Quadtree, Ray Shooting and Approximate Minimum Weight Steiner Triangulation*

Siu-Wing Cheng† Kam-Hing Lee‡

January 31, 2001

Abstract

We present a quadtree-based decomposition of the interior of a polygon with holes. The complete decomposition yields a constant factor approximation of the minimum weight Steiner triangulation (MWST) of the polygon. We show that this approximate MWST supports ray shooting queries in the query-sensitive sense as defined by Mitchell, Mount and Suri. A proper truncation of our quadtree-based decomposition yields another constant factor approximation of the MWST. For a polygon with n vertices, the complexity of this approximate MWST is $O(n \log n)$ and it can be constructed in $O(n \log n)$ time. The running time is optimal in the algebraic decision tree model.

*The work described in this paper has been supported by the Research Grants Council of Hong Kong, China (Project no. HKUST 650/95E). The work was conducted when the second author was studying at the Department of Computer Science, HKUST. A preliminary version appears in Proceedings of International Symposium on Algorithms and Computation, 1998, 367–376.

†Department of Computer Science, HKUST, Clear Water Bay, Hong Kong

‡eCyberPay.com Limited, 25/F, CEF Lend Lease Plaza, 663 King's Road, Hong Kong.

1 Introduction

The Steiner triangulation of a planar polygonal domain is a partition of the domain into triangles. Steiner points are allowed. Each input line segment must be the union of some edge(s) in the triangulation. The *weight* of the triangulation is the total edge length. No polynomial-time algorithm is known that computes the *minimum weight Steiner triangulation* (MWST for short) of a point set, a set of line segments, or a polygon. We are interested in approximate MWSTs of polygons and their application in ray shooting.

Approximate MWST Eppstein [7] showed that a constant factor approximation of the MWST of n points or a convex polygon with n vertices can be computed in $O(n \log n)$ time. The problem of designing an approximation algorithm for general polygons was posed by Bern and Eppstein [5]. We present a solution in this paper.

Average-case ray shooting The MWST problem is recently related to the *ray shooting* problem by Aronov and Fortune [1]. The ray shooting problem is to report the first obstacle hit by a query ray. In this paper, we assume that the query light source falls within a polygon P with holes and the boundary edges act as obstacles. Given a Steiner triangulation of P, a query is answered by following the ray across triangulation edges until the first obstacle is hit. Aronov and Fortune [1] showed that for a particular distribution of light source and ray direction, the average number of triangulation edges crossed is bounded by the ratio of the triangulation weight to the perimeter of P. Thus, a Steiner triangulation of low weight is desirable. In the same paper, Aronov and Fortune presented an approximation algorithm for a set of line segments enclosed within their convex hull. This approximation algorithm is based on the quadtree triangulation of the vertices. As observed by Eppstein [7], triangulating a polygon by refining the quadtree triangulation of the vertices may lead to an $\Omega(\log n)$ factor between the resulting triangulation and the MWST.

Query-sensitive ray shooting Mitchell, Mount and Suri [9] proposed the *C-complexity* to measure the geometric complexities of a polygon P and a query ray r. A *C-ball* D is a connected component of the intersection of P and a disk K. The center and radius of D are taken to be the center and radius of K. If we expand K by a factor of $1 + \epsilon$ without moving its center, then the expanded disk contains a unique C-ball that contains D. We denote this C-ball by $(1+\epsilon)D$. D is *simple* if it is incident on at most two edges of P. D is *ϵ-strongly simple* if both D and $(1+\epsilon)D$ are simple. The C-complexity of P, denoted by $cscc(P)$, is the minimum number of ϵ-strongly simple C-balls that cover P. $cscc(P)$ is expected to be $O(n)$ in practice, where n is the number of vertices of P. (See de Berg et al [3] for discussion of realistic input models.) Given a query ray r, let $cscc(r)$ denote the minimum number of ϵ-strongly simple C-balls that cover r up to the first intersection with an obstacle. Mitchell et al [9] argued that $cscc(r)$ is a reasonable reflection of the intrinsic complexity of solving a ray shooting query. They presented an algorithm to subdivide P into $O(n)$ regions of constant complexities. They also prove that the number of regions visited in following r is $O(cscc(r))$.

Summary of results

1. We develop a quadtree-based decomposition of the interior of a polygon with holes. This is inspired by the quadtree decomposition developed by Arya, Cheng and Mount [2] for multiple-tool milling.

2. Given a polygon P, our complete quadtree-based decomposition yields a constant factor approximation of the MWST of P. Its complexity is $O(cscc(P))$. Given a query ray r, it takes $O(\log cscc(P) + cscc(r))$ time to locate the source of r and traverse this approximate MWST to report the first obstacle hit.

3. We show that a proper truncation of the decomposition yields another constant factor approximation of the MWST of P. Its complexity is $O(n \log n)$ and it can be constructed in $O(n \log n)$ time. Our algorithm works under the algebraic decision tree model and the processing time is optimal in this model.

Outline In Section 2, we define our quadtree-based decomposition and prove its properties. In Section 3, we discuss query-sensitive ray shooting. In Section 4, we analyze the weight of our quadtree-based decomposition and the weight of triangulations derived from it. In Section 5, we present the $O(n \log n)$-time approximation algorithm.

2 Quadtree-based decomposition

Let P denote a polygon possibly with holes. We allow a hole to degenerate to a point. We assume that each vertex of P is incident on two edges or no edge at all. In the latter case, the vertex is a degenerate hole.

Define the *bounding box* of P to be the smallest and leftmost axes-parallel square containing P. We scale space such that the width of the bounding box is 1. The width of an axes-parallel square B is denoted by $width(B)$. For any $c \geq 1$, the *c-expansion* of B is the axes-parallel square with the same center as B and width $c \cdot width(B)$. The c-expansion of B is denoted by cB.

Quadtree box The bounding box of P is the initial quadtree box. Other quadtree boxes are obtained by recursive splitting. A quadtree box B is split by passing a horizontal line and a vertical line through the center of B. Four smaller quadtree boxes are obtained and B is their *parent*.

