# Uniformly Monotone Partitioning of Polygons Revisited*

Hwi Kim† Jaegun Lee‡ Hee-Kap Ahn§

## Abstract

Partitioning a polygon into simple pieces is a fundamental problem in computational geometry with a long history. In this paper, we revisit the problem of partitioning a simple polygon $P$ with $n$ vertices (including $R$ reflex vertices) and no holes into a minimum number of uniformly monotone subpolygons using open line segments drawn inside $P$. We present an $O(nR\log n + R^5)$-time algorithm for the problem by adding diagonals between pairs of vertices of $P$. When Steiner points can be placed on the boundary of $P$ and the subdivision is formed by adding diagonals between pairs of vertices of $P$ including Steiner points, we present an $O(n + R^5)$-time algorithm. We present an $O(n + R^4)$-time algorithm when Steiner points can be placed anywhere in $P$. Our algorithms improve upon the previously best ones for polygons with a small number of reflex vertices relative to the total number of vertices. We also present simple and efficient 2-approximation algorithms.

## 1 Introduction

Partitioning a polygon into disjoint simple pieces, such as triangles, trapezoids, convex polygons, and star-shaped polygons, is an important and fundamental problem in computational geometry [2, 3, 7, 8, 11, 12]. A classic and typical example is the triangulation of a simple polygon in the plane [2].

A simple polygon is called *monotone with respect to a line $\ell$* if for any line $\ell'$ perpendicular to $\ell$ the intersection of the polygon with $\ell'$ is connected. The problem of partitioning a polygon into *monotone* subpolygons has been well studied [7, 9]. Many geometric algorithms run faster asymptotically for monotone polygons than for more general ones, and it is often straightforward to implement algorithms for monotone polygons [12].

In this paper, we study the problem of partitioning a simple polygon with no holes into a minimum number of uniformly monotone subpolygons using diagonals (open line segments lying in the interior of the polygon) between pairs of vertices of the polygon. A partition is *uniformly monotone with respect to a line $\ell$* if every subpolygon in the partition is monotone with respect to $\ell$. Among all uniformly monotone partitions, we wish to compute one that minimizes the number of subpolygons in the partition. Below we define the problem formally.

**Minimum Uniformly Monotone Partition:** Given a simple polygon $P$ with $n$ vertices (including $R$ reflex vertices) and no holes, find a pair $(\ell^*, \mathsf{P}^*)$ of a line $\ell^*$ and a uniformly monotone partition $\mathsf{P}^*$ of $P$ with respect to $\ell^*$ such that the number of subpolygons in $\mathsf{P}^*$ is the minimum among all pairs $(\ell, \mathsf{P})$ of lines $\ell$ and uniformly monotone partitions $\mathsf{P}$ with respect to $\ell$.

We call such a pair $(\ell^*, \mathsf{P}^*)$ a *minimum partition-pair*, and such a partition $\mathsf{P}^*$ a *minimum partition* of $P$. In the rest of the paper, we may simply refer to the minimum uniformly monotone partition problem as the minimum partition problem.

We also consider two variants of the minimum partition problem with additional vertices, called *Steiner points*. Steiner points are added as part of the partition to reduce further the number of subpolygons. There are two ways of adding Steiner points, either placing them only on the boundary of $P$ or placing them anywhere in $P$. See Figure 1.

### 1.1 Previous works

Lee and Preparata [9] gave an $O(n\log n)$-time plane sweep algorithm to partition a simple polygon with $n$ vertices into monotone pieces, but not necessarily into a minimum number of monotone pieces. Liu and Ntafos [10] studied the minimum partition problem and gave an $O(nR^2\log n + nR^3 + R^5)$-time algorithm for the problem without using Steiner points. Using Steiner points lying on the boundary of the polygon, they gave an $O(nR^3\log n + R^5)$-time algorithm to compute a minimum partition.

