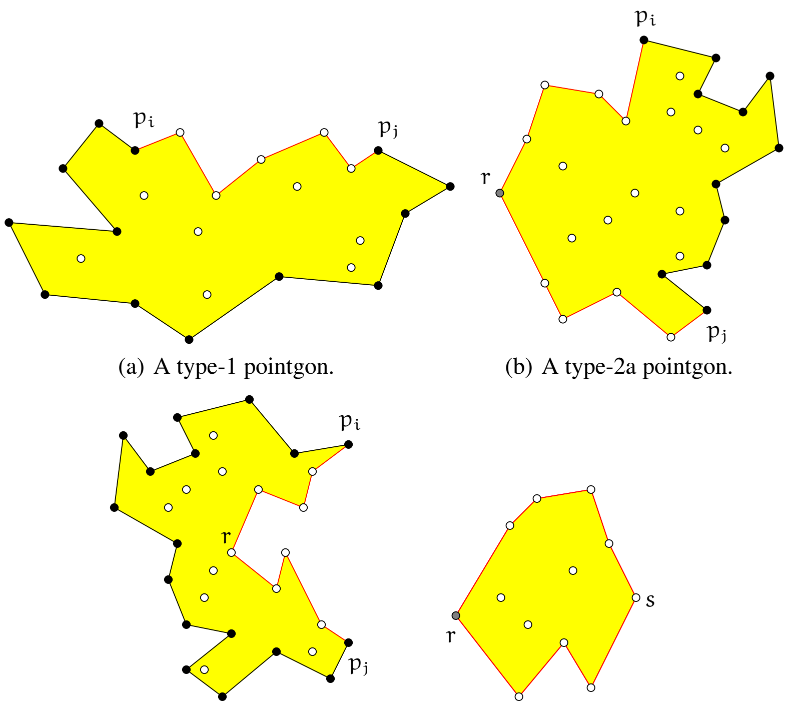
The pointgon $\Pi(i, j, T)$ is bounded by two kinds of polygonal chains: a chain $\bigcup_{\ell=i}^{j-1} \overline{p_\ell p_{\ell+1}}$ from the boundary of the original pointgon and a non-selfintersecting x-monotone inner path $\gamma(T)$. Recall that T can be empty. We call such a pointgon a *type-1 pointgon* in the following. See Fig. 3(a) for illustration.

Another class of pointgons is defined for $i, j \in \{0, n-k-1\}$, two disjoint subsets $T_1, T_2 \subseteq S$, and a vertex $r \in \mathsf{Vert}(\Pi)$. Then, $\Pi(i, j, T_1, T_2, r)$ is a pointgon bounded by the x-monotone path connecting i and r through T_1, the x-monotone path connecting j and r through T_2, and $\bigcup_{\ell=i}^{j-1} \overline{p_\ell p_{\ell+1}}$. (Again we only consider those tuples which are well defined, that is, where the paths described above are indeed x-monotone and do not cross each other.) We call such a pointgon a *type-2 pointgon* of Π, and divide them into two subclasses according to whether r is a convex (type-2a) or reflex (type-2b) vertex of the pointgon. Fig. 3(b) & 3(c) illustrate the definition.

The last kind of pointgons uses at most one vertex of P. For a vertex $r \in \mathsf{Vert}(\Pi)$ and two subsets $T_1, T_2 \subseteq S$ with $T_1 \cap T_2 = \{s\}$, we define the pointgon $\Pi(T_1, T_2, r)$ as one which is bounded by two x-monotone paths connecting r and s through T_1 and through T_2, respectively. We call such a pointgon a *type-3 pointgon* of Π. See Fig. 3(d) for an example.

Let us count the number of these pointgons. The number of type-1 pointgons is $O(2^k n^2)$; the number of type-2 pointgons is $O(3^k n^3)$; the number of type-3 pointgons is $O(3^k n)$. Therefore, the total number of these pointgons is $O(3^k n^3)$. Our goal in the following is to compute the weights of minimum weight triangulations of these pointgons efficiently. Denote by $\mathsf{mwt}(i, j, T)$ the weight of a minimum weight triangulation of a type-1 pointgon $\Pi(i, j, T)$. Similarly, we define $\mathsf{mwt}(i, j, T_1, T_2, r)$ and $\mathsf{mwt}(T_1, T_2, r)$ for type-2 and type-3 pointgons, respectively.

Before describing the algorithm, let us discuss why we only encounter these three types of pointgons in the recursion. For this, we have to be careful which vertex to choose as p in the recursion step. Recall that in any step of Recursion (1) there are two cases: either p is cut off by joining its neighbors by an edge, or the pointgon is subdi-



(a) A type-1 pointgon. (b) A type-2a pointgon.