Decomposition A *quadtree-based decomposition* is a recursive splitting of quadtree boxes and *subpolygons*. Subpolygons are connected components of the intersection of P and quadtree boxes. P is the only subpolygon of the initial quadtree box (the bounding box of P).

Let δ be a fixed non-negative constant. Let B be a quadtree box obtained during the recursive splitting. Take a subpolygon X of B. Define $Z(B, X)$ to be the connected component of $P \cap 3B$ that contains X. If $width(B) > \delta$ and $Z(B, X)$ has more than one vertex of P, then X is *crowded* in B. If B contains a crowded subpolygon, then B is split into four quadtree boxes B_i, $1 \leq i \leq 4$. The subpolygons of B_i are the connected components in the intersection of B_i and crowded subpolygons of B. The non-crowded subpolygons of B stay and become *chambers* of the final decomposition. We say that these chambers are *created* at B and B is the *home box* of its chambers. Figure 1 illustrates one step of this hierarchical decomposition. A *quadtree level* is a layer of quadtree boxes of the same width and their subpolygons in the hierarchical decomposition.

After a chamber is created at B, its edges may be further subdivided subsequently when a neighboring quadtree box is split. Edges of a chamber that lie on the boundary of P are called *solid edges*. The others are called *non-solid edges*. Two chambers are *adjacent* if they share a non-solid edge.

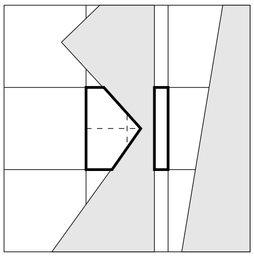

Figure 1: The shaded regions are parts of holes. There are two subpolygons shown with bold boundaries. The subpolygon on the right is not crowded and so it becomes a chamber. The subpolygon on the left is crowded and it is split further by the two dashed line segments.

We use $\mathcal{S}_P(\delta)$ to denote the final decomposition of P into chambers. If $\delta = 0$, then the quadtree-based decomposition of P is *complete* and each chamber has at most one vertex of P.

Notations for chambers Denote by $\mathit{parent\_box}(C)$ the parent of the home box of a chamber C. Denote by $\mathit{parent\_poly}(C)$ the subpolygon of $\mathit{parent\_box}(C)$ that contains C. The *size* of a chamber C, $\mathit{size}(C)$, is defined to be the width of its home box. C is *normal* if $\mathit{size}(C) > \delta$; otherwise C is *small*. All small chambers have the same size.

Properties We prove two properties of $\mathcal{S}_P(\delta)$. First, the sizes of two adjacent chambers differ by at most a constant factor. This resembles the "balance" and "smoothness" properties of other quadtree-based decompositions [6, 9]. Second, a chamber has constant complexity if it has at most one vertex of P.

Lemma 1 *The sizes of two adjacent chambers differ by at most a factor of 2.*

Proof. Assume to the contrary that C_1 and C_2 are two adjacent chambers but $\mathit{size}(C_2) < \mathit{size}(C_1)/2$. Let B_1 be the home box of C_1. Let $B = \mathit{parent\_box}(C_2)$ and let $X = \mathit{parent\_poly}(C_2)$. Since X is crowded in B and $\mathit{width}(B) = 2\mathit{size}(C_2) > \delta$, $Z(B, X)$ has more than one vertex of P. Since $\mathit{width}(B) = 2\mathit{size}(C_2) \leq \mathit{size}(C_1)/2$ and C_2 is adjacent to C_1, $Z(B, X) \subseteq Z(B_1, C_1)$. It follows that $Z(B_1, C_1)$ has more than one vertex of P. Moreover, $\mathit{size}(C_1) > 2\mathit{size}(C_2) > \delta$, so C_1 is crowded in B_1 which is a contradiction. $\square$

Lemma 2 *A chamber has constant complexity if it has at most one vertex of P.*

Proof. Let C be such a chamber and let B be its home box. Consider the time when C is created at B. Without loss of generality, assume that the width of any neighboring quadtree box is at least $width(B)$ at this time. Let E be the set of solid edges of C that do not lie on the boundary of B. At this time, the complexity of C is $O(|E|)$. If C has a vertex of P, then there are two solid edges incident on this vertex. Each other solid edge in E crosses the interior of B completely and it cuts off a corner of B. This implies that $|E| \leq 6$. Let E' be the set of edges of C that lie on the boundary of B. If C is a normal chamber, then a non-solid edge e in E' may be subdivided subsequently when a neighboring quadtree box is split. The complexity of C may then be increased. By Lemma 1, e can be subdivided at most once. It follows that $|E'|$ at most doubles and the complexity of C is still $O(1)$. $\square$

3 Query-sensitive ray shooting

We show that triangulating $\mathcal{S}_P(0)$ yields a constant factor approximation of the MWST with $O(cscc(P))$ complexity. It supports ray shooting query in $O(\log cscc(P) + cscc(r))$ time. We prove the complexity and query time bounds in this section. The weight analysis is more involved and is postponed to Section 4. Lemma 7 in Section 4 implies that the weight of any triangulation of $\mathcal{S}_P(0)$ is $O(wt(T))$, where $wt(T)$ is the weight of any Steiner triangulation T of P.

We first show that no ϵ-strongly simple C-ball can intersect a much smaller chamber in $\mathcal{S}_P(0)$.

Lemma 3 *Let D be an ϵ-strongly simple C-ball. If D intersects a chamber C in $\mathcal{S}_P(0)$, then $size(C) \geq \epsilon \cdot radius(D)/4\sqrt{2}$.*

Proof. Assume to the contrary that $size(C) < \epsilon \cdot radius(D)/4\sqrt{2}$. Let $B = parent\_box(C)$ and let $X = parent\_poly(C)$. Since B intersects D and $width(B) < \epsilon \cdot radius(D)/2\sqrt{2}$, any point in $3B$ is within a distance of

$2\sqrt{2}width(B) < \epsilon \cdot radius(D)$ from D. It follows that $Z(B, X) \subseteq (1+\epsilon)D$. $(1+\epsilon)D$ is simple, so $Z(B, X)$ has at most one vertex of P. X is thus not crowded in B, a contradiction. $\square$

We prove that an ϵ-strongly simple C-ball intersects $O(1)$ chambers in $\mathcal{S}_P(0)$. We basically use a packing argument, but it is complicated by the non-disjointness of the home boxes of chambers.

