Approximation Algorithms for Computing Partitions with Minimum Stabbing Number of Rectilinear and Simple Polygons

Mohammad Ali Abam[*]
Department of Computer Engineering, Sharif University of Technology
Tehran, Iran
abam@sharif.ir

Boris Aronov[†]
Department of Computer Science and Engineering, Polytechnic Institute of NYU
Brooklyn, USA
aronov@poly.edu

Mark de Berg
Department of Computing Science, TU Eindhoven
Eindhoven, the Netherlands
mdberg@win.tue.nl

Amirali Khosravi[‡]
Department of Computing Science, TU Eindhoven
Eindhoven, the Netherlands
a.khosravi.dehkordi@tue.nl

ABSTRACT

Let P be a rectilinear simple polygon. The stabbing number of a partition of P into rectangles is the maximum number of rectangles stabbed by any axis-parallel line segment inside P. We present a 3-approximation algorithm for the problem of finding a partition with minimum stabbing number. It is based on an algorithm that finds an optimal partition for histograms.

We also study Steiner triangulations of a simple (non-rectilinear) polygon P. Here the stabbing number is defined as the maximum number of triangles that can be stabbed by any line segment inside P. We give an $O(1)$-approximation algorithm for the problem of computing a Steiner triangulation with minimum stabbing number.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

[*]Work by Mohammad Ali Abam was done when he was employed by Technische Universität Dortmund.

[†]Work by Boris Aronov has been supported by grant No. 2006/194 from the U.S.-Israel Binational Science Foundation, by NSF Grant CCF-08-30691, and by NSA MSP Grant H98230-10-1-0210.

[‡]Work by Amirali Khosravi has been supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 612.000.631.

General Terms

Algorithms

Keywords

Computational geometry, Approximation algorithms, Stabbing number, Polygons, Rectangular decompositions, Steiner triangulations

1. INTRODUCTION

Background and problem statement.

Computing decompositions of simple polygons is one of the most fundamental problems in computational geometry. When the polygon at hand is arbitrary then one typically wants a decomposition into triangles, and when the polygon is rectilinear one wants a decomposition into rectangles. Sometimes any such decomposition will do; then one can just compute an arbitrary triangulation or, for rectilinear polygons, a vertical or horizontal decomposition. This can be done in linear time [2]. In other cases one would like the decomposition to have certain additional properties. The property we are interested in, is that the so-called stabbing number—see below for a definition—of the decomposition is low.

Let P be a simple polygon with n vertices, and consider a decomposition of P into triangles. The vertices of the triangles need not all be vertices of P: it is allowed to introduce additional vertices, on the boundary of P and/or in its interior. However, we require the decomposition to be *conforming*: two triangles can only intersect in a common vertex or in a common edge. In other words, no triangle edge can end in the interior of another triangle edge. Such a decomposition is called a *Steiner triangulation.* The triangle vertices that are not vertices of P are called Steiner vertices. Note that we allow Steiner vertices on the boundary of P. The *stabbing number* of a segment s with respect to a Steiner triangulation T is the number of triangles from T intersecting s, and the stabbing number of T is the maximum stabbing number of any segment s that lies in the interior of P. Having a Steiner triangulation with low stabbing number is useful for ray shooting in P: after locating the starting point of the ray in the triangulation, one can answer a ray-shooting query by walking through the triangulation until the boundary of P is reached. The time for the walk is bounded by the stabbing number of the triangulation. Hershberger and Suri [6] showed that any simple polygon has a Steiner triangulation with stabbing number $O(\log n)$. This bound is asymptotically tight in the worst case, because any Steiner triangulation of a convex polygon with n vertices has stabbing number $\Omega(\log n)$ [6].

The above concepts can also be studied for rectilinear polygons. We call a decomposition of a simple rectilinear polygon P into rectangles a *rectangular partition*. A rectangular partition need not be conforming: a rectangle edge can end in the interior of another rectangle edge. The stabbing number of a segment s with respect to a rectangular partition R is the number of rectangles intersected by s, and the *(rectilinear) stabbing number* of R is the maximum stabbing number of any axis-parallel segment s in the interior of P. De Berg and Van Kreveld [4] showed that any rectilinear polygon admits a rectangular partition with stabbing number $O(\log n)$. Again, this bound is asymptotically tight in the worst case: any rectangular partition of a staircase polygon with n vertices has stabbing number $\Omega(\log n)$ [4].

The algorithms of Hershberger and Suri [6] and of De Berg and Van Kreveld [4] guarantee partitions with stabbing number $O(\log n)$, which is tight in the worst case. However, some polygons admit a partition with a much smaller stabbing number than $\Theta(\log n)$. Indeed, it is easy to see that some simple polygons admit a Steiner triangulation with stabbing number $O(1)$ and, similarly, that some rectilinear polygons admit a rectangular partition with stabbing number $O(1)$. This leads to the topic of our paper: given a polygon P, we wish to compute an *optimal* partition of P, that is, a partition whose stabbing number is minimum over all such partitions of P.