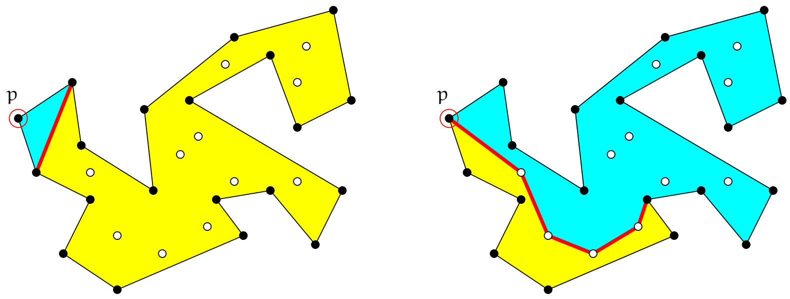
(c) A type-2b pointgon. (d) A type-3 pointgon.

Fig. 3: The three types of subpointgons of Π. The vertex r can be either a solid or an empty circle.

vided by an x-monotone inner path starting from p. Also recall that in Observation 2 we required p to be the leftmost point of the pointgon. If we apply the same argument as in Observation 2 to an arbitrary vertex of the pointgon, in the second case there appears an inner path starting from p that is *almost* x*-monotone*, i.e., x-monotone except for the first edge incident to p.

Initially we have a given pointgon $\Pi = (S, P)$ and choose the leftmost vertex as p. If p is cut off (Fig. 4(a)) the result is a type-1 pointgon where $T = \emptyset$. Any x-monotone inner path starting from p divides the pointgon into two type-1 pointgons (Fig. 4(b)).

When we apply Recursion (1) to a type-1 pointgon $\Pi(i, j, T)$, we choose as p the leftmost vertex of the inner path $\gamma(T)$ (which might consist just of a single edge joining p_i and p_j). If p is cut off, the result is either again a type-1 pointgon (Fig. 5(a)) or a type-2a pointgon (Fig. 5(b)). Otherwise, consider the vertex q on $\gamma(T)$ next to p. In every triangulation, the edge $\overline{pq}$ must belong to some triangle. To make such a triangle we need another vertex, say z. Let us choose z to be such that $\overline{pz}$ is the first edge of an almost x-monotone inner path γ' starting from p. If $z \in \mathsf{Vert}(P)$, then we get a type-1 pointgon, the triangle pqz and a type-2a pointgon when z is right of p (Fig. 5(c)), or a type-1 pointgon, the triangle pqz and a type-1 pointgon when z is left of p (Fig. 5(d)). If $z \in S$, then we have four subcases. When z is right of p and γ' ends at a vertex of $\gamma(T)$, we get a type-1 pointgon, the triangle pqz and a type-3 pointgon (Fig. 6(a)).



(a) The vertex p is cut off. (b) An inner path from p.

Fig. 4: Subdivisions obtained from Π. From now on in the pictures, the vertex p is indicated by a larger circle.

When z is right of p and γ' ends at a vertex of P, we get a type-1 pointgon, the triangle pqz and a type-2a pointgon (Fig. 6(b)). When z is left of p and γ' ends at a vertex of $\gamma(T)$, we get a type-2b pointgon, the triangle pqz and a type-3 pointgon (Fig. 6(c)). When z is left of p and γ' ends at a vertex of P, we get a type-2b pointgon, the triangle pqz and a type-2a pointgon (Fig. 6(d)).

When we apply Recursion (1) to a type-2a pointgon $\Pi(i, j, T_1, T_2, r)$, we choose r as p. If p is cut off the result is either again a type-2a pointgon or a type-1 pointgon (Fig. 7(a)). Otherwise, consider an x-monotone inner path starting from p. If the path ends at a vertex of P, we get two type-2a pointgons (Fig. 7(b)). If, on the other hand, the inner path ends at a vertex in S, then it subdivides the pointgon into a type-2a and a type-3 pointgons (Fig. 7(c)).

When we apply Recursion (1) to a type-2b pointgon, we choose as p the leftmost vertex of the inner path. Since p is a reflex vertex, p cannot be cut off. So, every x-monotone inner path starting from p subdivides the pointgon into two type-1 pointgons (Fig. 7(d)).

When we apply Recursion (1) to a type-3 pointgon $\Pi(T_1, T_2, r)$, we choose r as p. Then, no matter how we divide the pointgon by the operations in the recursion, the result is again a type-3 pointgon (Fig. 8).