Lemma 4 *An ϵ-strongly simple C-ball intersects a constant number of chambers in $\mathcal{S}_P(0)$.*

Proof. Let D be an ϵ-strongly simple C-ball. Consider the chambers with size s intersected by D. Denote the set of home boxes of these chambers by H_s. Pick a box B in H_s. Since $width(B) \geq \epsilon \cdot radius(D)/4\sqrt{2}$ by Lemma 3, B contains an axes-parallel square B' such that $width(B') = \epsilon \cdot radius(D)/8\sqrt{2}$ and B' intersects D. Let K be the disk with the same center as D and radius $(1+\epsilon/8)radius(D)$. Clearly, $B' \subseteq K$. The disjointness of boxes in H_s implies that $area(K) = \pi(1+\epsilon/8)^2(radius(D))^2 \geq |H_s| \cdot \epsilon^2 \cdot (radius(D))^2/128$, so $|H_s| \leq 2\pi(8+\epsilon)^2/\epsilon^2$. We bound $\sum_s |H_s|$ by analyzing the possible values of s. There are two cases.

`Case 1`: The size of the largest chamber intersecting D is at least $4radius(D)$. Let C be the largest chamber intersecting D and let B be its home box. We claim that for any other chamber C' intersecting D, $size(C') > size(C)/16$.

Suppose not. Let $B'' = parent\_box(C')$ and let $X = parent\_poly(C')$. Let S be the axes-parallel square with the same center as D and width $size(C)$. Since $width(B'') \leq size(C)/8$ and $radius(D) \leq size(C)/4$, $3B''$ lies inside S. D also lies inside S. Any point in S is within a L_∞-distance of $size(C)/2 + radius(D) \leq 3size(C)/4$ from B, so $S \subseteq 3B$. This implies that both $Z(B'', X)$ and D lie inside $3B$. Moreover, D intersects both X and C. We conclude that every point in $Z(B'', X)$ is connected to C by a path (via D) that lies inside $3B$. Thus, $Z(B'', X) \subseteq Z(B, C)$. Since C is a chamber, $Z(B, C)$ has at most one vertex of P and so does $Z(B'', X)$. This is a contradiction since X is crowded in B''.

By our claim, there are at most four possible values of s, so $\sum_s |H_s| \leq 8\pi(8+\epsilon)^2/\epsilon^2$.

`Case 2`: The size of any chamber intersecting D is less than $4radius(D)$. Lemma 3 implies that there are at most $\lceil \log_2 16\sqrt{2}/\epsilon \rceil$ possible values of s, so $\sum_s |H_s| \leq 2\pi((8+\epsilon)^2/\epsilon^2)\lceil \log_2 16\sqrt{2}/\epsilon \rceil$.

For any box B in H_s, D intersects at most two chambers in B. Otherwise, D is incident on three chamber edges that lie on three distinct edges of P. This is a contradiction since an ϵ-strongly simple C-ball is incident on at most two edges of P. Hence, D intersects at most $2\sum_s |H_s|$ chambers. □

We are ready to prove the main result of this section.

Theorem 1 *Let P be a polygon with n vertices. There is a Steiner triangulation of P with $O(cscc(P))$ size and $O(wt(T))$ weight, where $wt(T)$ is the weight of any Steiner triangulation T of P. Moreover, ray shooting query can be answered in $O(\log cscc(P) + cscc(r))$ time, where r is the query ray.*

Proof. We claim that any triangulation of $\mathcal{S}_P(0)$ satisfies the theorem. Its weight is $O(wt(T))$ by Lemma 7 in Section 4. By Lemma 4, the number of chambers in $\mathcal{S}_P(0)$ is bounded by the minimum number of ϵ-strongly simple C-balls that cover P. Since a chamber in $\mathcal{S}_P(0)$ has at most one vertex of P, the chamber complexity is $O(1)$ by Lemma 2. It follows that the complexities of $\mathcal{S}_P(0)$ and its triangulation are $O(cscc(P))$. Given any query ray r, we perform a planar point location to find the light source in $O(\log cscc(P))$ time. We then follow r to report the first obstacle hit. By Lemma 4, the number of chambers visited is $O(cscc(r))$. Each chamber is subdivided into $O(1)$ triangles. The query time is thus $O(\log cscc(P) + cscc(r))$. □

4 Weight analysis

The weight of our quadtree-based decomposition is the total edge length. In this section, we analyze the weight of $\mathcal{S}_P(\delta)$ and the weight of triangulations derived from it. For the sake of analysis, we conceptually refine $\mathcal{S}_P(\delta)$. Let T be an arbitrary Steiner triangulation of P. For each Steiner point of T, if it lies in the interior of P, add it as a hole; otherwise add it as a new vertex. Compute a quadtree-based decomposition of the resulting polygon as described in Section 2, except that the definition of crowdedness is modified. Let λ be an arbitrarily small positive constant. So λ is less than the size of the smallest chamber in $\mathcal{S}_P(\delta)$. Let B be a quadtree box and let X be a subpolygon of B. X is crowded in B if $width(B) > \lambda$ and $Z(B, X)$ has *at least one vertex* of T. Denote the final decomposition by $\mathcal{D}_{P,T}(\lambda)$. A chamber C of $\mathcal{D}_{P,T}(\lambda)$ is normal if $size(C) > \lambda$; otherwise C is small.

Our quadtree-based decomposition in Section 2 cannot be defined using the above modified definition of crowdedness. Otherwise, our result on query-sensitive ray shooting will not hold as Lemma 3 will break.

