Computing Optimal Diameter-Bounded Polygon Partitions

Mirela Damian* Sriram V. Pemmaraju †

Abstract

The *minimum α-small partition* problem is the problem of partitioning a given simple polygon into subpolygons, each with diameter at most α, for a given $\alpha > 0$. This paper considers the version of this problem that disallows Steiner points. This problem is motivated by applications in mesh generation and collision detection. The main result in the paper is a polynomial time algorithm that solves this problem, and either returns an optimal partition or reports the non-existence of such a partition. This result contrasts with the recent NP-completeness result for the minimum α-small partition problem for polygons with holes (C. Worman, 15th Canadian Conference on Computational Geometry, 2003). Even though the running time of our algorithm is a polynomial in the input size, it is prohibitive for most real applications and we seek faster algorithms that approximate an optimal solution. We first present a faster 2-approximation algorithm for the problem for simple polygons and then a near linear-time algorithm for convex polygons that produces, for any $\varepsilon > 0$, an $(\alpha + \varepsilon)$-small partition with no more polygons than in an optimal α-small partition. We also present an exact polynomial time algorithm for the minimum α-small partition problem with the additional constraint that each piece in partition be convex.

1 Introduction

A *decomposition* of a polygon P is a set of polygons whose union is exactly P; if the polygons in the set are allowed to overlap, the decomposition is called *covering*; otherwise, it is called a *partition*. The *diameter* of a simple polygon is the diameter of a smallest enclosing circle. A polygon with diameter no greater than a fixed value α is said to be *α-small*. The *minimum α-small partition* problem takes as input a simple polygon P and a positive real α and seeks a partition of P into fewest α-small polygons.

This problem is motivated by applications in collision detection algorithms that are used in the simulation of physically based motion [3]. Such real-time applications commonly use the bounding box heuristic, in which complex objects are approximated by simpler enclosing volumes (usually, boxes) whose pairwise intersection can be rapidly computed. To compute intersections between the actual objects, pairs of enclosing boxes are tested for intersection and if a pair of boxes intersect, only then the corresponding objects are tested. Clearly, fewer the "false" intersections (that is, a pair of boxes intersect, but the enclosed objects do not) more efficient the heuristic. It has been shown that if the objects whose motion is being simulated are well-shaped and are of the same relative size, then the bounding box heuristic works very well [14, 16]. These results motivate the use of a preprocessing step in collision detection algorithms that decomposes the given set of objects into well-shaped objects of roughly the same size. This preprocessing step imposes a significant overhead on the collision detection if it results in a large number of objects and so it is important to minimize the number of pieces into which the objects are decomposed.

Motivated by this application, preliminary work in [5] presents results on the minimum α-small partition problem and the *minimum α-fat partition problem*. The *aspect ratio* of a polygon is the ratio of its diameter to its width. The minimum α-fat partition problem seeks to find a partition of a given polygon into the fewest number of polygons, each of whose aspect ratio is at most α, for some given $\alpha > 0$.

*Department of Computer Science, Villanova University, Villanova, PA 19085, U.S.A. e-mail: mirela.damian@villanova.edu

†Corresponding author; Department of Computer Science, University of Iowa, Iowa 52242, U.S.A. e-mail: sriram@cs.uiowa.edu

In this paper we focus on the minimum α-small partition problem for simple polygons without holes; we also disallow Steiner vertices. For any real α, not all polygons have an α-small partition, if Steiner points are disallowed. We say that a polygon is α-*small decomposable* whenever it has an α-small partition. Also note that since Steiner vertices are disallowed, the problem requires the choice of a minimum set of diagonals that induces an α-small partition. In this setting we present as our main result, a dynamic programming based, polynomial time algorithm that solves the minimum α-small partition problem in time $O(mn^3)$. The algorithm outputs an optimal set of diagonals if the given polygon is α-small decomposable; otherwise it reports the non-existence of such a partition. Here n denotes the number of vertices and m denotes the number of edges in the visibility graph of the input polygon. This contrasts with the NP-completeness of the minimum α-small partition problem for polygons with holes, recently established by Worman [15]. To reduce the running time, we devise two approximation algorithms: the first is a 2-approximation algorithm that runs in $O(mn)$ time; the second is an algorithm that runs on convex polygons and produces, for any $\varepsilon > 0$, an $(\alpha + \varepsilon)$-small partition of size no greater than the size of an optimal α-small partition. Finally, we show how to extend these algorithms to produce partitions in which all polygons are restricted to be convex.

The minimum α-small partition problem is a classic example of the type of problems considered in the literature on polygon partitioning. Of all these problems, the problem of partitioning a polygon into a minimum number of convex polygons has received most attention. Keil and Snoeyink [11] provide an $O(n + r^2 \min\{r^2, n\})$ time solution to this problem, where n is the total number of vertices and r is the number of reflex vertices (vertices with interior angle greater than π). Their solution does not use Steiner points. Allowing Steiner points, Chazelle and Dobkin [2] give an intricate $O(n+r^3)$ algorithm to partition a simple polygon into fewest convex pieces. For polygons with holes, the problem becomes NP-hard [12], but can be approximated within a factor of 4 of optimal in time $O(n \log n)$ [7]. Kiel [9] has considered problems of partitioning a polygon into fewest spiral polygons, star-shaped polygons and monotone polygons. These problems become NP-hard on polygons with holes; see the survey papers [1, 10] for a summary of results on polygon partitioning problems.

2 Computing optimal α-small partitions

Suppose the input is a simple n-vertex polygon P with the vertices labeled 1 through n in counter clockwise order. A *diagonal* (i, j) of P is a line segment that joins two vertices i and j of P and lies entirely inside P. For ease of presentation, we refer to edges $(i, i+1)$ of P as diagonals also. The endpoints of a diagonal are *visible* from each other. The *visibility graph* of P is a graph whose nodes are vertices of P and whose edges are diagonals of P. For any diagonal (i, j) of P, with $i < j$, we denote by P_{ij} the subpolygon of P with vertices $i, i+1, \cdots j$ (see the shaded polygon in Figure 1b).