Our results.

We present approximation algorithms for this problem, for Steiner triangulations of simple polygons and for rectangular partitions of rectilinear polygons (We remark that these problems are not known to be NP-complete.) Our main result is a 3-approximation algorithm for the rectilinear case. It is based on an algorithm that computes an optimal rectangular partition for histograms. We also give a linear-time $O(1)$-approximation algorithm for computing a Steiner triangulation of a simple polygon.

Related work.

Chazelle *et al.* [3] studied the stabbing number of convex decompositions of polytopes. Later, Tóth [10] proved that any subdivision of d-dimensional space for $d \geqslant 2$ into n convex cells has stabbing number $\Omega((\log n / \log\log n)^{1/(d-1)})$. He also showed that any partition of d-dimensional space ($d \geqslant 2$) into n axis-aligned boxes has rectilinear stabbing number $\Omega(\log^{1/(d-1)} n)$, and presented a partitioning scheme achieving this bound [9]. Considering triangulations of point sets, Agarwal, Aronov and Suri [1] proved that one can triangulate n points in $\mathbb{R}^3$ (using Steiner points) with stabbing number $O(\sqrt{n} \cdot \log n)$.

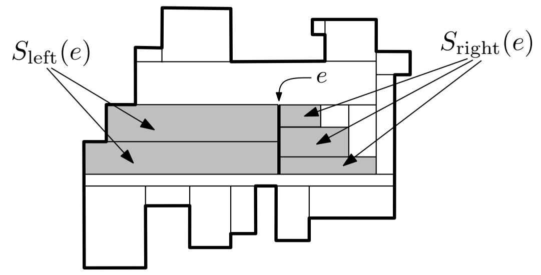


Figure 1: A maximal edge that is not anchored, and the set of rectangles bordering it.

2. RECTANGULAR PARTITIONS

Let P be a rectilinear simple polygon with n vertices. We denote the interior of P by $\text{int}(P)$ and its boundary by ∂P. In the remainder of this section, whenever we speak of partitions and stabbing numbers, we always mean rectangular partitions and rectilinear stabbing numbers. We denote the stabbing number of a partition R by $\sigma(R)$. The *horizontal stabbing number* of R, denoted $\sigma_{\text{hor}}(R)$, is defined as the maximum stabbing number of any horizontal segment $s \subset \text{int}(P)$. The *vertical stabbing number*, denoted $\sigma_{\text{vert}}(R)$, is defined similarly. Note that $\sigma(R) = \max(\sigma_{\text{hor}}(R), \sigma_{\text{vert}}(R))$.

2.1 The structure of optimal partitions

We start by proving some structural properties of optimal partitions. Consider a partition R of P. The partition is induced by a set $E(R)$ of *maximal edges*, that is, segments of maximal length that are a part of the union of one or more rectangle edges and are in the interior of ∂P. A maximal edge is *anchored* if at least one of its endpoints is a vertex of P. We first show that we can restrict our attention to anchored edges when computing optimal partitions.

LEMMA 1. *Any rectilinear simple polygon P has an optimal partition R_{opt} in which all maximal edges are anchored.*

PROOF. Over all optimal partitions of P, let R_{opt} be one with a minimum number of non-anchored edges and, among those partitions, one with a minimum number of rectangles. Suppose for a contradiction that R_{opt} has a maximal edge e that is not anchored, and assume without loss of generality that e is vertical. We denote the set of rectangles bordering e on the left by $S_{\text{left}}(e)$, and the set of rectangles bordering e on the right by $S_{\text{right}}(e)$; see Figure 1. Assume without loss of generality that $|S_{\text{left}}(e)| \leqslant |S_{\text{right}}(e)|$.

Now imagine moving e horizontally to the right until either (i) e hits a vertex of ∂P (and thus becomes anchored), or (ii) one of the rectangles in $S_{\text{right}}(e)$ disappears. Let R denote the resulting partition. We claim that $\sigma(R) \leqslant \sigma(R_{\text{opt}})$. Since R either has fewer non-anchored edges than R_{opt} or the same number of non-anchored edges and fewer rectangles, this contradicts the choice of R_{opt} and thus proves the lemma.