*This research was partly supported by the Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. 2017-0-00905, Software Star Lab (Optimal Data Structure and Algorithmic Applications in Dynamic Geometric Environment)) and (No. 2019-0-01906, Artificial Intelligence Graduate School Program(POSTECH)).

†Department of Computer Science and Engineering, Pohang University of Science and Technology, Pohang, Korea. hwikim@postech.ac.kr

‡Department of Computer Science and Engineering, Pohang University of Science and Technology, Pohang, Korea. jagunlee@postech.ac.kr

§Graduate School of Artificial Intelligence, Department of Computer Science and Engineering, Pohang University of Science and Technology, Pohang, Korea. heekap@postech.ac.kr

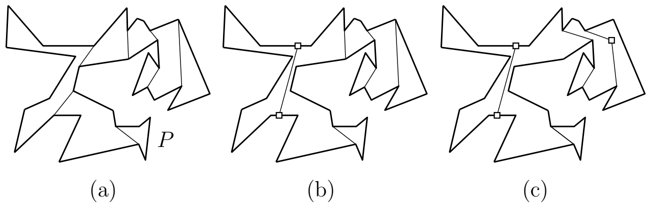

Figure 1: Uniformly monotone partitions of $P$ with respect to the $y$-axis. (a) A minimum partition (8 pieces) with no Steiner points. (b) A minimum partition (7 pieces) with two Steiner points (white squares) on the boundary of $P$. (c) A minimum partition (6 pieces) with three Steiner points (white squares) in $P$.

Wei et al. [13] considered the minimum partition problem for a polygon with $n$ vertices and $h$ holes. Using Steiner points lying in the polygon, they presented an $O(K(n \log n + h \log^3 h))$-time algorithm to compute a minimum partition of the polygon, where $K$ is the number of edges of the polygon's visibility graph.

For the problem of partitioning a polygon into a minimum number of subpolygons such that each subpolygon is monotone with respect to some line, Keil [6] gave an $O(n^4 R)$-time algorithm for polygons with $n$ vertices and no holes without using Steiner points. They also showed that the decision version of the problem is NP-complete for polygons with holes but without using Steiner points.

## 1.2 Our results

Our results are fourfold. First, we present an $O(nR \log n + R^5)$-time algorithm for the minimum partition problem with no Steiner points. The algorithm by Liu and Ntafos [10] is claimed to take $O(nR^2 \log n + nR^3 + R^5)$ time. But this may not hold for simple polygons with vertices of interior angle $\pi$ and the running time may increase to $O(n^2 + nR^3 + R^5)$. We give more details in Section 3.1. Our algorithm runs faster than their algorithms for $R = o(n^{1/2})$ while the running times are the same asymptotically for $R = \Omega(n^{1/2})$.

Second, we present an $O(n + R^5)$-time algorithm for the minimum partition problem when Steiner points can be placed on the boundary of $P$. Observe that the algorithm takes only $O(R^5)$ time in addition to the time linear to the input size. It runs faster than the $O(nR^3 \log n + R^5)$-time algorithm by Liu and Ntafos for $R = o(n^{1/2} \log^{1/2} n)$ while the running times are the same asymptotically for $R = \Omega(n^{1/2} \log^{1/2} n)$.

Third, we present an $O(n + R^4)$-time algorithm for the minimum partition problem when Steiner points can be placed anywhere in $P$. It takes only $O(R^4)$ time in addition to the time linear to the input size, and thus it runs fast for $R$ small relative to $n$. The algorithm by Wei et al. runs in $O(Kn \log n)$ time for this problem, but $K$ can be $\Theta(n^2)$.

Finally, we present simple factor-2 approximation algorithms for the minimum partition problem, that is, the number of subpolygons in the partition returned by our algorithm is at most twice the number of subpolygons in a minimum partition. We present an $O(n \log n)$-time algorithm with no Steiner points, and an $O(n + R \log n)$-time algorithm using Steiner points lying on the boundary of $P$. The solution returned by the latter algorithm is also a 2-approximation for the case that Steiner points can be placed anywhere in $P$.