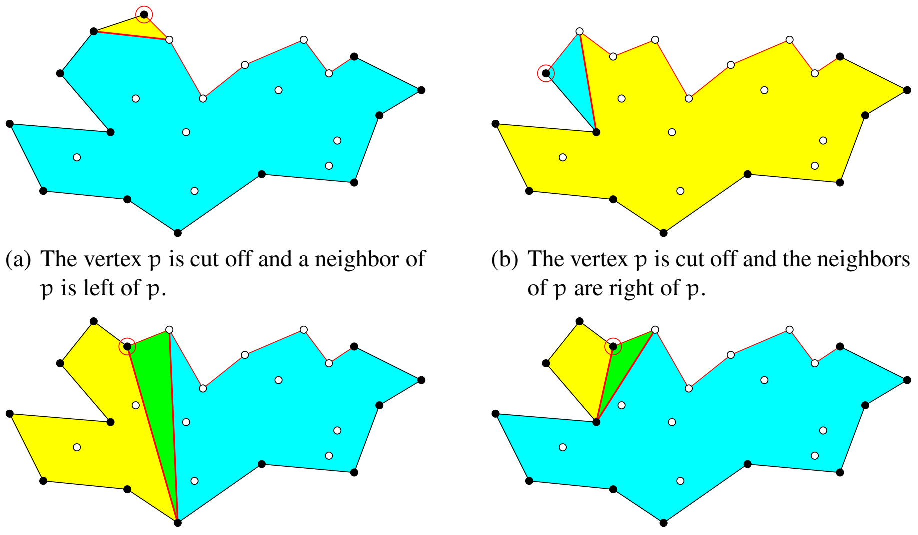
So much for preparation, and now we are ready to give the outline of our algorithm.

Step 1: Enumerate all possible type-1 pointgons $\Pi(i, j, T)$, type-2 pointgons $\Pi(i, j, T_1, T_2, r)$, and type-3 pointgons $\Pi(T_1, T_2, r)$.

Step 2: Compute the values $\mathsf{mwt}(i, j, T)$, $\mathsf{mwt}(i, j, T_1, T_2, r)$, and $\mathsf{mwt}(T_1, T_2, r)$ for some of them, which are sufficient for Step 3, by dynamic programming.

Step 3: Compute $\mathsf{mwt}(\Pi)$ according to Eq. (2).

We already argued that Step 3 can be done in $O(2^k n)$ time. In the next section we will show that Steps 1 & 2 can be done in $O(6^k n^5 \log n)$ time, which dominates the overall running time.



(a) The vertex p is cut off and a neighbor of p is left of p.

(b) The vertex p is cut off and the neighbors of p are right of p.

(c) The vertex z is a solid circle and z is right of p.

(d) The vertex z is a solid circle and z is left of p.

Fig. 5: Subdivisions obtained from a type-1 pointgon.

3.3 Dynamic Programming

Now, we are going to compute the values of $\mathsf{mwt}(i, j, T)$, $\mathsf{mwt}(i, j, T_1, T_2, r)$, and $\mathsf{mwt}(T_1, T_2, r)$ for all possible choices of i, j, T_1, T_2, r.

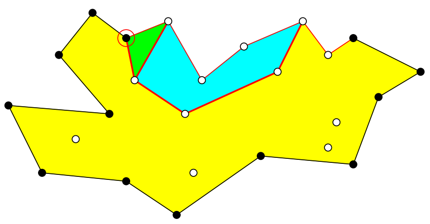
First we enumerate all possibilities of i, j, T_1, T_2, r. Each of them can be enumerated in $O(1)$ time, and each of them can be identified as a well-defined pointgon or not (i.e., the inner paths do not intersect each other nor the boundary) in $O(n \log n)$ time. (Apply the standard line segment intersection algorithm [15].) Therefore, it takes $O(3^k n^3 \cdot 1 \cdot n \log n) = O(3^k n^4 \log n)$ time. This completes Step 1 of our algorithm.

Then, we perform the dynamic programming. Determine the vertex p and consider all possible subdivisions with respect to p as described in the previous section. Each subdivision replaces Π by two smaller pointgons. Then, as we saw, these two pointgons can be found among those enumerated in Step 1.

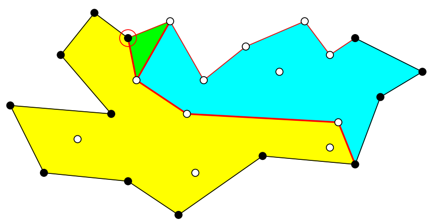
We can associate a parent-child relation between two pointgons Π_1, Π_2 in our enumeration: Π_1 is a parent of Π_2 if Π_2 is obtained as a smaller pointgon when we partition Π_1 by a path starting from p (which is fixed as in the previous section) or through the edge cutting off p. It can also be thought as defining a directed graph on the enumerated pointgons: namely, draw a directed edge from Π_1 to Π_2 if the same condition as above is satisfied.