To prove the claim, we observe that the horizontal movement of e clearly does not increase the horizontal stabbing number. The vertical stabbing number does not increase either, since any maximal vertical segment still stabs the same set of rectangles or it stabs the rectangles from $S_{\text{left}}(e)$ instead of those from $S_{\text{right}}(e)$, and $|S_{\text{left}}(e)| \leqslant |S_{\text{right}}(e)|$. The claim follows. □

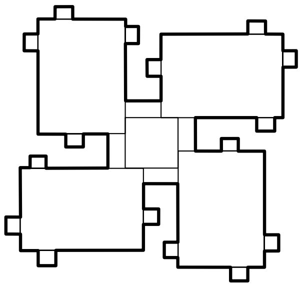


Figure 2: A rectilinear polygon for which the optimal rectangular partition is not a BSP**.**

A *binary space partition*, or BSP for short, of a rectilinear polygon P is a rectangular partitioning obtained by the following recursive process. First, P is cut into two subpolygons with an axis-parallel segment inside P, and then the two subpolygons are partitioned recursively in the same way. A BSP is *anchored* if each of its cuts is anchored. One may think that any rectilinear polygon admits an optimal partition that is an anchored BSP. Unfortunately this is not the case: Fig. 2 shows an example of a polygon that has a non-BSP partition with stabbing number 3 and for which any BSP has stabbing number at least 4. However, for so-called histograms we can show that there is always an optimal partition that is an anchored BSP. In fact, we show that any anchored partition of a histogram is a BSP. A *(vertical) histogram* is a rectilinear polygon H that has a horizontal edge seeing every point $q \in \text{int}(H)$. Here we say that an edge e *sees* a point $q \in P$ if there is an axis-parallel segment s connecting e to q that is completely lying inside $\text{int}(P)$ except possibly for its endpoints. We call the horizontal edge that sees all points in the histogram the *base* of the histogram and denote it by $\text{base}(H)$. A horizontal histogram is defined similarly: it has a vertical base that can see any point in the interior of the polygon.

LEMMA 2. *Any anchored partition of a histogram is a* BSP.

PROOF. Let R be an anchored partition of a histogram H. We will prove the statement by induction on n, the number of vertices of H. For $n = 4$ the statement is trivially true, so assume $n > 4$.

If $\text{base}(H)$ has only a single rectangle $r \in R$ adjacent to it, then $H \setminus r$ consists of one or more subhistograms, which are separated from r by horizontal BSP cuts. The partitionings inside these subhistograms induced by R are anchored and, hence, they are BSPs by induction. If there is more than one rectangle adjacent to $\text{base}(H)$, then there is a maximal edge ending in the interior of $\text{base}(H)$ with a vertex of P as its other endpoint. This is a BSP cut, which partitions H into two subhistograms. The partitions inside these subhistograms induced by R are BSPs by induction. ☐

2.2 A 3-approximation algorithm

Next we present our 3-approximation algorithm for the problem of finding an optimal rectangular partition of a given rectilinear polygon P. The algorithm constructs the partition in two steps. In the first step we split P into a set of histograms such that any axis-parallel segment inside P stabs at most three histograms. This can be done in $O(n)$ time [7]—see also [4]. In the second step we compute an optimal rectangular partition for each resulting histogram H. By proving that this can be done in polynomial time, we have the following theorem.

THEOREM 1. *Let P be a rectilinear simple polygon with n vertices. Then we can compute a rectangular partition of P with stabbing number at most $3 \cdot \text{OPT}$ in polynomial time, where* OPT *is the minimum stabbing number of any rectangular partition of P.*

PROOF. Consider a partition of P into a set of histograms using the above approach. Let us denote the set of histograms in the partition by $H = \{H_1, \ldots, H_i\}$. Let OPT_j denote the stabbing number of an optimal partition of a histogram $H_j \in H$. We claim that $\text{OPT}_j \leqslant \text{OPT}$ for each $H_j \in H$. Let R_{opt} denote an optimal partition of P. A property of the partitioning into histograms [7, 4] is that it only adds segments that "completely cut through P", that is, segments both of whose endpoints are in ∂P. This implies that any rectangle $r \in R_{\text{opt}}$ is cut into rectangles by the partition (if it is cut at all)—no other shape arise. This means that the part of R_{opt} inside H_j is a valid rectangular partition of H_j. This implies $\text{OPT}_j \leqslant \text{OPT}$. Any axis-aligned segment $s \in \text{int}(P)$ intersects at most three histograms of H, and we have $\text{OPT}_j \leqslant \text{OPT}$ for each histogram. Thus, finding OPT_j in polynomial time for each H_j results in a partition of P with stabbing number at most $3 \cdot \text{OPT}$. ☐

2.3 An optimal algorithm for histograms

Let H be a histogram. With a slight abuse of notation, we use n to denote the number of vertices of H. We assume without loss of generality that H is a vertical histogram lying above its base. By Lemmas 1 and 2, the histogram H admits an optimal partition that is an anchored BSP. We will need the following additional properties.