Observation 1 *Any subpolygon that is found crowded during the construction of $\mathcal{S}_P(\delta)$ is also crowded under the modified definition of crowdedness. It follows that $\mathcal{D}_{P,T}(\lambda)$ is a refinement of $\mathcal{S}_P(\delta)$.*

Observation 2 *Lemmas 1 and 2 hold for $\mathcal{D}_{P,T}(\lambda)$ after adapting the proofs to the modified definition of crowdedness.*

If a chamber has a vertex of T, then it must be small. Moreover, when λ is sufficiently small, a small chamber has at most one vertex of T. Thus, Observation 2 and Lemma 2 imply the following.

Observation 3 *Each chamber in $\mathcal{D}_{P,T}(\lambda)$ has constant complexity.*

We need a technical result.

Lemma 5 *Let C_1 be a non-square chamber of $\mathcal{D}_{P,T}(\lambda)$ that does not have any vertex of T. If the length of each solid edge of C_1 is less than $size(C_1)/4$, then C_1 is adjacent to a non-square chamber C_2 and one of the following holds:*

(i) C_2 does not have any vertex of T and the length of some solid edge of C_2 is greater than $size(C_2)/2 \geq size(C_1)/4$.

(ii) C_2 has a vertex of T and both C_1 and C_2 are small.

Proof. Let B_1 be the home box of C_1. By our assumptions about C_1, C_1 is incident on an edge h of P such that h crosses the interior of B_1 completely and $|h \cap B_1| < size(C_1)/4$. So h cuts off exactly one corner of B_1. h is incident on a non-square chamber C_2 adjacent to C_1. Let B_2 be the home box of C_2.

If $size(C_2) = size(C_1)/2$, then $size(C_1) > \lambda$ and $C_2 \subseteq Z(B_1, C_1)$. Since $size(C_1) > \lambda$ and C_1 is not crowded in B_1, $Z(B_1, C_1)$ is free of vertices of

T and so is C_2. This implies that h crosses B_2 completely, so $|h \cap B_2| > size(C_2) - size(C_1)/4 = size(C_2)/2 = size(C_1)/4$ and (i) holds.

By Observation 2 and Lemma 1, the remaining possibility is that $size(C_2) \geq size(C_1)$. If C_2 does not have any vertex of T, then h crosses B_2 completely, so $|h \cap B_2| > size(C_2) - size(C_1)/4 > size(C_2)/2 > size(C_1)/4$ and (i) holds. If C_2 has a vertex of T, then C_2 is small. This implies that $size(C_1) = size(C_2) = \lambda$, so C_1 is also small and (ii) holds. □

We are ready to bound the weight of $\mathcal{S}_P(\delta)$.

Lemma 6 *The weight of $\mathcal{S}_P(\delta)$ is $O(wt(T))$, where $wt(T)$ is the weight of any Steiner triangulation T of P.*

Proof. Let T be a Steiner triangulation of P. By Observation 1, it suffices to show that the weight of $\mathcal{D}_{P,T}(\lambda)$ is $O(wt(T))$. We classify the chambers in $\mathcal{D}_{P,T}(\lambda)$ into three types and deal with them separately. Let m be the number of vertices of T.

Chambers with vertices of T.

There are at most m such chambers and all of them are small. The total perimeter of these chambers is $O(m\lambda)$ which approaches zero as λ approaches zero.

Square chambers without any vertex of T.

Let C be such a chamber. Let Δ be the triangle in T that covers the center of C. Our approach is to charge the perimeter of C to a nearby edge of T.

`Case 1.1:` A vertex w of Δ lies inside $8C$. Let e and f be the two edges of Δ incident on w. Since the vertices of Δ lie outside C, the length of some edge of Δ is at least $size(C)/2$. Triangle inequality implies that $|e| + |f| \geq size(C)/2$, so $\max\{|e|, |f|\} \geq size(C)/4$. We charge the perimeter of C to e or f whichever is longer. Observe that C lies inside a copy of $9C$ centered at w.

`Case 1.2:` No vertex of Δ lies inside $8C$. Let $B = parent\_box(C)$ and let $X = parent\_poly(C)$. Since X is crowded in B, $Z(B, X)$ has a vertex of T. It follows that $Z(B, X)$ has a vertex v of T that can see the center c

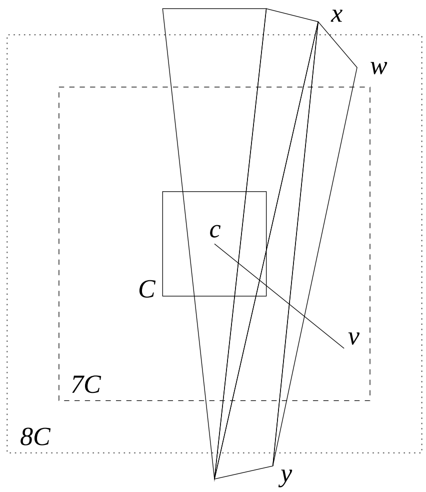

Figure 2: The solid square is C. The dashed and dotted squares are $7C$ and $8C$ respectively.

of C. The line segment cv stabs a sequence of triangles in T. We visit this sequence in order starting from Δ. Since cv lies inside $Z(B, X) \subseteq 7C$, we eventually reach a triangle Δ' such that a vertex w of Δ' lies inside $8C$. See Figure 2 for an illustration. Let xy be the edge of Δ' at which we step into Δ'. Observe that $|xy| \geq 2size(C)$ since x and y lie outside $8C$ and $xy \cap 7C \neq \emptyset$ (xy crosses cv). Triangle inequality implies that $|wx|+|wy| \geq 2size(C)$, so $\max\{|wx|, |wy|\} \geq size(C)$. We charge the perimeter of C to wx or wy whichever is longer. Observe that C lies inside a copy of $9C$ centered at w.