**Sketches of our algorithms.** Our algorithms for the minimum partition problem are based on the work by Liu and Ntafos [10]. Given a line $\ell$, their algorithm first computes *peaks*, each of which is a vertex of $P$ and a source of local non-monotonicity with respect to $\ell$. Then the algorithm removes the peaks using a minimum number of non-intersecting diagonals between vertices and obtains a minimum partition of $P$ with respect to $\ell$ in $O(n \log n + nR + R^3)$ time.

To compute a minimum number of non-intersecting diagonals between vertices, they use a *circle graph* $G$. The vertices of $G$ correspond to the peaks in order along the boundary of $P$. There is a *chord* between two vertices of $G$ if and only if their corresponding peaks can be removed by adding the diagonal between them. Since each peak can be removed by a diagonal and each diagonal removes at most two peaks, a minimum partition of $P$ can be obtained by computing a *maximum independent chord set (MICS)* of $G$ in $O(m^3)$ time for $m$ vertices in $G$. By running this algorithm for each of $O(R^2)$ distinct lines defined by pairs of reflex vertices, Liu and Ntafos compute a minimum partition-pair of $P$ in $O(nR^2 \log n + nR^3 + R^5)$ time.

Our algorithms also construct circle graphs and compute their MICSs. Our algorithms are different to the ones by Liu and Ntafos in three aspects. The first difference is that our algorithms use a data structure for geodesic queries while the algorithms by Liu and Ntafos compute visibility polygons repeatedly for vertices and edges in peaks. From this, we reduce the time for computing diagonals.

The second difference is that our algorithm maintains and updates the vertices and the chords of the circle graph efficiently over $O(R^2)$ distinct lines. The algorithm by Liu and Ntafos constructs the circle graph from scratch for each of the lines. Our algorithm uses certain coherence between the circle graphs induced by two lines of consecutive orientations among the orientations defined by pairs of reflex vertices. Imagine that a line $\ell$ rotates. Then the circle graph $G$ induced by $\ell$ changes at certain orientations: a vertex of $G$ is removed, a new vertex is inserted to $G$, a chord of $G$ is removed, or a new chord is inserted to $G$. By keeping track of these changes in the order of orientations using a priority queue, our algorithm maintains and updates the circle graph and the minimum number of monotone subpolygons efficiently for $O(R^2)$ lines.

The third difference is that our algorithm computes MICSs without computing their corresponding partitions explicitly while the algorithm by Liu and Ntafos computes a partition of $P$ explicitly for each of $O(R^2)$ distinct lines. Whenever the circle graph is updated, our algorithm computes an MICS, but it does not compute the corresponding partition explicitly. Instead, it maintains and updates the minimum number of monotone subpolygons for the MICS. Once our algorithm is done with $O(R^2)$ lines, it computes a minimum partition $\mathsf{P}^*$ using the line corresponding to the minimum number of monotone subpolygons. This leads to further improvements on the time complexity.

When Steiner points are allowed to be placed anywhere in the polygon, the cardinality of an MICS of the circle graph equals the cardinality of the maximum matching of the circle graph. We show that the circle graph is a *chordal bipartite graph* under updates, and thus its maximum matching can be computed efficiently.

Our approximation algorithms achieve factor 2 by computing the minimum number of peaks of $P$ among all orientations. This is because the number of subpolygons in a minimum partition cannot be less than half of the number of peaks.

## 2 Preliminaries

We denote by $P$ the input simple polygon with $n$ vertices (including $R$ reflex vertices) and no holes. We assume that $P$ is given as a sequence of vertices in clockwise order along its boundary. For a compact set $X$, we use $\partial X$ to denote the boundary of $X$. For a point $p$, we denote by $y(p)$ the $y$-coordinate of $p$.

A *diagonal* is an open line segment that connects two vertices of $P$ and lies in the interior of $P$. A set of non-intersecting diagonals induces a *partition* of $P$ into subpolygons.