LEMMA 3. *There is an optimal partition R_{opt} for H that is an anchored* BSP *and such that, for every rectangle $r \in R_{\text{opt}}$, we have*

(i) the bottom edge of r is contained in either the top edge of a single rectangle $r' \in R_{opt}$ or in $\text{base}(H)$, and

(ii) the top edge of r contains an edge of H.

PROOF. By Lemmas 1 and 2 there is an optimal partition R_{opt} that is an anchored BSP. We claim that R_{opt} already has property (i) and that we can convert it into a partition having property (ii) as well without increasing its stabbing number.

Let e_{bot} be the bottom edge of some rectangle r in R_{opt}. Because H is a histogram, e_{bot} is either contained in $\text{base}(H)$ or in the union of some top edges of other rectangles in the partition. If the former is the case then r has property (i), so assume e_{bot} is contained in the union of some top edges of other rectangles. If e_{bot} would overlap with more than a single top edge, then there would be a vertical edge e in the partition whose top endpoint lies on e_{bot}. But because H lies above its base, the maximal edge containing e cannot have a vertex of H as bottom endpoint, and e would not be anchored. Thus e_{bot} must be contained in the top edge of a single rectangle, proving that r has property (i).

Next we convert R_{opt} as follows. For each rectangle r that does not yet have property (ii) we push its top edge e_{top} upward until it hits ∂H; see Figure 3. After doing this, the

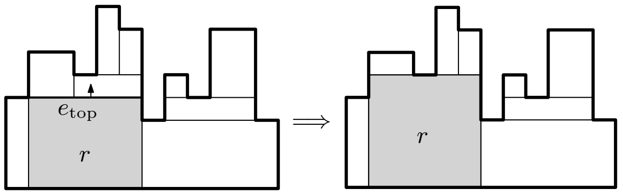


Figure 3: Illustration for the proof of Lemma 3

edge has property (ii). Because of property (i), there are no rectangles whose bottom edge overlaps only partially with e_{top} and so the partition remains a partition into rectangles. The stabbing number is not increased by this operation, so after pushing up all edges we have an optimal partition with properties (i) and (ii).

Note that the partition is still an anchored BSP after pushing all edges upward. Indeed, any vertical maximal edge is anchored at its top vertex, so it remains anchored (unless it completely disappears because of the pushing operations). Every horizontal maximal edge must also be anchored otherwise it would have been pushed up further. Thus the partition is anchored and by Lemma 2 we conclude that it is still a BSP. □

In the sequel we only consider partitions with properties (i) and (ii) from Lemma 3 (but not all partitions are anchored).

Reduction to a decision problem.

Our algorithm will do a binary search for the smallest value k such that H admits a partition with stabbing number k. Since there is always a partition with stabbing number at most $2\log_2 n$ [4], the binary search needs $O(\log\log n)$ steps. It remains to describe our decision algorithm, called HistogramPartition(H,k), which decides whether H has a partition with stabbing number at most k.

Canonical chords.

Define a *chord* of H to be a maximal horizontal segment contained in the interior of H except for its endpoints. A chord s partitions H into two parts. The part above s is a histogram, which we denote by $H(s)$. Note that any partition R of H induces a partition of $H(s)$; this partition is denoted by $R(s)$. Now consider a partition of H obtained by adding a chord from each vertex of H for which this is possible; this partition is sometimes called the horizontal decomposition of H. We call the resulting set of chords the *canonical chords* of H—see Figure 4(a). It will be convenient to also treat base(H) as a canonical chord.

The basic idea behind the algorithm is to treat the canonical chords from top to bottom. Now consider a canonical chord s_i with, say, two canonical chords s_j and s_ℓ immediately above it. Here we say that s_i is *immediately above* s_j, if we can connect s_i to s_j with a vertical segment that does not cross any other canonical chord. One may hope that if we have optimal partitions for $H(s_j)$ and $H(s_\ell)$ then we can somehow "extend" these to an optimal partition for $H(s_i)$. Unfortunately this is not the case, since an optimal partition need not be composed of optimal subpartitions. The next idea is to compute all possible partitions for $H(s_i)$. These can be obtained by considering all combinations of a possible partition for $H(s_j)$ and a possible partition for $H(s_\ell)$. Implementing this idea naively would lead to an exponential-time algorithm, however. Next we show how to compute a subset of all possible partitions that has only polynomial size and is still guaranteed to contain an optimal partition.