In the above two cases, if an edge e of T receives a charge of $4size(C)$ from a square chamber C, then $|e| \geq size(C)/4$. Moreover, C lies inside a copy of $9C$ centered at an endpoint of e. Within the two copies of $9C$ centered at the endpoints of e, there are $O(1)$ square chambers at the same quadtree level as C. They are the only square chambers that may charge to e at this quadtree level, so the maximum charge is $O(size(C))$. The sizes of square chambers decrease geometrically down the quadtree hierarchy. It follows that the total charge at e telescopes to $O(|e|)$. This shows that the total perimeter of square chambers in $\mathcal{D}_{P,T}(\lambda)$ is $O(wt(T))$.

Non-square chambers without any vertex of T.

Let C_1 be such a chamber. Our approach is to charge the perimeter of C_1 to the boundary of P.

`Case 2.1`: The length of some solid edge of C_1 is at least $size(C_1)/4$. We charge the perimeter of C_1 to this solid edge. Thus, the total perimeter of non-square chambers in case 2.1 is bounded by the perimeter of P which is $O(wt(T))$.

`Case 2.2`: The length of each solid edge of C_1 is less than $size(C_1)/4$. Lemma 5 implies that we can divide the non-square chambers in case 2.2 into two groups. For each chamber C_1 in the first group, C_1 is adjacent to a non-square chamber C_2 that satisfies Lemma 5(i). C_2 falls into case 2.1 and the perimeter of C_2 is charged to the perimeter of P. We charge the perimeter of C_1 by summing the perimeter of C_2 in case 2.1 once more. Since the chamber complexity is $O(1)$ by Observation 3, the perimeter of C_2 is summed a constant number of times for chambers adjacent to C_2. Thus, the $O(wt(T))$ bound in case 2.1 is not affected. For each chamber C_1 in the second group, C_1 is adjacent to a chamber C_2 that satisfies Lemma 5(ii). At most m chambers satisfy Lemma 5(ii). For each such C_2, there are $O(1)$ adjacent small chambers and their total perimeter is $O(\lambda)$ (chamber complexity is $O(1)$ by Observation 3). Thus, the total perimeter of non-square chambers in the second group is $O(m\lambda)$ which is negligible for sufficiently small λ.

□

Our approximate MWSTs are computed in two steps. We construct $\mathcal{S}_P(\delta)$ for some appropriate δ. Afterwards, we triangulate the chambers in $\mathcal{S}_P(\delta)$. The following lemma bounds the weight of the triangulation obtained.

Lemma 7 *Triangulating $\mathcal{S}_P(\delta)$ produces a Steiner triangulation of P with $O(wt(T)+\delta n)$ weight, where $wt(T)$ is the weight of any Steiner triangulation T of P and n is the number of vertices of P.*

Proof. The weight of $\mathcal{S}_P(\delta)$ is $O(wt(T))$ by Lemma 6. By Lemma 2, triangulating chambers with at most one vertex of P increases the weight by a constant factor. The chambers with more than one vertex of P are small, so their sizes are at most δ. Moreover, the total complexity of these chambers is $O(n)$. It follows that triangulating these chambers increases the weight by an additive $O(\delta n)$ term. □

5 A fast approximation algorithm

In this section, we present an approximate MWST with $O(n \log n)$ complexity and an $O(n \log n)$-time algorithm to construct it. Let h be the number of holes in P. We first compute h non-crossing diagonals to connect the hole boundaries and the outer boundary of P. If we view the holes and the outer boundary as vertices of a graph, then the h non-crossing diagonals form a spanning tree. Thus, we call them *spanning diagonals*. One can triangulate P in $O(n \log n)$ time [4] and then traverse the triangulation edges to identify the spanning diagonals. Figure 3 shows an example.

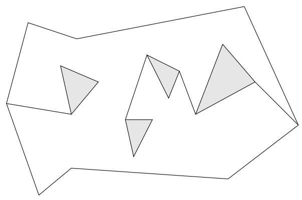

Figure 3: The line segments attached to the holes are the spanning diagonals.

We compute all subpolygons at one quadtree level before splitting the crowded ones and proceeding to the next level. This ordering is important for identifying crowded subpolygons efficiently. In Section 5.1, we show that subpolygons at a quadtree level can be processed in linear time to generate subpolygons at the next quadtree level. In Section 5.2, we analyze the running time of the approximation algorithm. For the sake of completeness, we show in Section 5.3 that $\Omega(n \log n)$ is a lower bound on the running time.

5.1 Processing at a quadtree level

Invariants We keep track of the following information for each subpolygon X of a quadtree box B inductively.

`Invariant 1.` The number, n_X, of vertices of P that X has.

`Invariant 2.` The *c-subpolygon* X^* which is obtained by cutting X along a subset of spanning diagonals.

`Invariant 3.` The set, A_X, of subpolygons and chambers currently adjacent to X.

Figure 4 shows examples of c-subpolygons. The boundary of B is divided into intervals that are incident on c-subpolygons and intervals that are not.

We keep track of these intervals ordered around B.

`Invariant 4.` *intervals*(B).

Some c-subpolygons may have cuts along spanning diagonals. These cuts produce degenerate intervals which are points on the boundary of B. We treat these degenerate intervals as not incident on any c-subpolygon. Each interval incident on a c-subpolygon is associated with the name of the c-subpolygon. The other intervals are marked as *unused*.

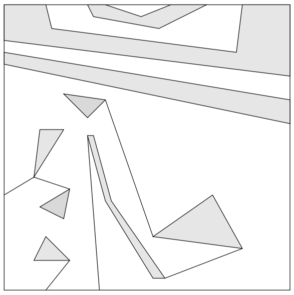

Figure 4: There are four c-subpolygons, but only one has cuts along spanning diagonals.

In the basis case, B is the bounding box of P; there is only one subpolygon $X = P$; $n_P = n$; P^* is obtained by cutting along all spanning diagonals; and A_P is empty. *intervals*(B) can be computed by sorting the intersections of the boundaries of B and P^*. This takes $O(n \log n)$ time.