For a partition $\mathsf{P}$ of $P$, we denote by $|\mathsf{P}|$ the number of subpolygons in $\mathsf{P}$. A *minimum partition* of $P$ with respect to a given line $\ell$, called the *scan line*, is a uniformly monotone partition $\mathsf{P}^*_\ell$ with respect to $\ell$ such that $|\mathsf{P}^*_\ell| \leq |\mathsf{P}_\ell|$ for any uniformly monotone partition $\mathsf{P}_\ell$ of $P$ with respect to $\ell$. A minimum partition of $P$ is a uniformly monotone partition $\mathsf{P}^*$ of $P$ with respect to some scan line such that $|\mathsf{P}^*| \leq |\mathsf{P}|$ for any minimum partition $\mathsf{P}$ of $P$ with respect to some scan line.

For two points $p$ and $q$ in the plane, we denote by $pq$ the line segment connecting them. Two points $p, q \in P$ are *visible* to each other if $pq$ is contained in $P$. Two edges $e_1, e_2$ of $P$ are visible to each other if there are points $p \in e_1, q \in e_2$ such that $p$ and $q$ are visible to each other.

A *super-vertex* of $P$ is a maximal chain of consecutive and collinear vertices of $P$. A super-vertex is *reflex* if both its endpoints are reflex vertices.

For a fixed scan line $\ell$, a *peak* is a reflex vertex or a reflex super-vertex $v$ of $P$ such that both neighboring vertices of $v$ lie in one side of the line through $v$ and perpendicular to $\ell$. A peak with respect to $\ell$ is called a *normal-peak* if it is a reflex vertex of $P$, and a *super-peak* if it is a reflex super-vertex of $P$. The peaks are sources of local non-monotonicity with respect to $\ell$. See Figure 2(a).

A super-peak and a normal-peak are visible to each other if the super-peak has a vertex that is visible from the normal-peak. Two super-peaks are visible to each other if there are two vertices, one from each super-peak, that are visible to each other. See Figure 2(b).

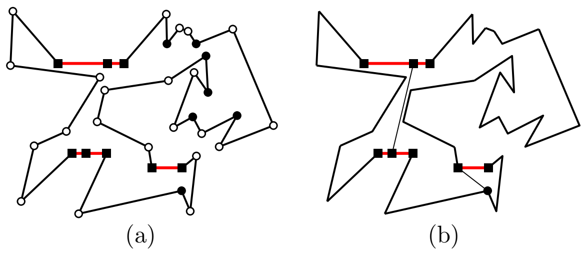

Figure 2: (a) The vertices of $P$ are shown as white disks, black disks, and black squares. Black disks are normal peaks and red chains (connecting black squares) are super-peaks with respect to the $y$-axis. (b) Two super-peaks are visible to each other, and a normal-peak and a super-peak are visible to each other.

Missing proofs and details can be found in the full version of the paper.

## 3 Maximum independent chord set of a circle graph

In this section, we describe the approach of using a *circle graph* given by Liu and Ntafos [10], and give an outline of their algorithm using the approach.

We fix the scan line to be the $y$-axis throughout this section unless stated otherwise. We call a normal-peak $v$ a *top normal-peak* if both neighboring vertices of $v$ lie above $v$, and a *bottom normal-peak* otherwise. Similarly, we call a super-peak a *top super-peak* if both neighboring vertices of the chain (super-peak) lie above it, and a *bottom super-peak* otherwise. A diagonal is *full* if it connects a top peak $u$ and a bottom peak $v$ with $y(u) \geq y(v)$. Thus, by adding a full diagonal, two peaks are removed simultaneously.

A minimum partition of $P$ with respect to a scan line can be obtained by adding a minimum number of non-intersecting diagonals that remove all peaks of $P$ with respect to the scan line. Observe that each peak can be removed by a diagonal and each diagonal removes at most two peaks. Thus, a minimum partition is obtained by finding a maximum number of full diagonals such that no peak is incident to more than one full diagonal. The algorithm by Liu and Ntafos computes a maximum number of full diagonals using a *circle graph* $G$. The vertices of $G$ correspond to the peaks in one-to-one mapping in order along $\partial P$. There is a *chord* between two vertices of $G$ if and only if there is a full diagonal between the peaks corresponding to the vertices. A minimum partition of $P$ can be obtained by computing an MICS of $G$.