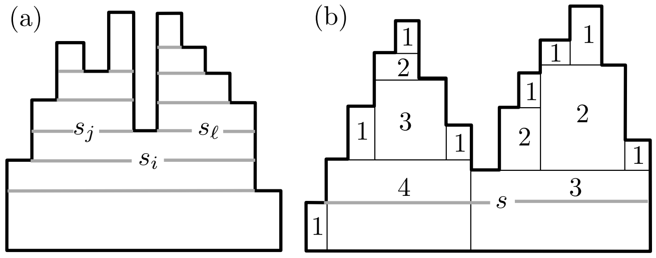


Figure 4: (a) Partitioning a histogram using canonical chords. (b) A partition with a unimodal labeling. The label sequence of the chord s is $4, 3$ and the label sequence of the base is $1, 4, 3$.

Labeled partitions and label sequences.

We first introduce some notation and terminology. Let R be any partition of H (satisfying the properties (i) and (ii) in Lemma 3). We say that a rectangle $r \in R$ is *on top of* a rectangle $r' \in R$ if the bottom edge of r is contained in the top edge of r'. When the bottom edge of r is contained in base(H) then we say that r is on top of the base. A *labeling* of R assigns a positive integer label $\lambda(r)$ to each rectangle $r \in R$. We say that a labeled partition is *valid* (with respect to the stabbing number k we are aiming for) if it satisfies the following conditions:

- if r is on top of r' then $\lambda(r) < \lambda(r')$;
- the vertical stabbing number of R equals the maximum label of any rectangle $r \in R$;
- the stabbing number of R is at most k.

Observe that the first two conditions together imply that the stabbing number of R is equal to the maximum label assigned to any rectangle on top of base(H). Also note that any partition with stabbing number k has a valid labeling: for example, one can set $\lambda(r)$ to be equal to the maximum number of rectangles that can be stabbed by a vertical segment whose lower endpoint lies inside r. We will use the labelings to decide which partitions can be ignored and which ones we need to keep.

Consider a chord s of H. We define the *label sequence of* s with respect to a labeled partition R as the sequence of labels of the rectangles crossed by s, ordered from left to right; here we say that s crosses a rectangle r if s intersects int(r) or the bottom edge of r. We denote this sequence by $\Sigma(s,R)$; see Figure 4(b) for an example. We say that a label sequence is *valid* if it consists of at most k labels and the maximum label is at most k. Note that a labeled partition is valid if and only if the label sequence of each of its canonical chords is valid. A label sequence $\lambda_1,\ldots,\lambda_t$ is called *unimodal* if there is an index i such that $\lambda_1 \leqslant \cdots \leqslant \lambda_i$ and $\lambda_i \geqslant \cdots \geqslant \lambda_t$. A labeling of a partition is unimodal if the label sequence of any chord is unimodal. A given label sequence can be made unimodal using the following simple procedure.

MakeUnimodal(Σ)
Let $\Sigma = \lambda_1,\ldots,\lambda_t$, and let λ_{i^*} be a maximum

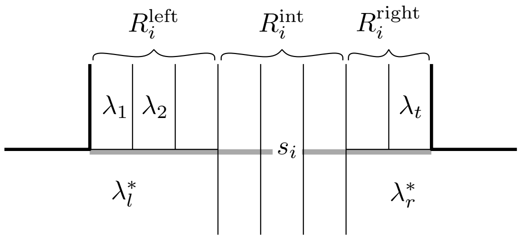


Figure 5: Illustration for the proof of Lemma 4.

label in the sequence. For each $i < i^*$ set $\lambda_i := \max_{j \leqslant i} \lambda_j$, and for each $i > i^*$ set $\lambda_i := \max_{j \geqslant i} \lambda_j$.

The next lemma states that we can make the label sequences of all canonical chords unimodal, and still keep a valid sequence.

LEMMA 4. *Any anchored partition of H of stabbing number at most k admits a valid unimodal labeling.*

PROOF. Let R be a partition of H. We first create a valid labeling for R as explained earlier: we set the label of a rectangle r to be equal to the maximum number of rectangles that can be stabbed by a vertical segment whose lower endpoint lies inside r. Next we turn this labeling into a valid unimodal labeling using the following process. Let $s_1, \ldots, s_m$ be the set of all canonical chords, sorted from bottom to top (that is, by increasing y-coordinates) and breaking ties arbitrarily. For each chord s_i in order, we apply MAKEUNIMODAL to the label sequence $\Sigma(s_i, R)$. We claim that this process satisfies the following invariant: after handling s_i we have that (i) the labeling is still valid, and (ii) the label sequence of any chord $s_{i'}$ with $i' < i$ is unimodal.