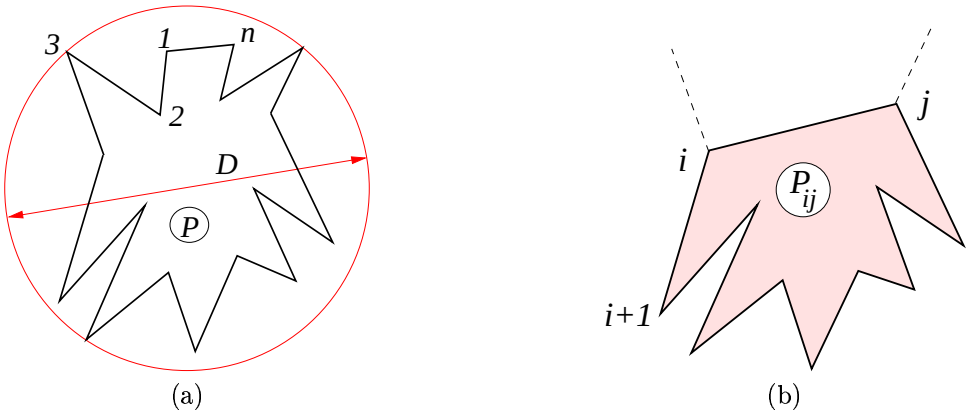

Figure 1: Definitions: (a) P is α-small if and only if $D \leq \alpha$, (b) the subpolygon P_{ij}.

To solve the minimum α-small partition problem, we consider diagonals (i, j), with $i < j$, in increasing order of $j - i$, and compute an optimal α-small partition O_{ij} of each P_{ij}. P_{1n} is the entire polygon and therefore O_{1n} is a solution to the minimum α-small partition problem.

2.1 Properties of O_{ij}

For the purpose of the discussion in this section, assume that P_{ij} is α-small decomposable and therefore O_{ij} is nonempty. Let G_{ij} be the visibility graph of P_{ij}. We attach a weight w_{kl} to each edge (k, l) of G_{ij} as follows. If P_{kl} is α-small decomposable, then w_{kl} is the size of an optimal α-small partition O_{kl} of P_{kl}. By convention, $w_{kl} = 0$ if $l = k + 1$. If P_{kl} is not α-small decomposable, then $w_{kl} = \infty$.

Let Q be the polygon in O_{ij} adjacent to (i, j) and let C be the smallest circle enclosing Q, as illustrated in Figure 2a. Clearly, Q is an α-small polygon. It is important to note that C is tangent to two or more vertices of Q and three such vertices suffice to completely define C. Also there is a circle C_1 of diameter exactly α that contains Q and passes through two vertices of P_{ij} (see Figure 2b).

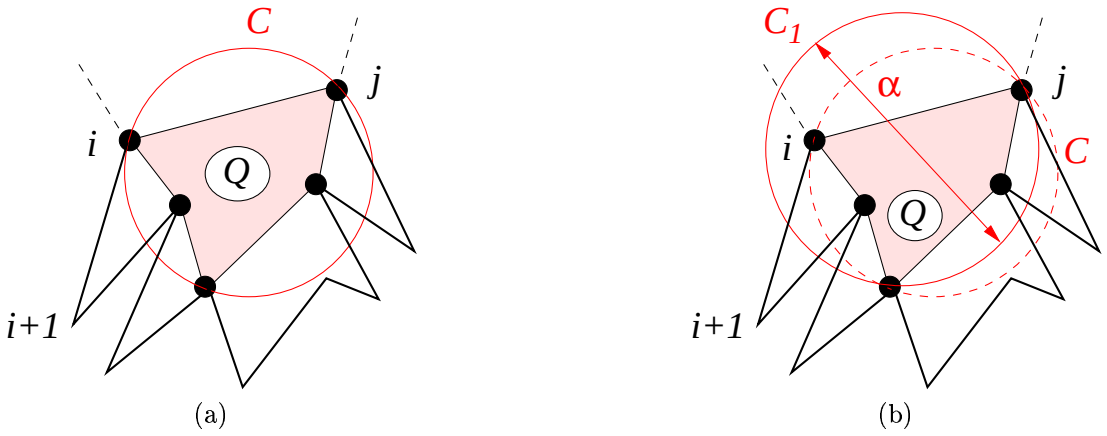

Figure 2: Properties of O_{ij} (a) Q in O_{ij} is adjacent to (i, j); Q's circumscribed circle C has diameter no greater than α. (b) C_1 has diameter exactly α and encloses Q.

Let $G_{ij}[C_1]$ be the subgraph of G_{ij} induced by diagonals of P_{ij} that lie inside C_1. Refer to Figure 3b for an example, which shows $G_{ij}[C_1]$ corresponding to P_{ij} and C_1 from Figure 3a. Define the *weight* of a path as the sum of the weights of its constituent edges. Then the path in $G_{ij}[C_1]$ induced by the vertices of Q is a path of minimum weight from i to j (see the path marked with thick lines in Figure 3b).

We will use these properties in determining the subpolygon Q in a minimum α-small partition of P_{ij}. The dynamic programming approach ensures that the optimal α-small partition of the polygonal pieces of P_{ij} obtained by removing Q are available at this point. Then the union of these optimal α-small partitions and Q gives us an optimal α-small partition of P_{ij}.

2.2 Computing O_{ij}

We use the properties of O_{ij} from section 2.1 to compute an optimal α-small partition O_{ij} of P_{ij}, using dynamic programming.

The base case is $j = i + 1$. In this case $O_{ij} = \perp$ if the length of the edge $(i, i + 1)$ exceeds α, otherwise O_{ij} is the empty set. Here $\perp$ indicates the absence of an α-small partition.

For $j > i+1$, first check if the diameter of P_{ij} exceeds α. If it does not then, P_{ij} is itself α-small and O_{ij} is simply P_{ij}. Otherwise consider all circles C of diameter α that contain both i and j and pass through two non-reflex vertices of P. Assuming that r of the n vertices of P are reflex, there are $O((n - r)^2)$ such circles. For each such circle C, let $G_{ij}[C]$ be the visibility graph of P_{ij}, restricted to vertices of P_{ij} that lie inside C. Each edge (k, l) of $G_{ij}[C]$ has a weight w_{kl}, as defined in section 2.1 (see Figure 3b). At this

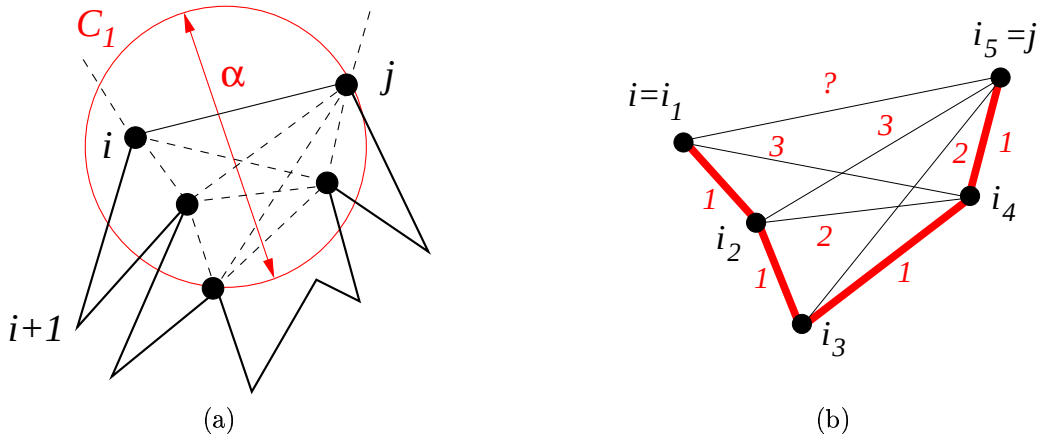

Figure 3: (a) Marked with dashed lines are diagonals of P_{ij} that lie inside C_1 (b) The graph $G_{ij}[C_1]$; a minimum weight path from i to j is $\{i_1, i_2, i_3, i_4, i_5\}$, of weight 4; this is not unique: another minimum weight path is $\{i_1, i_2, i_5\}$.