**Lemma 1 ([10, 13])** *The number of subpolygons in a minimum partition of $P$ with respect to a fixed scan line is $N_p - N_m + 1$, where $N_p$ is the total number of peaks of $P$ with respect to the scan line and $N_m$ is the cardinality of an MICS of the circle graph induced by $P$ and the scan line.*

### 3.1 Minimum partition algorithms by Liu and Ntafos

Using Lemma 1, Liu and Ntafos [10] gave an $O(n \log n + nR + R^3)$-time algorithm that computes a minimum partition of $P$ with respect to a fixed scan line using no Steiner points. Their algorithm first reduces the minimum partition problem to the problem of computing an MICS of the circle graph $G$ induced by $P$ and the scan line. The algorithm finds all the top and bottom peaks as vertices of $G$. Then it computes the visibility polygon of each top peak using an $O(n)$-time algorithm for finding the visibility polygon of a point in a simple polygon with no holes. From the visibility polygon of a top peak $u$, it identifies all bottom peaks $v$ such that the open line segment connecting $u$ and $v$ is a full diagonal. Each such full diagonal contributes a chord to $G$. Thus, $G$ can be computed in $O(nR)$ time.

They gave an $O(m^3)$-time algorithm to compute an MICS of a circle graph with $m$ vertices. Since there are $O(R)$ vertices in $G$, the algorithm computes an MICS of $G$ in $O(R^3)$ time.

After the algorithm partitions $P$ into subpolygons using the set of full diagonals corresponding to the MICS, no subpolygon in the partition has a full diagonal. The algorithm removes all the remaining peaks in the partition in $O(n \log n)$ time using the subdivision algorithm by Lee and Preparata [9]. Thus, in total it takes $O(n \log n + nR + R^3)$ time.

Liu and Ntafos extend their algorithm to find a minimum partition-pair of $P$. There are $O(R^2)$ disjoint intervals of orientation, which are defined by pairs of reflex vertices, such that the set of peaks and the set of full diagonals remain the same for any orientation in an interval. With this observation, Liu and Ntafos claim that a minimum partition-pair of the input polygon can be computed in $O(nR^2 \log n + nR^3 + R^5)$ time, by running the algorithm for a fixed scan line for $O(R^2)$ directions, once for each orientation interval.

However, their algorithms may take more time than what they claim for simple polygons with vertices of interior angle $\pi$. Their algorithm for a fixed scan line works as follows. For each vertex $u$ in a super-peak, it computes the visibility polygon of $u$ and identifies the vertices $v$ such that there is a full diagonal connecting $u$ and $v$. Since there can be $\Theta(n)$ non-reflex vertices in $O(R)$ super-peaks, the time for computing the visibility polygons increases to $O(n^2)$. Then their algorithm takes $O(n^2 + R^3)$ time for a fixed scan line, and $O(n^2 + nR^3 + R^5)$ time for finding a minimum partition-pair of $P$.

When Steiner points are allowed to be placed on $\partial P$, there is a full diagonal $pq$ between a top super-peak $u$ and a bottom super-peak $v$ for the $y$-axis scan line and a point $p \in u$ and a point $q \in v$ if $y(p) \geq y(q)$, and $uv$ is a diagonal. See Figure 1(b). The algorithm by Liu and Ntafos considers each super-peak as an edge of $P$ and computes the visibility polygon of the edge. Using the visibility polygons, it computes pairs of super-peaks visible to each other and Steiner points lying on them together with full diagonals connecting them. The rest of the algorithm is the same with the algorithm for no Steiner points. Using an $O(n \log n)$-time algorithm for finding the visibility polygon of an edge in a simple polygon with no holes, their algorithm takes $O(nR \log n + R^3)$ time for a fixed scan line, and $O(nR^3 \log n + R^5)$ time for finding a minimum partition-pair.