Test for crowdedness If $n_X > 1$ and $width(B) > \delta$, then X is crowded in B. Suppose that $n_X \leq 1$. Let A be the subset of A_X that contains subpolygons and chambers at the same quadtree level as X. $A_X - A$ consists of normal chambers created earlier. So $Z(B, X) = (\bigcup_{Y \in A} Y) \cup (\bigcup_{Y \in A_X - A} Y \cap 3B) \cup X$. The complexity of X is $O(1)$ as $n_X \leq 1$. It follows that both $|A_X|$ and $|A|$ are $O(1)$. We compute $\sum_{Y \in A} n_Y$ in constant time. We traverse the boundaries of chambers in $A_X - A$ to count the vertices of P in $3B$. By Lemma 2, chambers in $A_X - A$ have constant complexities, so the counting takes constant time. Thus, we can decide in constant time whether X is crowded in B.

Lemma 8 *The crowdedness of a subpolygon can be tested in constant time.*

Helper polygon To prepare for splitting crowded subpolygons, we merge the corresponding c-subpolygons into a helper (simple) polygon Q_B. Scan *intervals*(B) to find intervals incident on the non-crowded c-subpolygons. Mark these intervals as unused. In doing so, if two unused intervals become adjacent on the same side of B, then combine them. Afterwards, choose a minimal subset of unused intervals to connect the crowded c-subpolygons together. Figure 5(a) shows an example. Q_B is obtained by doubling each unused interval chosen. Some edges of Q_B may lie on the boundary of B. We push these edges just outside B. Figure 5(b) shows an example. Both the time needed to construct Q_B and the complexity of Q_B are bounded by the total complexity of the c-subpolygons of B.

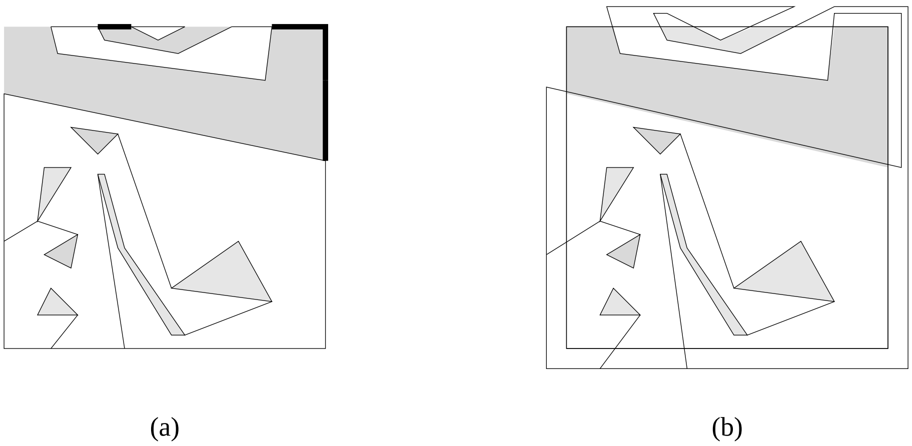

Figure 5: Figure (a) shows the crowded c-subpolygons in Figure 4. As an example, a c-subpolygon in Figure 4 is assumed to be non-crowded and removed. The unused intervals connecting the c-subpolygons are shown in bold. Figure (b) shows the doubling and perturbation.

Polygon clipping Let B_q be a quadrant of B. We discuss a procedure to clip Q_B within B_q which is a crucial step in splitting. The procedure returns the following output:

1. $Q_B \cap B_q$.
2. The partition $I(B_q)$ of the boundary of B_q into intervals inside and outside Q_B. $I(B_q)$ is ordered around B_q. Each interval in $I(B_q)$ is associated with the name of the polygon in $Q_B \cap B_q$ incident on it.

The processing time is $O(|Q_B|)$, where $|Q_B|$ denotes the complexity of Q_B. Readers who are familiar with clipping simple polygons may skip this part.

Let s be a side of B_q. Traverse the boundary of Q_B and report the intersections with s ordered along the boundary of Q_B. This takes $O(|Q_B|)$ time and generates $O(|Q_B|)$ intersections. Apply a linear-time Jordan sorting algorithm (e.g., see [8]) to order these intersections along s. The sorted intersections partition s into intervals inside and outside Q_B. The above is repeated for the other three sides of B_q. In the end, we obtain $I(B_q)$. The cardinality of $I(B_q)$ is $O(|Q_B|)$.

Label the endpoints of each interval as source and destination such that B_q lies locally to the right of each interval directed from its source to its destination. Pick an interval I inside Q_B and start at the destination of I. Turn in the anti-clockwise direction to find an edge of Q_B. Follow this edge of Q_B and apply the same turn-and-follow strategy at each vertex encountered until the source of another interval I' is reached. Jump to the destination of I' and repeat. When we cycle back to the source of I, we have discovered one simple polygon in $Q_B \cap B_q$. We associate the name of this polygon with I and all intervals encountered before cycling back to the source of I. Figure 6 shows an example. Afterwards, pick another interval inside Q_B that has not been traversed before and repeat the above. This continues until all intervals of $I(B_q)$ inside Q_B have been traversed. In the end, we obtain $Q_B \cap B_q$. The processing time is $O(|Q_B|)$. The total complexity of polygons in $Q_B \cap B_q$ is also $O(|Q_B|)$.

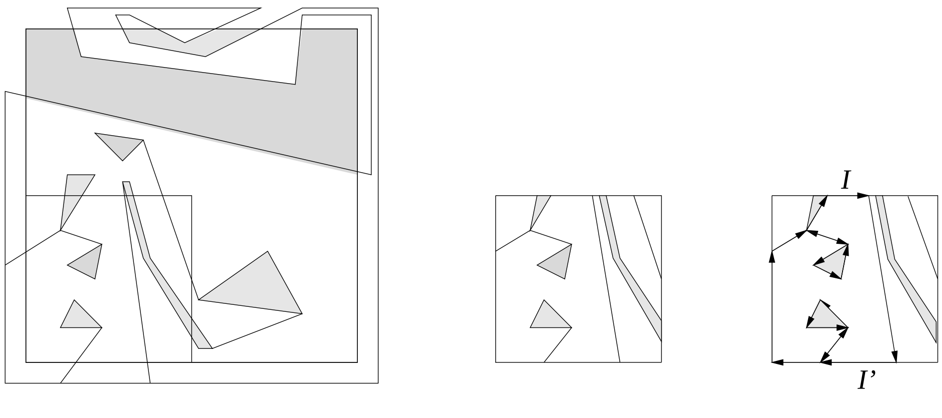

Figure 6: The figure on the left shows Q_B. The lower left square is the quadrant B_q. The figure in the middle shows the set of polygons in $Q_B \cap B_q$. The figure on the right shows the traversal to identify one polygon in $Q_B \cap B_q$.