stage of the dynamic programming algorithm, we have optimal partitions O_{kl} for all polygons P_{kl} that are smaller than P_{ij}. This implies that all weights in $G_{ij}[C]$ are well defined, except for w_{ij}, which we initially set to ∞ and seek to improve.

To compute the polygon Q adjacent to (i, j) in P_{ij}, we compute, for each choice of a circle C, a path $p_{ij}[C]$ of minimum weight from i to j in $G_{ij}[C]$. Then Q is defined by the vertices of the path p_{ij} of smallest weight taken over all circles C, that is, p_{ij} is the lightest path among $\{p_{ij}[C] \mid \text{over all circles } C\}$. Note that Q has at least one vertex k visible from both i and j. Therefore, in computing $p_{ij}[C]$, we pick a point k in $G_{ij}[C]$, adjacent to both i and j, with $i < k < j$, that minimizes the weight of the path $p_{ik}[C] \cup p_{kj}[C]$. If we can such find k, then clearly $p_{ij}[C] = p_{ik}[C] \cup p_{kj}[C]$. Otherwise, $p_{ij}[C] = \perp$. Thus, $\perp$ indicates the absence of a path.

We now use p_{ij} in determining an optimal α-small partition O_{ij} of P_{ij} as follows. If $p_{ij} = \perp$, then $O_{ij} = \perp$. Otherwise note that the polygon Q induced by the vertices of p_{ij} is α-small. In this case we set $O_{ij} = \cup_{(k,l) \subset E(p_{ij})} O_{kl} \cup \{Q\}$, where $E(p_{ij})$ denotes the set of edges along p_{ij}.

Theorem 1 *O_{ij} is a minimum size α-small partition of P_{ij}.*

Proof: The proof is by induction on $(j - i)$. The base case is when $j - i = 1$. In this case P_{ij} is a degenerate polygon with only one edge. Then O_{ij} is the empty set if the length of the edge (i, j) does not exceed α; otherwise $O_{ij} = \perp$.

The inductive hypothesis is that for some natural number s and for all i, j, such that $1 \leq j - i \leq s$, O_{ij} is a minimum size α-small partition of P_{ij}. To prove the inductive step we consider the α-small partition O_{ij} for some i, j such that $j - i = s + 1$.

If P_{ij} is not α-small decomposable, then clearly $O_{ij} = \perp$. So suppose that P_{ij} has an α-small partition. If P_{ij} is α-small, then the partition is the polygon itself. Otherwise let Q^* be the polygon adjacent to (i, j) in a *minimum* α-small partition O^*_{ij} of P_{ij}. Since Q^* is α-small, our algorithm picks a circle C of diameter α that entirely encloses Q^*. Then the vertices of Q^* induce a path p^*_{ij} in $G_{ij}[C]$. The path p^*_{ij} is a candidate for the minimum weight path p_{ij} computed by our algorithm and therefore the size of $O_{ij} = \cup_{(k,l) \subset E(p_{ij})} O_{kl} \cup Q$ is no greater than the size of $O^*_{ij} = \cup_{(k,l) \subset E(p^*_{ij})} O_{kl} \cup Q^*$, which is a minimum size α-small partition. It follows that O_{ij} is a minimum size α-small partition. □

Theorem 2 *Let P be an n-vertex polygon with r reflex vertices and m edges in its visibility graph. There is an algorithm that, in $O(m(n-r)^2 n)$ time, computes a minimum size α-small partition of P, if P is α-small decomposable; otherwise the algorithm indicates that P is not α-small decomposable.*

Proof: Our algorithm computes α-small partitions for $O(m)$ polygons P_{ij}. For each polygon P_{ij}, there are $O((n-r)^2)$ circles C to consider. For a given circle C, computing $p_{ij}[C]$ takes $O(n)$ time. Thus the running time of the algorithm is $O(m) \times O(n-r)^2 \times O(n) = O(m(n-r)^2 n)$. This however requires two preprocessing steps.

(a) The visibility graph of P needs to be precomputed. This can be done in $O(m + n \log\log n)$ time using an algorithm of Hershberger [8].

(b) The set of polygons P_{ij} which are α-small needs to be precomputed. This can be done in $O(m \cdot n)$ time by computing a smallest enclosing circle for each subpolygon P_{ij} and comparing its diameter with α. The running time follows from the fact that Meggido's algorithm [13] solves the smallest enclosing circle problem for any set of n points in $\Theta(n)$ time and the fact that there are $O(m)$ subpolygons P_{ij} to consider.

□

The algorithm presented in this section proves that the minimum α-small partition problem is polynomial time solvable, however, it may not be practical because its running time if a high degree polynomial in the input size. In the following we present faster approximation algorithms.

3 A 2-approximation algorithm

In [6] we report an $O(mn)$ 4-approximation result for the α-small partition problem. This result is described in detail in [5]. However, the algorithm presented in [5] produces α-small partitions in which all subpolygons are restricted to be convex. We have improved this result and developed an algorithm to produce simple partitions of size within a factor of 2 of optimal in time $O(mn)$, which we describe next. This algorithm is quite similar to the well-known dynamic programming algorithm that computes a minimum weight triangulation of a simple polygon [4]. The algorithm can be easily adapted to produce convex partitions of size within a factor of 2 of optimal in time $O(mn)$ as well.

The algorithm computes a partition D_{ij} of each polygon P_{ij}. For $j = i+1$, D_{ij} is the empty set if the length of the edge (i,j) is no greater than α and is $\perp$, otherwise. To compute an α-small partition D_{ij} of P_{ij} for any $j > i+1$, first check if P_{ij} is α-small. If so, D_{ij} is simply $\{P_{ij}\}$. Otherwise, pick a point k, $i < k < j$, visible from both i and j, that minimizes the number of polygons in $D_{ik} \cup D_{kj} \cup \{\triangle ikj\}$. It is possible that for every k, $i < k < j$, visible from i and j, either $D_{ik} = \perp$ or $D_{kj} = \perp$ or $\triangle ijk$ is not α-small. In this case, D_{ij} is set to $\perp$.