## 4 Minimum uniformly monotone partition

We present our algorithms for the minimum partition problem. In Section 4.1, we describe our approach for maintaining the circle graph efficiently. In Sections 4.2-4.4, we present algorithms, one using no Steiner points, one using Steiner points lying on $\partial P$, and one using Steiner points lying anywhere in $P$.

### 4.1 Maintaining the circle graph

We show certain coherence between the circle graphs induced by two lines of consecutive orientations among the orientations defined by pairs of reflex vertices. We use this to compute a minimum partition of $P$ efficiently, provided that we have an algorithm, denoted by fd-algo, for computing a minimum partition of $P$ with respect to a fixed scan line.

Imagine that the scan line $\ell$ rotates around the origin in clockwise by $\pi$, starting from the $y$-axis. Let $\ell_\theta$ denote the scan line of orientation $\theta \in [0, \pi)$. The set of peaks and the set of full diagonals with respect to $\ell_\theta$ may change at certain discrete orientations $\theta$ at which one of the following occurs.

- *Diagonal event*: a diagonal connecting a top normal-peak and a bottom normal-peak becomes perpendicular to $\ell_\theta$.
- *Peak event*: an edge incident to a reflex vertex becomes perpendicular to $\ell_\theta$.

For super-vertices, it suffices to consider their peak events for capturing changes to the circle graph because a super-vertex induces no diagonal event. We construct a circle graph $G$ and maintain it for these events at discrete orientations. Initially, we construct $G$ as a complete graph with vertices corresponding to reflex vertices and reflex super-vertices of $P$ in order along $\partial P$. The circle graph induced by $P$ and $\ell_\theta$ is a subgraph of $G$. We mark each vertex and each chord of $G$ either as *used* or *unused*. At a diagonal event, we change the mark of the chord of $G$ corresponding to the diagonal accordingly. At a peak event, we change the mark of the vertex and the chords of $G$ corresponding to the peak accordingly.

Our algorithm works as follows. It constructs $G$ as above and mark the vertices and chords of $G$ such that the subgraph of $G$ induced by used vertices and chords is the circle graph induced by $P$ and the $y$-axis. It also computes the orientations of diagonal and peak events, and stores them in a priority queue with the orientations as keys. Then it processes the events one by one in order using the priority queue. For each event and its orientation $\theta$, it updates the number of peaks with respect to $\ell_\theta$ and updates the mark of the vertex or chord of $G$ corresponding to the event accordingly. Then it computes an MICS of the subgraph of $G$ induced by the used vertices and chords. The number of subpolygons in a minimum partition of $P$ with respect to $\ell_\theta$ is determined by the number of peaks and the cardinality of the MICS by Lemma 1. It updates the minimum number of subpolygons if the number with respect to $\ell_\theta$ is smaller than the minimum number we have so far.

After processing all the events, we have the line $\ell^*$ such that the minimum partition $\mathsf{P}^*$ of $P$ with respect to $\ell^*$ is a minimum partition of $P$. Finally, we compute a minimum partition $\mathsf{P}^*$ of $P$ with respect to $\ell^*$, and return $(\ell^*, \mathsf{P}^*)$ as a minimum partition-pair.

We analyze the time complexity of the algorithm. We denote by $T_c, T_m$ and $T_r$ the times for constructing the circle graph induced by $P$ and a fixed scan line, for computing an MICS of the circle graph, and for computing a uniformly monotone partition corresponding to the MICS, respectively.

**Lemma 2** *We can compute a minimum partition of $P$ in $O(n+R^2\log R+T_c+R^2T_m+T_r)$ time using $O(n+R^2)$ space.*

## 4.2 With no Steiner points

We present an $O(nR\log n + R^5)$-time algorithm for the minimum partition problem with no Steiner points, improving upon the result by Liu and Ntafos. Our algorithm uses a geodesic query data structure for finding full diagonals between super-peaks efficiently. Here, the *geodesic* between any two points $p, q \in P$, denoted by $\pi(p, q)$, is the unique shortest path between $p$ and $q$ that is contained in $P$.