To prove that the invariant is maintained, consider a chord s_i. Let $r_1, \ldots, r_t$ be the rectangles ending on or properly intersecting s_i. We denote the current label of a rectangle r_ℓ by λ_ℓ, and the label after applying MAKEUNIMODAL to s_i by $\overline{\lambda}_\ell$. Let $\Sigma_i := \lambda_1, \ldots, \lambda_t$. Let R_i^{int} be the set of rectangles properly intersecting s_i. The rectangles in R_i^{int} are consecutive, because the partition is anchored. Let R_i^{left} and R_i^{right} be the sets of rectangles to the left and right of R_i^{int}, respectively—see Figure 5. Let Σ_i^{int} be the subsequence of Σ_i consisting of the labels of the rectangles in R_i^{int}. Note that Σ_i^{int} is already unimodal before s_i is handled, because these rectangles were handled in a previous step.

We claim that the labels in Σ_i^{int} are not modified when we make Σ_i unimodal. To prove this claim, note that if the label of some $r_\ell \in R_i^{\text{int}}$ has been changed, then there must be labels $\lambda_{\ell'}$ and $\lambda_{\ell''}$, with $\ell' < \ell < \ell''$, such that $\lambda_\ell < \min(\lambda_{\ell'}, \lambda_{\ell''})$. We cannot have $\lambda_{\ell'}, \lambda_{\ell''} \in \Sigma_i^{\text{int}}$, because that would mean that Σ_i^{int} was not unimodal before s_i was handled. Let λ_l^* and λ_r^* be the labels of the rectangles below the rectangles in R_i^{left} and R_i^{right}, respectively. It follows from the definition of labeling that for any rectangle $r_j \in R_i^{\text{left}}$ we have $\lambda_j < \lambda_l^*$. Similarly, for any rectangle $r_k \in R_i^{\text{right}}$ we have $\lambda_k < \lambda_r^*$. Since the labeling below s_i was unimodal before handling s_i, we have $\lambda_l^* \leqslant \lambda_\ell$ or $\lambda_r^* \leqslant \lambda_\ell$, or both. If both inequalities hold, then the claim holds since at least one of (the rectangles corresponding to) $\lambda_{\ell'}$ or $\lambda_{\ell''}$ belongs to one of R_i^{left} or R_i^{right} and is smaller than λ_ℓ. Otherwise, assume without loss of generality that $\lambda_l^* \leqslant \lambda_\ell$ and $\lambda_\ell \leqslant \lambda_r^*$. If $\lambda_{\ell'} \in R_i^{\text{left}}$ then since $\lambda_{\ell'} < \lambda_l^*$ we have $\lambda_{\ell'} < \lambda_l$ and the claim holds. When $\lambda_{\ell'}$ is not in R_i^{left}, it should be in R_i^{int}. Since $\lambda_\ell \leqslant \lambda_r^*$, and from the fact that the labeling below s_i before modification was unimodal, we can conclude that $\lambda_{\ell'} \leqslant \lambda_\ell$ and the claim holds.

Since the labels in Σ_i^{int} are not modified, we clearly have property (ii): the label sequence of any chord $s_{i'}$ with $i' < i$ is still unimodal. We also have property (i). Indeed, if a rectangle $r_\ell \in R_i^{\text{left}}$ gets a new label, then there is a label $\lambda_{\ell'} \in \Sigma_i^{\text{left}}$ to the left of it that is at least as large as the new label $\overline{\lambda}_\ell$. But this implies that the label of the rectangle below all rectangles in R_i^{left} is larger than this label. Hence, the modification of the label of r_ℓ does not make the labeling invalid. A similar argument shows that the modification of the labels of rectangles in R_i^{right} does not make the labeling invalid. □

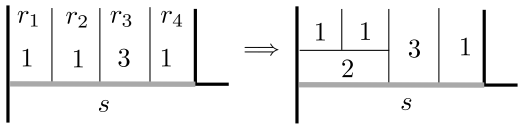


Figure 6: Merging two rectangles.

Dominated and non-dominated sequences.

Next we explain how the labelings help us decide which partitions can safely be discarded. Consider an algorithm that handles the chords from top to bottom, and suppose that the algorithm reaches a chord s. Let R_1 and R_2 be two labeled partitions of $H(s)$. Suppose that $\Sigma(s, R_1)$ is a subsequence of $\Sigma(s, R_2)$. Then there is no need to keep R_2: both partitions have stabbing number at most k so far, and it is easy to see that if we can complete R_2 to a partition with stabbing number k of the full histogram H then we can do so with R_1 as well. As another example in which we can ignore one of the two partitions, suppose $\Sigma(s, R_1) = 1, 1, 3, 1$ and $\Sigma(s, R_2) = 2, 3, 1$, and let $r_1, \ldots, r_4$ be the four rectangles in R_1 reaching the chord s. Then we could have *merged* r_1 and r_2 just before reaching s, that is, we could have terminated r_1 and r_2 and start a new rectangle with label "2"—see Figure 6. The new subsequence is then 2,3,1. This is a subsequence of $\Sigma(s, R_2)$—in fact, it happens to be equal to $\Sigma(s, R_2)$— so we can disregard R_2.