If the visibility graph of P has m edges, the running time of the algorithm is $O(mn)$. To see this first note that there are $O(m)$ edges (i,j) to consider, $O(n)$ vertices k for each edge (i,j). For each edge (i,j) and each vertex k, $D_{ik} \cup D_{kj} \cup \{\triangle ijk\}$ and its size can be computed in $O(1)$ time.

We now show that the partition produced by this algorithm is within 2 times the optimal. For any pair of vertices i and j, $i < j$, visible from each other, let $d_{ij} = |D_{ij}|$, with $d_{ij} = \infty$ if $D_{ij} = \perp$. Suppose that P_{ij} is α-small decomposable and let O_{ij} denote an optimal α-small partition of P_{ij}. We will now show that $d_{ij} \leq 2|O_{ij}| - 1$.

If P_{ij} is α-small, then $d_{ij} = |O_{ij}| = 1 \leq 2|O_{ij}| - 1$. Otherwise, let Q be the polygon in O_{ij} that contains the edge (i,j). Vertex i has two neighbors in Q, j is one of them, let us call the other neighbor ℓ. D_{ij} is at least as small as $D_{i\ell} \cup D_{\ell j} \cup \{\triangle ij\ell\}$ and so $d_{ij} \leq d_{i\ell} + d_{\ell j} + 1$.

We now claim that O_{ij} contains at least as many polygons as in $O_{i\ell} \cup O_{\ell j}$ and so $|O_{ij}| \geq |O_{i\ell}| + |O_{\ell j}|$. To see this partition O_{ij} into three sets of polygons: $A \subseteq O_{ij}$ contains polygons in O_{ij} that are contained in $P_{i\ell}$; $B = \{Q\}$; and $C = O_{ij} - A - B$. Construct a new polygon Q' from Q by deleting the edges (i,ℓ) and (j,i) and replacing these by (ℓ, j). Now note that A is an α-small decomposition of $P_{i\ell}$ and $\{Q'\} \cup C$ is an α-small decomposition of $P_{\ell j}$. Also note that $|O_{ij}| = |A| + |\{Q'\} \cup C|$. Finally, letting $O_{i\ell}$ and $O_{\ell j}$ denote optimal α-small partitions of $P_{i\ell}$ and $P_{\ell j}$ respectively, we have that $|O_{i\ell}| \leq |A|$ and $|O_{\ell j}| \leq |\{Q'\} \cup C|$ and therefore $|O_{ij}| \geq |O_{i\ell}| + |O_{\ell j}|$.

Using the inductive hypothesis that $d_{i\ell} \leq 2|O_{i\ell}| - 1$ and $d_{\ell j} \leq 2|O_{\ell j}| - 1$ we get

$$\begin{aligned} d_{ij} &\leq (2|O_{i\ell}| - 1) + (2|O_{\ell j}| - 1) + 1 \\ &\leq 2(|O_{i\ell}| + |O_{\ell j}|) - 1 \\ &\leq 2|O_{ij}| - 1 \end{aligned}$$

The inductive step of this proof goes through for any constant c used instead of the 2, however the base case holds only for $c \geq 2$. The above proof yields the following theorem.

Theorem 3 *Let* OPT *be the size of an optimal α-small partition of a polygon P. There is an algorithm that produces an α-small partition of P of size d, $d \leq 2|\text{OPT}| - 1$, in time $O(mn)$.*

For a tight example refer to polygon with $2n$ vertices shown in Figure 4(a). The vertices of the polygon are numbered 1 through $2n$ in counter clockwise order, as shown in the figure. Suppose that for i, $1 \leq i \leq n$, vertex i has coordinates $(i, 0)$ and for $n < i \leq 2n$, vertex i has coordinates $(2n - i + 1, 1)$. Thus every pair of adjacent vertices in the polygon are at unit distance from each other. Now let $\alpha = 1.5$. Then the diagonals that can be used in an α-small partition are shown in Figure 4(b). It is not hard to verify that an optimal α-small partition consists of n polygons, as shown in Figure 4(c), whereas the above algorithm produces an α-small partition with $2n - 1$ polygons, as shown in Figure 4(d).

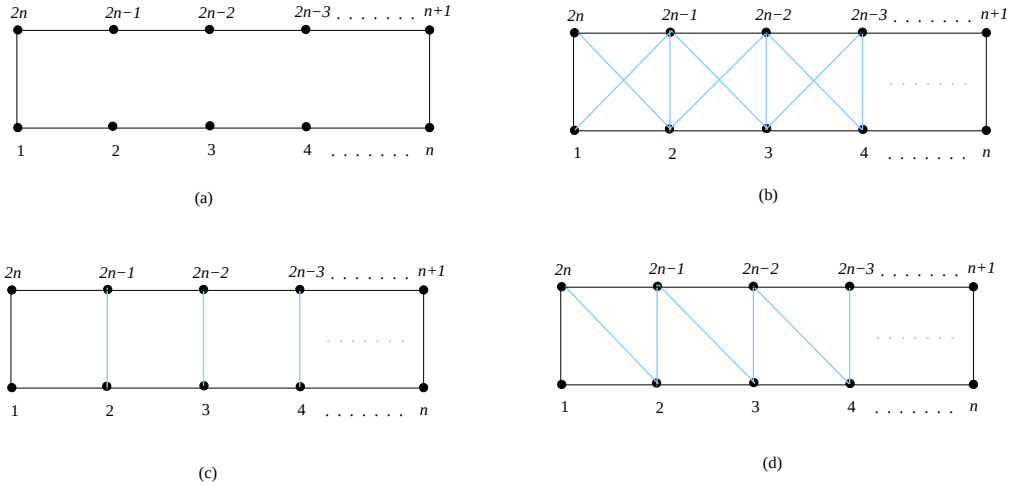

Figure 4: (a) A polygon P with $2n$ vertices, (b) all diagonals of P that can be used in an α-small partition, with $\alpha = 1.5$, (c) an optimal 1.5-small partition of P consisting of n polygons, and (d) a 1.5-small partition produced by the 2-approximation algorithm, consisting of $2n - 1$ polygons.

The algorithm described in this section works under arbitrary metrics.

4 Computing optimal $(\alpha+\varepsilon)$-small partitions of a convex polygon

We now show how to compute, for any $\varepsilon > 0$, an optimal $(\alpha + \varepsilon)$-small partition of a convex polygon P in time that is linear in n, but depends polynomially on α and $1/\varepsilon$. Here optimality refers to the fact that the number of $(\alpha + \varepsilon)$-small polygons that P is partitioned into is no more than the size of an optimal α-small partition of P. For ease of presentation, in this section we define the diameter of a polygon as the maximum distance between any pair of polygon points.