**Lemma 3** *Given a top super-peak $H_1$ and a bottom super-peak $H_2$ of $P$ consisting of $n_1$ and $n_2$ vertices, respectively, we can check if there is a full diagonal between a vertex of $H_1$ and a vertex of $H_2$ in $O(\min\{n_1, n_2\}\log n)$ time, after an $O(n)$-time preprocessing using $O(n)$ space.*

By efficiently computing the chords corresponding to pairs of super-peaks as in Lemma 3, and using the result in Section 4.1, we have our result for the minimum partition problem with no Steiner points.

**Theorem 4** *Given a simple polygon $P$ with $n$ vertices (including $R$ reflex vertices) and no holes, we can compute a minimum partition-pair of $P$ in $O(nR\log n+R^5)$ time using $O(n+R^2)$ space, when no Steiner points are allowed.*

## 4.3 With boundary Steiner points

We present an $O(n + R^5)$-time algorithm for the minimum partition problem using Steiner points lying on the boundary of $P$, improving upon the result by Liu and Ntafos. Our algorithm first constructs the circle graph efficiently using geodesic queries for vertices in peaks.

**Lemma 5** *We can compute full diagonals of $P$ for the $y$-axis scan line in $O(n + R^2\log n + R^3)$ time.*

After constructing the circle graph using Lemma 5, our algorithm computes an MICS of the graph. Then, it computes a minimum partition from the MICS efficiently using ray shooting queries, and uses the result in Section 4.1.

**Theorem 6** *Given a simple polygon $P$ with $n$ vertices (including $R$ reflex vertices) and no holes, we can compute a minimum partition-pair of $P$ in $O(n + R^5)$ time using $O(n+R^2)$ space, using Steiner points lying on the boundary of $P$.*

## 4.4 With boundary and interior Steiner points

We give a sketch of an $O(n+R^4)$-time algorithm for the minimum partition problem using Steiner points lying on the boundary and interior of $P$.

The number of subpolygons in a minimum partition of $P$ may be reduced if Steiner points are allowed to be placed in the boundary and interior of $P$. See Figure 1(b,c). In Figure 1(c), a $y$-monotone chain connects a top peak and a bottom peak that are not visible to each other.

We refer to a chain that connects a pair of vertices of $P$ including Steiner points on $\partial P$, and is contained in the interior of $P$, except at the end vertices, as a *diagonal chain* of $P$. For a scan line $\ell_\theta$, a diagonal chain is *full* if it connects a top peak $u$ and a bottom peak $v$ with $y_\theta(u) \geq y_\theta(v)$, and it is monotone with respect to $\ell_\theta$. Here, $y_\theta(p)$ for a peak $p$ is the $y$-coordinate of the point obtained by rotating $p$ around the origin in counterclockwise by $\theta$. We use full diagonal chains, instead of full diagonals, for constructing chords of a circle graph. Then Lemma 1 holds for the circle graph.

We can show that there is a full diagonal chain between a pair of top and bottom peaks for scan line $\ell_\theta$ if and only if there are a vertex $u$ in the top peak and a vertex $v$ in the bottom peak with $y_\theta(u) \geq y_\theta(v)$ such that the geodesic between $u$ and $v$ is monotone with respect to $\ell_\theta$. Thus, we can compute the chords of the circle graph $G$ by computing for each pair of peaks, the geodesic between two vertices, one from each peak, and checking if it is monotone with respect to $\ell_\theta$.

After constructing $G$, we compute an MICS of $G$. Lemma 2 holds by modifying the definition of the diagonal event as follows.

- *Diagonal event*: the geodesic between a vertex in a top peak and a vertex in a bottom peak becomes monotone with respect to the scan line $\ell_\theta$.