Observe that if Π_1 is a parent of Π_2, then the number of inner points in Π_2 is less than that in Π_1 or $|T_1| + |T_2|$ is smaller in Π_2 than in Π_1. Therefore, the parent-child relation is well-defined (i.e., there is no directed cycle in the directed-graph formulation).

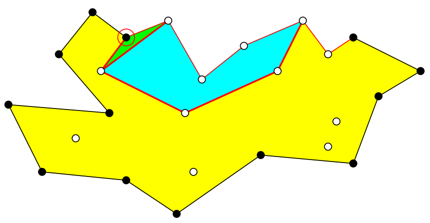
Now, to do the bottom-up computation, we first look at the lowest descendants (or the sinks in the directed-graph formulation). They are triangles. So, the weights can be



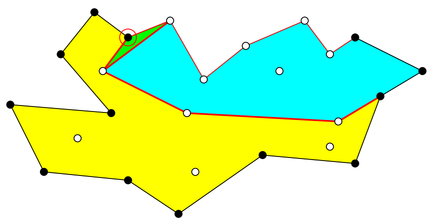
(a) The vertex z is right of p and the path γ' ends at an empty circle.



(b) The vertex z is right of p and the path γ' ends at a solid circle.



(c) The vertex z is left of p and the path γ' ends at an empty circle.



(d) The vertex z is left of p and the path γ' ends at a solid circle.

Fig. 6: Subdivisions obtained from a type-1 pointgon (continued). The vertex z is an empty circle.

easily computed in constant time. Then, we proceed to their parents. For each parent, we look up the values of its children. In this way, we go up to the highest ancestor, which is Π. Thus, we can compute $\mathsf{mwt}(\Pi)$.

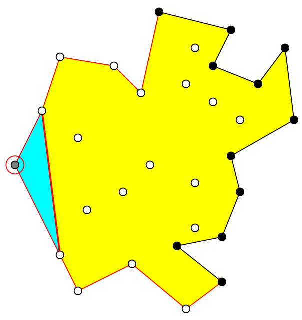
What is the time complexity of the computation? First, let us estimate the time for the construction of the parent-child relation. The number of enumerated pointgons is $O(3^k n^3)$. For each of them, the number of possible choices of non-selfintersecting x-monotone paths is $O(2^k n)$. For each of the paths, we can decide whether it really defines a non-selfintersecting path in $O(n \log n)$ time. Therefore, the overall running time for the construction is $O(3^k n^3 \cdot 2^k n \cdot n \log n) = O(6^k n^5 \log n)$.

In the bottom-up computation, for each pointgon we look up at most $O(2^k n)$ entries and compute the value according to Eq. (1). Therefore, this can be done in $O(3^k n^3 \cdot 2^k n) = O(6^k n^4)$.

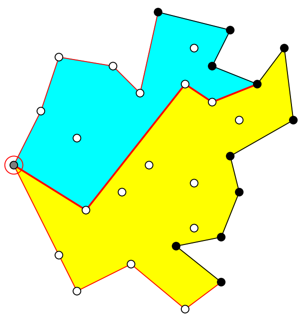
Hence, the overall running time of the algorithm is $O(3^k n^4 \log n + 6^k n^5 \log n + 6^k n^4) = O(6^k n^5 \log n)$. This completes the proof of Theorem 1.

4 Conclusion

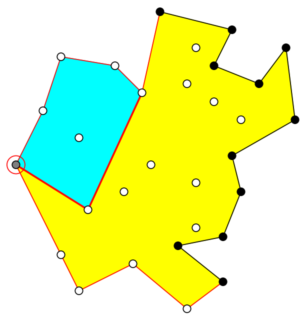
In this paper, we studied the minimum weight triangulation problem from the viewpoint of fixed-parameter tractability. We established an algorithm to solve this problem for a simple polygon with some inner points which runs in $O(6^k n^6 \log n)$ time when n is the total number of input points and k is the number of inner points. Therefore, the problem is fixed-parameter tractable with respect to the number of inner points. We believe the number of inner points in geometric optimization problems plays a role similar to the treewidth in graph optimization problems.



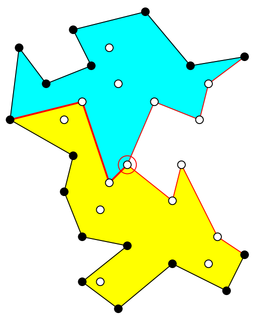
(a) Type-2a. p is cut off.