We now make this idea formal. We say that $\Sigma(s, R_1)$ *dominates* $\Sigma(s, R_2)$ if we can obtain a sequence $\Sigma(s, R_1')$ that is a subsequence of $\Sigma(s, R_2)$ by applying the following *merging operation* zero or more times to $\Sigma(s, R_1)$:

- Replace a subsequence $\lambda_i, \ldots, \lambda_j$ of $\Sigma(s, R_1)$ by the single label "$\max(\lambda_i, \ldots, \lambda_j) + 1$". Note that we can have $i = j$; in this case the operation just adds 1 to the label λ_i.

Intuitively, if a label sequence dominates another label sequence, then the first sequence has postponed some merging operations that we can still do later on. Thus there is no need to maintain partitions with dominated label sequences. (Note that postponing a merging operation implies that the resulting partition may not be anchored anymore. This is not a problem; it just means that in the algorithm presented below, we cannot restrict ourselves to anchored partitions.) The next lemma gives a bound on the number of label sequences that we need to maintain in the worst case.

Lemma 5. *Let $\mathcal{S}$ be any collection of valid unimodal sequences such that no sequence in $\mathcal{S}$ dominates any other sequence in $\mathcal{S}$. Then $|\mathcal{S}| = O(2^{3k/2}/\sqrt{k})$.*

Proof. For a unimodal sequence Σ, let $\text{inc}(\Sigma)$ denote the largest (not necessarily strictly) increasing subsequence of Σ. Since Σ is unimodal, $\text{inc}(\Sigma)$ is the prefix of Σ ending at the rightmost occurrence of the maximum value in the sequence. Similarly, let $\text{dec}(\Sigma)$ denote the largest decreasing subsequence of Σ. Define $\mathcal{S}_{\text{inc}} \subset \mathcal{S}$ to be the set of sequences for which $|\text{inc}(\Sigma)|$, the length of $\text{inc}(\Sigma)$, is at most $|\text{dec}(\Sigma)|$. We now bound the number of sequences in $\mathcal{S}_{\text{inc}}$; the other sequences (for which the reverse holds) can be counted in the same way.

Consider two different unimodal sequences Σ, Σ' such that $\text{inc}(\Sigma) = \text{inc}(\Sigma')$. We claim that either Σ dominates Σ' or vice versa. Indeed, traverse $\Sigma \setminus \text{inc}(\Sigma)$ and $\Sigma' \setminus \text{inc}(\Sigma')$ simultaneously until the first position where they differ. Suppose the label of Σ is smaller than the label of Σ' at this position (or Σ has ended). Then it is easy to see that Σ dominates Σ'. We conclude that $\mathcal{S}_{\text{inc}}$ does not contain two sequences Σ, Σ' with $\text{inc}(\Sigma) = \text{inc}(\Sigma')$. Hence, the number of label sequences in $\mathcal{S}_{\text{inc}}$ is bounded by the number of different (non-strictly) increasing subsequences in $\mathcal{S}_{\text{inc}}$.

We conclude that the number of label sequences in $\mathcal{S}_{\text{inc}}$ is bounded by the number of different (non-strictly) increasing sequences of length at most k and consisting of the integers $1, \ldots, k$. This is equivalent to the number of sequences of length exactly k and consisting of integers $0, \ldots, k$. This, in turn, is equivalent to the number of ways in which one can place k balls into $k+1$ labeled bins. Now we note that the maximum label of any two sequences in $\mathcal{S}_{\text{inc}}$ is the same—otherwise the sequence with the smaller maximum label would dominate the other sequence. Moreover, the number of times the maximum label, M, occurs can differ by at most one. Suppose that M occurs x or $x+1$ times in any sequence. We now only consider the sequences where M occurs x times; to obtain the final bound we just have to multiply by two. Then, in terms of the balls and bins metaphor, we only have to look at sequences that all put exactly the same number of balls into one of the bins. But that means we can ignore these balls (and this bin). Since we have $|\text{inc}(\Sigma)| \leqslant |\text{dec}(\Sigma)|$ for each of the subsequences in $\mathcal{S}_{\text{inc}}$, this means that we have to consider at most $\lfloor k/2 \rfloor$ balls. The number of ways in which one can put $\lfloor k/2 \rfloor$ into k bins is $\binom{\lfloor \frac{3k-3}{2} \rfloor}{k} = O(2^{3k/2}/\sqrt{k})$. ☐

The algorithm.

We can now describe our decision algorithm.