The main idea here is an initial "simplification" of P that replaces dense clusters of vertices of P by single vertices. This step is parameterized by ε and ensures that the resulting simplified polygon "approximates" P. This simplification can be done in linear time and after that we can use the dynamic

programming algorithm described in Section 2.2 to compute an optimal partition of the simplified polygon in near linear time, with sufficiently small polygons. We then show how this partition of the simplified polygon can be transformed into an $(\alpha+\varepsilon)$-small partition of P with no increase in the number of polygons.

Let B be the smallest axis-parallel rectangle enclosing P. Place a grid of square cells on B, with the dimension of each cell being β, for some β whose value will be specified later. Without loss of generality assume that each vertex in P is in the *interior* of some grid cell. This can always be done because any grid for which this is not so can be perturbed to make this happen. For any cell ψ in B, let the restriction of the boundary of P to ψ be denoted P_ψ. Since P is convex, P_ψ contains at most 2 connected components. Now transform P into a polygon P_S as follows. For each cell ψ in B and for each connected component C of P_ψ, if C contains a vertex of P, replace C by the line segment connecting the two points of intersection of C with the boundary of ψ. The two points where C intersects the boundary of ψ become vertices of P_S. This transformation is shown in Figure 5. Consider the top-left grid cell in this figure; the boundary of P intersects this cell in one connected component, say C. C contains two vertices a and b of P and it intersects the boundary of the cell at points c_1 and c_2. In the transformation from P to P_S, C is replaced by the line segment connecting c_1 and c_2.

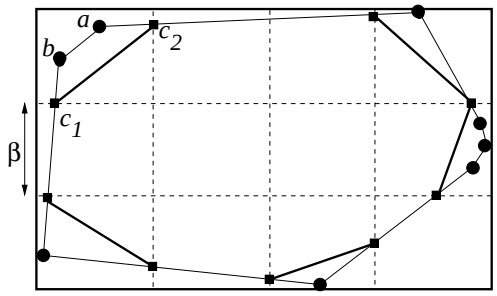

Figure 5: Polygon P is shown with vertices as disks. The vertices of P_S are shown as filled squares. The polygon P_S is contained in the polygon P.

It should be clear that P_S is a convex polygon wholly contained in P. Furthermore, for any vertex in P there is a vertex in P_S that is at a distance of at most $\beta\sqrt{2}$ from it and similarly for any vertex in P_S there is a vertex in P that is a distance of at most $\beta\sqrt{2}$ from it.

Lemma 4 *If P has an α-small partition of size s, then P_S has an $(\alpha + \sqrt{2}\beta)$-small partition of size at most s.*

Proof: Each vertex v in P lies in some cell ψ of B. Starting from v traverse the boundary of P in clockwise direction until a vertex w of P_S is encountered. Assign v to w and in general for any vertex v of P, denote by $A(v)$ the vertex v has been assigned to. Let $\mathcal{L}$ be an α-small partition of P given as a collection of diagonals of P. From this we construct a set $\mathcal{L}_S$ of diagonals of P_S as follows. For each diagonal (u, v) in $\mathcal{L}$, add the diagonal $(A(u), A(v))$ to $\mathcal{L}_S$, provided $A(u) \neq A(v)$.

We now verify that no two diagonals in $\mathcal{L}_S$ intersect. To obtain a contradiction suppose a pair of diagonals $(A(u_1), A(v_1))$ and $(A(u_2), A(v_2))$ do intersect. Recall that $A(u_1)$, $A(v_1)$, $A(u_2)$, and $A(v_2)$ are points on the boundary of P. Without loss of generality assume that these 4 points are in the order shown in Figure 6a, on the boundary of P. Then it is easy to see that the vertices u_1, v_1, u_2, and v_2 of P are as shown in the figure. Specifically, for any $p \in \{u_1, u_2, v_1, v_2\}$, if we traverse P in clockwise direction starting from p, we encounter $A(p)$ before $A(q)$ for any $q \in \{u_1, u_2, v_1, v_2\} - \{p\}$. This would mean that the diagonals (u_1, v_1) and (u_2, v_2) intersect – a contradiction. This implies that no two diagonals in $\mathcal{L}_S$ intersect and therefore the diagonals in $\mathcal{L}_S$ form a partition of P_S.

We now argue that every polygon in this partition has diameter at most $(\alpha + \sqrt{2}\beta)$. It suffices to show that for any pair of vertices u and v of P_S, if u and v belong to the same polygon in the partition induced by $\mathcal{L}_S$, then the distance between u and v is at most $(\alpha + \sqrt{2}\beta)$. If u and v belong to the same polygon

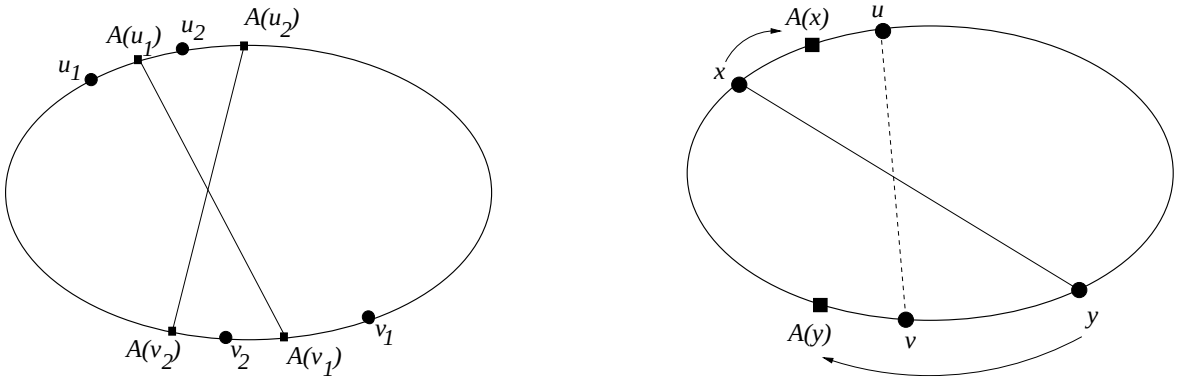

Figure 6: (a) If diagonals $(A(u_1), A(v_1))$ and $(A(u_2), A(v_2))$ intersect, then so do the diagonals (u_1, v_1) and (u_2, v_2) (b) The situation in which u and v do not belong to the same polygon in $\mathcal{L}$, but do belong to the same polygon in $\mathcal{L}_S$.