Splitting of crowded subpolygons The first step is to clip Q_B within each quadrant B_q of B. The clipping procedure returns $Q_B \cap B_q$ and $I(B_q)$. Its running time is bounded by the total complexity of c-subpolygons. Given a c-subpolygon X^*, the number of spanning diagonals cut to obtain X^* from X is equal to the number of holes in X. This shows that the complexity of X^* is bounded by the complexity of X. Thus, the total complexity of c-subpolygons is $O(K_B)$, where K_B is the total complexity of subpolygons of B.

To recover the subpolygons of B_q, the polygons in $Q_B \cap B_q$ have to be merged appropriately. Construct the dual graph G of the subdivision induced by $Q_B \cap B_q$. There is a vertex in G for each polygon in $Q_B \cap B_q$. Two vertices (possibly the same) in G are connected by an edge for each edge shared between the two corresponding polygons (possibly the same). G may have self-loops and two vertices may be connected by more than one edge, but this is not a concern. Traverse G in linear time to construct a spanning forest F_G. Finally, we glue the polygons in $Q_B \cap B_q$ together at the dual of the edges of F_G. In doing so, we also merge intervals in $I(B_q)$ correspondingly and update the names of polygons associated with these intervals. The merging takes $O(K_B)$ time. Figure 7 shows an example.

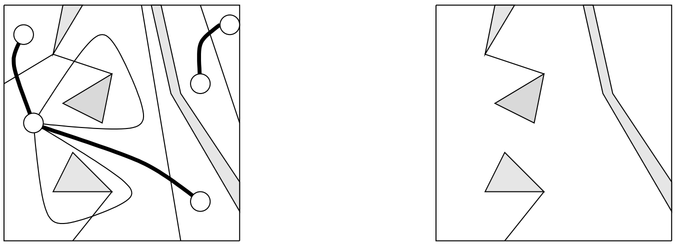

Figure 7: In the figure on the left, the disks and the curves comprise the dual graph G of $Q_B \cap B_q$ in Figure 6. The curves in bold are the edges of F_G. The figure on the right shows the merged polygons.

Each subpolygon Y of B_q is a connected component of $X \cap B_q$ for some crowded subpolygon X of B. So Y is the union of polygons in a component of F_G. The merging glues these polygons together to form a simple polygon Y^*. Y^* is the simple polygon required by invariant 2. The updated $I(B_q)$ is $intervals(B_q)$ required by invariant 4. To maintain invariant 1, we traverse the boundary of Y^* once to compute n_Y.

Lemma 9 *Let K_B be the total complexity of subpolygons of a quadtree box*

B. It takes $O(K_B)$ time to split B and its crowded subpolygons. Moreover, invariants 1, 2 and 4 for the resulting subpolygons and quadtree boxes can be maintained in the same time bound.

Adjacency maintenance Let ℓ be the current quadtree level. We discuss how to maintain invariant 3 after splitting all crowded subpolygons at level ℓ. In the process, the boundaries of chambers created earlier will be subdivided if necessary.

We first compute the adjacency between subpolygons at level $\ell + 1$ and chambers created earlier. Let X be a crowded subpolygon of B at level ℓ. Let A' be the set of chambers in A_X. Chambers in A' are normal. If A' is empty, then there is nothing to do. Suppose not. X has at most one vertex of P; otherwise chambers in A' would be crowded which is impossible. So the complexity of X is $O(1)$ which implies that $|A'| = O(1)$. Let B_q be a quadrant of B. Let Y be a subpolygon in $X \cap B_q$. We search A' sequentially to find chambers that are in contact with Y. The search time is bounded by the complexity of Y since $|A'|$ and the complexities of chambers in A' are $O(1)$. We enter each chamber found into A_Y and split the boundary of the chamber accordingly. The above is repeated for all subpolygons at level $\ell+1$. The processing time is bounded by the total complexity of subpolygons at level $\ell + 1$. It follows from Lemma 9 that this time bound is $O(K)$, where K is the total complexity of subpolygons at level ℓ.

It remains to compute the adjacency among subpolygons at level $\ell + 1$. By Lemma 9, $intervals(B_q)$ is available for every quadtree box B_q at level $\ell + 1$. Make a copy L of $intervals(B_q)$. Remove the unused intervals from L. When a degenerate unused interval is removed, combine the two neighboring intervals if they lie on the same side of B_q. (A degenerate unused interval is a point and the two neighboring intervals are incident on the same subpolygon of B_q.) The intervals in the modified L are the intersections of the boundaries of B_q and the subpolygons of B_q. Divide L into four lists, $L(B_q, s)$, that contain intervals on each side s of B_q. Finally, for every pair of quadtree boxes B_q and B'_q at level $\ell + 1$ with a common side s, we sequentially scan $L(B_q, s)$ and $L(B'_q, s)$. This allows us to update the A_Y's of all subpolygons Y at level $\ell + 1$ incident on s in linear time.

Lemma 10 *Let ℓ be the current quadtree level. Let K be the total complexity of subpolygons at level ℓ. After splitting the crowded subpolygons at level ℓ, it takes $O(K)$ time to maintain invariant 3 for subpolygons at* *level $\ell + 1$.*

5.2 Analysis

Lemmas 8–10 imply that the processing time for one quadtree level is bounded by the total complexity of subpolygons at that level. We show that this is $O(n)$.