It is known that the cardinality of an MICS of $G$ equals the cardinality of a maximum matching of $G$ [13]. We can show that $G$ is a *chordal bipartite graph* for any fixed scan line, which is a bipartite graph such that every cycle $C$ of length at least 6 in the graph has an edge not in $C$ that connects two vertices in $C$. A maximum matching of a chordal bipartite graph can be computed efficiently [1, 14]. Thus, we can efficiently compute the cardinality of an MICS of $G$.

By computing the cardinalities of MICSs as above and using the result in Section 4.1, we compute the orientation $\theta^*$ corresponding to a minimum partition of $P$. Then, we explicitly compute an MICS of the circle graph induced by $P$ and the scan line of orientation $\theta^*$. We compute a partition of $P$ corresponding to the MICS as follows. We place Steiner points on line segments lying in $P$ and perpendicular to the scan line, each of which is incident to a reflex vertex of $P$. We assign each Steiner point to the corresponding chord in the MICS, and connect the assigned Steiner points for each chord in the MICS to obtain a full diagonal.

From each remaining peak in the partition induced by the full diagonals, we shoot a ray parallel to the scan line of orientation $\theta^*$, and add the line segment obtained from the ray that lies in the subpolygon containing the peak. The resulting partition is a minimum partition of $P$. In total, it takes $O(n + R^4)$ time using $O(n + R^2)$ space to compute a minimum partition-pair of $P$.

## 5 Approximation algorithms

We give simple factor-2 approximation algorithms for the minimum partition problem. Observe that the number of subpolygons in a minimum partition cannot be less than half of the number of peaks. Thus, any partition of $P$ with the scan line of the orientation, that minimizes the number of peaks among all orientations, induced by diagonals each removing at least one peak, is a 2-approximation for the minimum partition problem. We compute such a partition in $O(n \log n)$ time with no Steiner points, and in $O(n + R \log n)$ time using Steiner points lying on the boundary of $P$. The solution returned by the latter algorithm is also a 2-approximation for the case that Steiner points can be placed anywhere in $P$.

## References

[1] M.-S. Chang. Algorithms for maximum matching and minimum fill-in on chordal bipartite graphs. *7th International Symposium on Algorithms and Computation (ISAAC)*, 146-155, 1996.

[2] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6(3):485–524, 1991.

[3] B. Chazelle and D. P. Dobkin. Optimal convex decompositions. *Machine Intelligence and pattern recognition*, 2:63–133, 1985.

[4] B. Chazelle, H. Edelsbrunner, M. Grigni, L. Guibas, J. Hershberger, M. Sharir and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12(1):54–68, 1994.

[5] L. Guibas and J. Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39(2):126–152, 1989.

[6] J. M. Keil. Decomposing a polygon into simpler components. *SIAM Journal on Computing*, 14(4):799–817, 1985.

[7] J. M. Keil. Polygon Decomposition. In J.R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, chapter 11, pages 491–518. Elsevier, 2000.

[8] H. Kim, J. Lee and H.-K. Ahn. Rectangular Partitions of a Rectilinear Polygon. `arXiv:2111.01970 [cs.CG]`, 2021.

[9] D. T. Lee and F. P. Preparata. Location of a point in a planar subdivision and its applications. *SIAM Journal on Computing*, 6:594–606, 1977.

[10] R. Liu and S. Ntafos. On decomposing polygons into uniformly monotone parts. *Information Processing Letters*, 27:85–89, 1988.

[11] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer, 1985.

[12] M. van Kreveld, O. Schwarzkopf, M. de Berg and M. Overmars. *Computational Geometry: Algorithms and Applications.* Springer, 2000.

[13] X. Wei, A. Joneja and D. M. Mount. Optimal uniformly monotone partitioning of polygons with holes. *Computer-Aided Design*, 44(12):1235–1252, 2012.

[14] J. P. Spinrad. Doubly lexical ordering of dense 0–1 matrices. *Information Processing Letters*, 45(5):229–235, 1993.