in the partition of P induced by $\mathcal{L}$, then the distance between u and v is at most α. So we assume that u and v do not belong to the same polygon in the polygon partition of P induced by $\mathcal{L}$. Let (x, y) be the diagonal in $\mathcal{L}$ that intersects (u, v) and is encountered first as we travel from u to v along the line segment (u, v). Without loss of generality assume that the vertices u, v, x, and y are as shown in Figure 6b. Also since the diagonal $(A(x), A(y))$ in $\mathcal{L}_s$ does not intersect (u, v), both $A(x)$ and $A(y)$ are either to the left or to the right of (u, v). Without loss of generality assume that both $A(x)$ and $A(y)$ are to the left of (u, v). For y to be assigned to the point $A(y)$, it must be the case that segment of the boundary of P defined by starting at y and walking clockwise to $A(y)$ belongs to a single $\beta \times \beta$ cell. Hence, $distance(y, v) \leq \sqrt{2}\beta$. Also, because of our choice of the diagonal (x, y), we have that u and y belong to the same polygon in $\mathcal{L}$ and hence $distance(u, y) \leq \alpha$. By the triangle inequality, $distance(u, v) \leq (\alpha + \sqrt{2}\beta)$. □

Lemma 5 *From an $(\alpha+\sqrt{2}\beta)$-small partition of P_S of size s, we can obtain in $O(n)$ time an $(\alpha+3\sqrt{2}\beta)$-small partition of P of size no more than s.*

Proof: Each vertex v in P_S lies on some edge, say (a, b) of P. Furthermore, there is a $p \in \{a, b\}$ such that v lies on the boundary of a cell containing p. Assign v to p and denote by $A(v)$ the vertex to which v is assigned. Let $\mathcal{L}$ be an $(\alpha + \sqrt{2}\beta)$-small partition of P_S given as a collection of diagonals of P_S. Replace each diagonal (u, v) in $\mathcal{L}$ by $(A(u), A(v))$, provided $A(u) \neq A(v)$. If $A(u) = A(v)$ then simply delete (u, v). By arguments very similar to those in the proof of Lemma 4, it is verified that no two diagonals in $\mathcal{L}$ intersect. These diagonals therefore define a partition of P and it is easily verified (using the triangle inequality) that each polygon in this partition has diameter at most $(\alpha + 3\sqrt{2}\beta)$. □

The implications of these lemmas are as follows. Let OPT denote the size of an optimal α-small partition of P. By Lemma 4, there is a $(\alpha + \sqrt{2}\beta)$-small partition of P_S of size at most OPT. Therefore, if we can compute an optimal $(\alpha + \sqrt{2}\beta)$-small partition D^* of P_S and this has size d^*, then $d^* \leq$ OPT. By Lemma 5, from D^* we can construct an $(\alpha + 3\sqrt{2}\beta)$-small partition of P of size no greater than OPT. Choosing $\beta = \varepsilon/3\sqrt{2}$ we get that D^* is an $(\alpha + \varepsilon)$-small partition of P. We now describe how to compute D^*.

Let $\delta = \alpha + 3\sqrt{2}\beta$. For any vertex i in P_S, how many vertices of P_S lie within a distance of at most δ from i? The disk of radius δ centered at i contains all such vertices and this disk is wholly contained in a $2\delta \times 2\delta$ square. This square intersects with at most $(1 + 2\delta/\beta)^2$ cells. Since each cell contains at most 2 vertices of P_S, there are at most $2(1 + 2\delta/\beta)^2 < 8(1 + \delta/\beta)^2$ vertices of P_S within a distance δ from i. For any i, these $O((1 + \delta/\beta)^2)$ can be located in $O((1 + \delta/\beta)^2)$ time by preprocessing the vertices of P_S into a data structure that answers such range queries in time proportional to the output returned.

In the dynamic programming algorithm in Section 2.2 we consider pairs of vertices (i, j) such that $i < j$ and i and j are visible from each other and compute optimal δ-small partitions of each polygon P_{ij}. Clearly, if the length of (i, j) exceeds δ, then P_{ij} has no δ-small partition. So, for each i, we need only consider j for which the length of (i, j) does not exceed δ, and from the preceding discussion we know that there are $O((1 + \delta/\beta)^2)$ such vertices j. Since i can be any one of n vertices we consider a total of $O(n(1 + \delta/\beta)^2)$ pairs (i, j), rather than m pairs.

In the original dynamic programming algorithm, for each pair (i, j), we consider out of the $O(n^2)$ circles of diameter δ passing through p and q, only those circles that contained both i and j. The $O(n^2)$ independent choices of p and q is what was responsible for the $O(n^2)$ possible choices of circles. But now there are only $O((1 + \delta/\beta)^4)$ choices of pairs p and q because both p and q have to be at a distance of δ or less from i. Furthermore, in the original algorithm, for each choice of a circle we constructed a weighted graph with $O(n)$ vertices and then computed a shortest path in this graph in $O(n)$ time. Now there are at most $O((1 + \delta/\beta)^2)$ vertices in the circle and therefore the weighted graph has $O((1 + \delta/\beta)^2)$ vertices and we can compute the required shortest path in $O((1 + \delta/\beta)^2)$ time. This yields a total running time of $O(n(1+\delta/\beta)^8)$. Since $\delta = \alpha + 3\sqrt{2}\beta$ and $\beta = \varepsilon/3\sqrt{2}$, the total running time of the dynamic programming algorithm is $O(n(1 + \alpha/\varepsilon)^8)$. The first step, in which we mapped points to their representatives and constructed P_S from P, takes $O(n)$ time and therefore we have the following theorem.

Theorem 6 *Let* OPT *be the size of an optimal α-small partition of a convex polygon P. For any real $\varepsilon > 0$, there is algorithm that computes an $(\alpha + \varepsilon)$-small partition of P of size no greater than* OPT *in time $O(n(1 + \alpha/\varepsilon)^8)$.*

5 Computing optimal α-small convex partitions

A partition is *convex* if all polygons in the partition are convex. In this section we show how to alter the algorithm presented in section 2.2, to produce an optimal α-small convex partition. The main idea there was to consider each subpolygon P_{ij} and identify the polygon Q that is incident on edge (i, j) in an optimal α-small partition of P_{ij}. Identifying Q translates into computing a shortest path $p_{ij}[C]$ from i to j in the graph $G_{ij}[C]$. Recall that $G_{ij}[C]$ is the visibility graph of the polygon P_{ij} restricted to vertices that lie within the circle C. In Section 2.2, we assigned to each edge (s, t), $s < t$, of $G_{ij}[C]$, not including edge (i, j), a weight w_{st} that equals the size of a minimum α-small partition of polygon P_{st}. Here we assign to each edge (s, t) a weight w_{st} that equals the size of a minimum *convex* α-small partition of polygon P_{st}. As before, the weight w_{ij} is initially ∞. The shortest path $p_{ij}[C]$ in $G_{ij}[C]$ is a candidate for the boundary of Q. Since we are now seeking a convex partition, Q needs to be convex and therefore we are interested in the shortest *convex* path from i to j in the graph $G_{ij}[C]$. The key difficulty is in the step in which a shortest path $p_{ij}[C]$ is computed by considering all vertices k visible from both i and j and assigning to $p_{ij}[C]$ the path $p_{ik}[C] \cup p_{k\ell}[C]$ that has least weight. The specific difficult being that even if the paths $p_{ik}[C]$ and $p_{k\ell}[C]$ are convex, the path $p_{ik}[C] \cup p_{k\ell}[C]$ need not be convex. In the following we describe how we get around this problem.