(b) Type-2a. The inner path ends at a solid circle.



(c) Type-2a. The inner path ends at an empty circle.



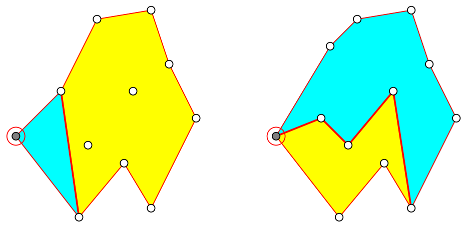
(d) Type-2b.

Fig. 7: Subdivisions obtained from a type-2 pointgon.

Since our algorithm is based on a simple idea, it can be extended in several ways. For example, we can also compute a maximum weight triangulation in the same time complexity. (It seems quite recent that attention has been paid to a maximum weight triangulation [10, 16].) To do that, we just replace "min" in Eqs. (1) and (2) by "max." Another direction of extension is to incorporate some heuristics. For example, there are some known pairs of vertices which appear as edges in all minimum weight triangulations, e.g. the β-skeleton for some β and the LMT-skeleton; see [4, 5, 17] and the references therein. Because of the flexibility of our algorithm, we can insert these pairs at the beginning of the execution as edges, and proceed in the same way except that we can use the information from the prescribed edges.

The framework proposed in this paper looks promising when we deal with the complexity of geometric problems concerning a finite point set on the plane. Study of another problem within the same framework is interesting. Recently, the traveling salesman problem was considered and it was shown that the problem can be solved in polynomial time when $k = O(\log n)$ [6].

The obvious open problem is to improve the time complexity of our algorithm. For example, is it possible to provide a polynomial-time algorithm for the minimum weight triangulation problem when $k = O(\log^2 n)$?



(a) The vertex p is cut off.

(b) An inner path from p.

Fig. 8: Subdivisions obtained from a type-3 pointgon.

References

1. O. Aichholzer: The path of a triangulation. Proc. 15th SoCG (1999) 14–23.
2. O. Aichholzer, G. Rote, B. Speckmann and I. Streinu: The zigzag path of a pseudo-triangulation. Proc. 8th WADS, Lect. Notes Comput. Sci. **2748** (2003) 377–388.
3. E. Anagnostou and D. Corneil: Polynomial-time instances of the minimum weight triangulation problem. Comput. Geom. **3** (1993) 247–259.
4. P. Bose, L. Devroye and W. Evans: Diamonds are not a minimum weight triangulation's best friend. Internat. J. Comput. Geom. Appl. **12** (2002) 445–453.
5. S.-W. Cheng and Y.-F. Xu: On β-skeleton as a subgraph of the minimum weight triangulation. Theor. Comput. Sci. **262** (2001) 459–471.
6. V.G. Deĭneko, M. Hoffmann, Y. Okamoto and G.J. Woeginger: The traveling salesman problem with few inner points. Proc. 10th COCOON, Lect. Notes Comput. Sci. **3106** (2004). To appear.
7. R.G. Downey and M.R. Fellows: Parameterized Complexity. Springer, Berlin, 1999.
8. M. Garey and D. Johnson: Computers and Intractability. W.H. Freeman & Company, 1979.
9. P.D. Gilbert: New results in planar triangulations. Master Thesis, University of Illinois, Urbana, 1979.
10. S. Hu: A constant-factor approximation for maximum weight triangulation. Proc. 15th CCCG (2003) 150–154.
11. Y. Kyoda, K. Imai, F. Takeuchi, and A. Tajima: A branch-and-cut approach for minimum weight triangulation. Proc. 8th ISAAC, Lect. Notes Comput. Sci. **1350** (1997) 384–393.
12. G. Klincsek: Minimal triangulations of polygonal domains. Annals Discrete Math. **9** (1980) 121–123.
13. C. Levcopoulos and D. Krznaric: Quasi-greedy triangulations approximating the minimum weight triangulation. J. Algor. **27** (1998) 303–338.
14. R. Niedermeier: Invitation to fixed-parameter algorithms. Habilitation Thesis, Universität Tübingen, 2002.
15. M.I. Shamos and D. Hoey: Geometric intersection problems. Proc. 17th FOCS (1976) 208–215.
16. C.A. Wang, F.Y. Chin, and B.T. Yang: Maximum weight triangulation and graph drawing. Inform. Proccess. Lett. **70** (1999) 17–22.
17. C.A. Wang and B. Yang: A lower bound for β-skeleton belonging to minimum weight triangulations. Comput. Geom. **19** (2001) 35–46.