Lemma 11 *Let n be the number of vertices of P. The total complexity of subpolygons at a quadtree level is $O(n)$.*

Proof. Let X be a subpolygon of a quadtree box B. Our approach is to charge the vertices of X to some nearby vertices of P.

`Case 1`: X has some vertex of P. Let v be a vertex of X that is not a vertex of P. If v is connected to some vertex w of P by an edge of X, then we charge v to w. Otherwise, either v is a corner vertex of B or v is incident on an edge of X that cuts off a corner of B. This implies that X has at most $3n_X + 8$ vertices, where n_X is the number of vertices of P that X has. Thus, it suffices to leave $O(1)$ charge at each of the n_X vertices.

`Case 2`: X does not have any vertex of P. Clearly, X has at most eight vertices. Let B' be the parent of B.

(i) Some edge h of P incident on X does not cross $3B'$ completely. Let w be an endpoint of h inside $3B'$. Let h' be the other edge of P incident on w. We charge the vertices of X to w. X lies inside a copy of $8B$ centered at w. Within this copy of $8B$ centered at w, there are $O(1)$ subpolygons at the same quadtree level as X that h or h' can be incident on. These are the only subpolygons that may charge to w in case 2(i). It follows that the total charge accumulated at w is $O(1)$.

(ii) Each edge of P incident on X crosses $3B'$ completely. Let X' be the subpolygon of B' that contains X. Since X' is crowded in B', $Z(B', X')$ has a vertex of P. This implies that $Z(B', X')$ has a vertex w of P such that w can see some point in X. The vertices of X are charged to w. We analyze the charge accumulated at w. We claim that X is the only subpolygon of B that falls into case 2(ii) and is visible from w.

If there are two such subpolygons, then they are separated by some edge h of P that crosses $3B'$ completely. So h blocks w from one of the two subpolygons, which is a contradiction.

B is contained in a copy of $8B$ centered at w. Within this copy of $8B$ centered at w, there are $O(1)$ quadtree boxes at the same quadtree level as B. Only the subpolygons of these quadtree boxes may charge to w. By our claim, at most one subpolygon per such box may do so. It follows that the total charge accumulated at w is $O(1)$.

$\square$

We have all the ingredients to derive the $O(n \log n)$-time approximation algorithm.

Theorem 2 *Let P be a polygon with n vertices. There is a Steiner triangulation of P with $O(n \log n)$ complexity and $O(wt(T))$ weight, where $wt(T)$ is the weight of any Steiner triangulation T of P. Moreover, this triangulation can be computed in $O(n \log n)$ time.*

Proof. We compute $\mathcal{S}_P(1/n)$ and then triangulate it. Denote this triangulation by T'. By Lemma 7, the weight of T' is $O(wt(T)+1)$. Since P must be in contact with two parallel sides of its bounding box, the perimeter of P is no less than 2. So $wt(T) \geq 2$ and the weight of T' is $O(wt(T))$. Since $\delta = 1/n$, there are $O(\log n)$ quadtree levels. It follows from Lemma 11 that the complexity of T' is $O(n \log n)$. Lemmas 8–11 imply that T' can be constructed in $O(n \log n)$ time. $\square$

5.3 Lower bound

The $O(n \log n)$ running time is optimal in the algebraic decision tree model. We show that sorting can be reduced to the problem of constructing a Steiner triangulation of a polygon with holes. Let $x_1 \cdots x_n$ be n integers to be sorted. Let R be the rectangle $[\min_i\{x_i\} - 1, \max_i\{x_i\} + 1] \times [0, 1]$. Add the points $(x_i, 0.5)$, $1 \leq i \leq n$, as holes to R to form a polygon P. Clearly, P can be constructed in $O(n)$ time. Suppose that a Steiner triangulation of P can be computed in $T(n)$ time. The complexity of the triangulation is at most

$T(n)$. We search for the triangle containing the point $(\min_i\{x_i\} - 1, 0.5)$ in $T(n)$ time. Afterwards, we traverse the triangulation by following the horizontal ray from $(\min_i\{x_i\} - 1, 0.5)$ to $(\infty, 0.5)$. We will encounter the holes in increasing x-coordinates, so the points x_i's can be reported in sorted order. The overall processing time is $O(T(n)+n)$ which implies that $T(n) = \Omega(n \log n)$.

References

[1] B. Aronov and S. Fortune, Average-case ray shooting and minimum weight triangulation, in *Proc. 13th ACM Symposium on Computational Geometry*, 204–211, 1997.

[2] S. Arya, S.-W. Cheng, and D.M. Mount, Approximation algorithms for multiple-tool milling, *Proc. 14th ACM Symposium on Computational Geometry* 1998, 297–306. Also to appear in the *International Journal of Computational Geometry and Applications.*

[3] M. de Berg, M. Katz, A.F. van der Stappen, and J. Vleugels, Realistic input models for geometric algorithms, in *Proc. 13th ACM Symposium on Computational Geometry*, 294–303, 1997.

[4] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: algorithms and applications*, Springer, 1998.

[5] M. Bern and D. Eppstein, Mesh generation and optimal triangulation, *Computing in Euclidean Geometry*, D.-Z. Du and F.Hwang (eds.), World Scientific, 1992, 23–90.

[6] M. Bern, D. Eppstein, and J. Gilbert, Provably good mesh generation, *Journal of Computer and System Sciences*, 48 (1994) 384–409.

[7] D. Eppstein, Approximating the minimum weight Steiner triangulation, *Discrete & Computational Geometry*, 11 (1994) 163–191.

[8] K.Y. Fung, T.M. Nicholl, R.E. Tarjan, and C.J. Van Wyk, Simplified linear-time Jordan sorting and polygon clipping. *Information Processing Letters*, 35 (1990) 85–92.

[9] J.S.B. Mitchell, D.M. Mount, and S. Suri, Query-sensitive ray shooting, *International Journal of Computational Geometry & Applications*, 7 (1997) 317–347.