Having fixed vertices, i and j, $1 < i < j < n$ and a circle C the rest of our discussion in this section focuses on the graph $G_{ij}[C]$. So any reference to a vertex should be interpreted as a reference to a vertex in $G_{ij}[C]$ and when we say two vertices are adjacent or neighboring, we mean that they are adjacent in $G_{ij}[C]$.

Let $q_{ij}[C]$ denote a *convex* path from i to j with least weight. For any neighbor ℓ of j, define $L(\ell, j)$ as the directed line passing through ℓ and j and visiting ℓ first and then j. For any neighbor ℓ of j, let $q_{ij\ell}[C]$ denote a convex path from i to j of least weight, that lies completely to the left of $L(\ell, j)$. We assume in this definition that the set of points to the left of $L(\ell, j)$ include all points on $L(\ell, j)$ as well. These pieces of notation are illustrated in Figure 7.

We now show how to compute $q_{ij}[C]$. If j has no neighbor except for i, then there is no path from i to j of weight less than ∞ and and therefore $q_{ij}[C] = \perp$ and w_{ij} remains unchanged. Otherwise, let ℓ be the largest neighbor of j. Given the order in which vertices of the polygon P are numbered, ℓ is the last vertex

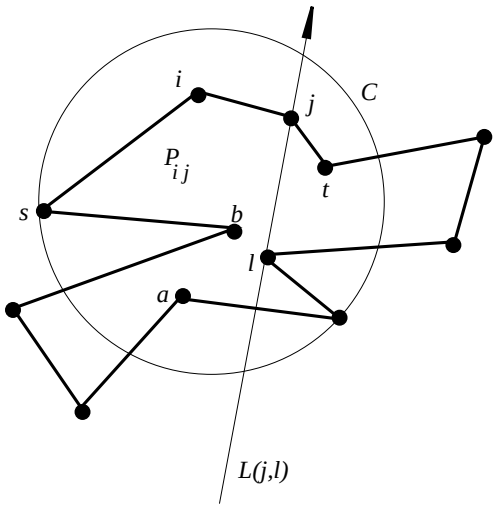

Figure 7: The polygon P_{ij} and the circle C are shown here. Vertex j has 5 neighbors in $G_{ij}[C]$; in counter clockwise order these are i, s, b, ℓ, t. There are 6 convex paths from i to j that lie to the left of $L(\ell, j)$: (i, j), (i, s, j), (i, s, b, j), (i, b, j), (i, b, ℓ, j), and (i, ℓ, j).

in the sequence of neighbors of j listed in counter clockwise order starting from i. Then $q_{ij}[C] = q_{ij\ell}[C]$ because any convex path from i to j has to lie completely to the left of $L(\ell, j)$. We now show how to compute $q_{ij\ell}[C]$ for an arbitrary neighbor $\ell \neq i$ of j. Note that there are two possibilities depending on whether $q_{ij\ell}[C]$ passes through ℓ or not. Let p_1 denote a convex path with least weight from i to j in $G_{ij}[C]$ that lies to the left of $L(\ell, j)$ *and* passes through ℓ. Let p_2 denote a convex path with least weight from i to j in $G_{ij}[C]$ that lies to the left of $L(\ell, j)$ *and* does not pass through ℓ. Then $q_{ij\ell}[C]$ is either p_1 or p_2, whichever has smaller weight. There are two candidates for p_1 and three candidates (excluding (i, j)) for p_2 in Figure 7.

Consider the sequence of vertices on the boundary of P_{ij}, in clockwise order starting from ℓ. Let vertex a be the first vertex in this sequence that is in C, to the left of $L(\ell, j)$, and is visible from ℓ. Vertex a is identified in Figure 7. Now observe that if $a \neq j$, then p_1 is simply $q_{i\ell a}[C] \cup \{j\}$. If $a = j$, then there is no vertex besides j on the left of $L(\ell, j)$ that is visible from ℓ. This implies that $p_1 = \perp$.

Consider the sequence of vertices on the boundary of P_{ij}, in clockwise order starting from ℓ. Let vertex b be the first vertex in this sequence that is in C, to the left of $L(\ell, j)$, and is visible from j. It is easy to observe that if $b \neq i$, then p_2 is $q_{ijb}[C]$. If $b = i$ then there is no path of finite weight from i to j that is to the left of $L(\ell, j)$ that does not pass through ℓ. So in this case $p_2 = \perp$.

In this manner we compute $q_{ij\ell}[C]$ for each neighbor ℓ of j, $\ell \neq j$, and then compute $q_{ij}[C]$. The weight w_{ij} assigned to edge (i, j) is set to the weight of $q_{ij}[C]$, if $q_{ij}[C] \neq \perp$. Otherwise, it remains unchanged at ∞, indicating that there is no α-small convex decomposition of P_{ij} that is possible.

Theorem 7 *A convex path $q_{ij}[C]$ of least weight from i to j in $G_{ij}[C]$ can be computed in $O(n^2)$ time, where n is the number of vertices in P.*

Proof: To compute $q_{ij}[C]$, we need to compute $q_{ij\ell}[C]$ for all neighbors ℓ of j, with the exception of i. For a given neighbor ℓ, computing $q_{ij\ell}[C]$ takes $O(1)$ time, given the values of $q_{i\ell a}[C]$ and $q_{ijb}[C]$. However, finding a and b takes $O(n)$ time in the worst case. Since j may have $O(n)$ time in the worst case, the total running time to compute $qij[C]$ is $O(n^2)$. □

Corollary 8 *There is an algorithm that computes a minimum size α-small convex partition of P in $O(m(n-r)^2 n^2)$ time.*

6 Open Problems

We have presented the first polynomial time algorithm for the minimum α-small partition problem for simple polygons without holes. We considered the version of the problem that disallows Steiner points. We also provide faster approximation algorithms for this problem. It may be possible to obtain a faster exact algorithm using more sophisticated techniques for pruning the dynamic programming search space. It may also be possible to improve the running time of our 2-approximation algorithm.

There are many related open problems, the most interesting of which is the complexity of the minimum α-small *covering* problem for simple polygons without holes. It is possible to construct examples of simple polygons without holes that can be covered by a very small number of α-small subpolygons, but any α-small partition contains a large number of subpolygons. An example simple polygon is shown in Figure 8 for which there is an $\alpha > 0$ such that an optimal α-small cover has size 2 and an optimal α-small partition has size $\Omega(n)$, where n is the number of vertices in the polygon. Figure 8(a) shows a simple polygon that is symmetric with respect to the x-axis that passes through the midpoint of edge (u_1, u_1'). A vertex u_i or u_i' with odd i is only visible to the 6 vertices on the boundary of the rectangle defined by $u_i, u_{i+1}u_{i+1}'u_i'$. A vertex u_i or u_i' with even i is only visible from vertices on the boundary of the two rectangles, the "left rectangle" $u_{i-1}u_iu_i'u_{i-1}'$ and the "right" rectangle $u_{i+1}u_{i+2}u_{i+2}'u_{i+1}'$. Now modify this polygon by inserting vertices labeled w_j and w_j' in Figure 8(b) These vertices have the property of being far enough from each other. More precisely, we have that $|w_{2i-1}w_{2j-1}'| > \alpha$ for any $1 \leq i, j \leq n$. All other diagonals have length no greater than α. An α-small covering of this polygon contains only two polygons: one is the shaded polygon Q_1 in Figure 8b; the other polygon is symmetric to Q_1 with respect to the x-axis. However, it can be easily seen that an α-small partition has size at least $n + 1$.

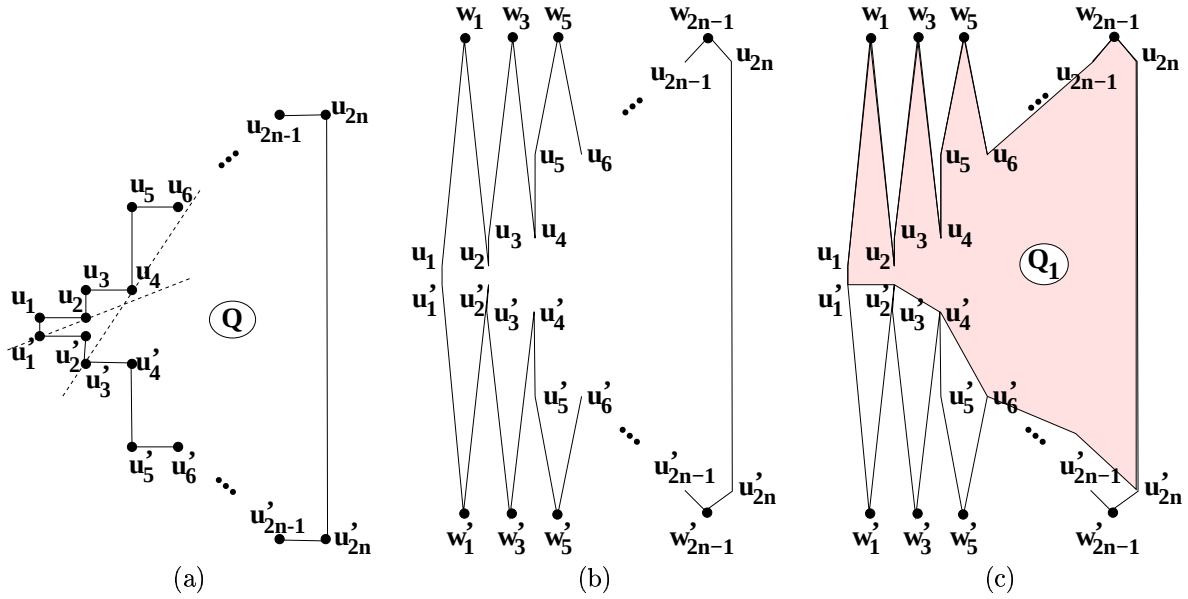

Figure 8: (a) The initial polygon, (b) the modified polygon obtained by adding vertices w_i and w_i', and (c) a minimum α-small covering of this polygon has size two: Q_1 and its symmetric polygon; a minimum α-small partition has size at least $n + 1$.

In the collision detection applications mentioned in the beginning of the paper, it does not seem to matter if the subpolygons overlap. Therefore, given how much smaller the size of an optimal α-small cover can be relative to the size of any α-small partition, it makes a lot of sense to compute an α-small cover, if at all possible.

References

[1] M. Bern. Triangulations. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 413–428. CRC Press, Boca Raton, 1997.

[2] B. Chazelle and D. P. Dobkin. Optimal convex decompositions. *Computational Geometry, G. Toussaint*, 1985.

[3] J.D. Cohen, M.C. Lin, D. Manocha, and M.K. Ponamgi. I-collide: An interactive and exact collision detection system for large scale environments. *Proc. of ACM Interactive 3D Graphics Conference*, pages 189–196, 1995.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press and McGraw-Hill Book Co., 1990.

[5] M. Damian. Shape constrained polygon decomposition and graph domination problems. 2000. PhD thesis, University of Iowa, Iowa City, IA 52242.

[6] M. Damian and S. Pemmaraju. Computing optimal α-fat and α-small decompositions. *Proc. of the 12th Ann. ACM-SIAM Symp. on Discr. Algh.*, pages 338–340, 2001.

[7] D.H. Greene. The decomposition of polygons into convex parts. In F.P. Preparata, editor, *Computational Geometry, Advances in Computer Research*, pages 235–259. JAI Press, 1983.

[8] J. Hershberger. An optimal visibility graph algorithm for triangulated simple polygons. *Algorithmica*, 4:141–155, 1989.

[9] J. M. Keil. Decomposing a polygon into simpler components. *SIAM Journal on Computing*, 14:799–817, 1985.

[10] J. M. Keil. Polygon decomposition. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry.* Elsevier Science B.V. North-Holland, 1999.

[11] J. M. Keil and J. Snoeyink. On the time bound for convex decomposition of simple polygons. *Proc. of the 10th Canad. Conf. Comp. Geom.*, pages 338–340, 1998.

[12] A. Lingas. The power of nonrectilinear holes. In *Automata, Languages and Programming, Lect. Notes in Comp. Sci.*, volume 140, pages 369–383. Springer-Verlag, 1982.

[13] N. Megiddo. Linear time algorithm for linear programming in R^3 and related problems. *SIAM Journal on Computing*, 12(4):759–776, 1983.

[14] S. Suri, P. M. Hubbard, and J. F. Hughes. Collision detection in aspect and scale bounded polyhedra. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 127–136, 1998.

[15] C. Worman. Decomposing polygons into bounded diameter components. In *Proceedings of the 15th Canadian Conference on Computational Geometry*, pages 103–107, August 2003.

[16] Y. Zhou and S. Suri. Analysis of a bounding box heuristic for object intersection. In *Proceedings of the 10th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 830–839, 1999.