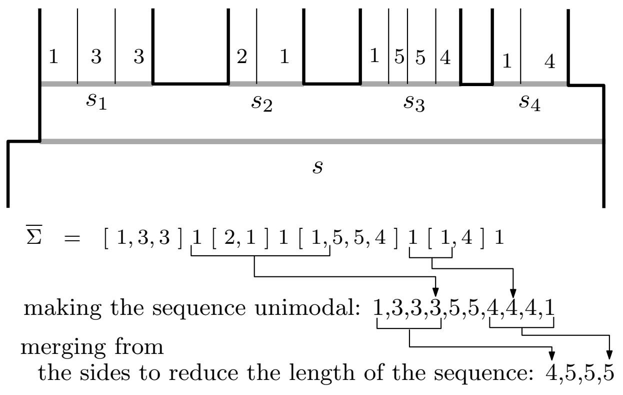
```
HistogramPartition($(H, k)$)
1. Compute the set of canonical chords of $H$ and sort the
   chords by decreasing $y$-coordinate.
2. For each chord $s$ in order, compute a collection $\mathcal{R}(s)$
   of labeled partitions of $H(s)$, as follows.
   (i) If $H(s)$ is a rectangle then set $\mathcal{R}(s) := \{H(s)\}$.
   (ii) Otherwise $s$ has one or more chords $s_1, \ldots, s_t$ immediately
        above it—see Figure 7. We compute all valid unimodal partitions
        of $H(s)$ that can be obtained from any combination of partitions
        in $\mathcal{R}(s_1), \ldots, \mathcal{R}(s_t)$ and whose label sequence
        is not dominated by the label sequence of any other such partition.
        (How this is done will be explained below.) Let $\mathcal{R}(s)$ be
        the set of all computed partitions. If $\mathcal{R}(s)$ is empty, then
        report that no partition with stabbing number $k$ exists for $H$,
        and exit.
3. Return any partition in $\mathcal{R}(\text{base}(H))$.
```

Next we explain how Step 2(ii) is performed. We assume that $t > 1$, that is, that s has several chords immediately above it; the case $t = 1$ can be handled in a similar (but much simpler) way. In the sequel, we identify each partition with its label sequence and only talk about label sequences. Note that the operations we perform on the label sequences can be easily converted into the corresponding operations on the partitions. For every pair of labeled partitions $R_1 \in \mathcal{R}(s_1), R_t \in \mathcal{R}(s_t)$ we proceed as follows.

(a) For each $1 < i < t$, consider the set $\mathcal{R}(s_i)$. Note that the label sequences in $\mathcal{R}(s_i)$ all have the same maximum value, M_i. This is true because a label sequence dominates any label sequence with larger maximum value. (The number of times the maximum label occurs can differ by at most one.) We pick an arbitrary label sequence $\Sigma_i \in \mathcal{R}(s_i)$ for which M_i occurs the minimum number of times.

(b) We now have, besides the partitions $\Sigma_1 \in \mathcal{R}(s_1)$ and $\Sigma_t \in \mathcal{R}(s_t)$, picked a partition Σ_i from each $\mathcal{R}(s_i)$ with $1 < i < t$. Let $\overline{\Sigma}$ be the label sequence obtained by concatenating the sequences Σ_i in order, inserting a label "1" for any horizontal histogram edge incident to a chord s_i, as illustrated in Figure 7. The labels "1" correspond to new rectangles that we can start, whose top edge is the given histogram edge. We then make $\overline{\Sigma}$ unimodal. This is done using a variant of the procedure MakeUnimodal explained earlier: the difference is that if we give several consecutive labels the same value, then we merge them into a single new label—see Figure 7.

(c) If the number of labels in $\overline{\Sigma}$ is at most k, then we put $\overline{\Sigma}$ into $\mathcal{R}(s)$. Otherwise $\overline{\Sigma}$ is invalid because it contains too many labels, and we have to merge some rectangles to obtain a shorter sequence. This is done as follows. Suppose that $\overline{\Sigma}$ contains $k+x$ labels $\lambda_1, \ldots, \lambda_{k+x}$. Then we have to get rid of x labels by merging. Let $x_{\text{left}}, x_{\text{right}}$ be integers such that $x_{\text{left}} + x_{\text{right}} = x + 2$ and both $x_{\text{left}}, x_{\text{right}}$ are non-zero, or $x_{\text{left}} + x_{\text{right}} =$



1
3
3
2
1
1
5
5
4
1
4
s_1
s_2
s_3
s_4
s
$\overline{\Sigma}$ = [1,3,3] 1 [2,1] 1 [1,5,5,4] 1 [1,4] 1
making the sequence unimodal: 1,3,3,3,5,5,4,4,4,1
merging from the sides to reduce the length of the sequence: 4,5,5,5

Figure 7: A chord s, the chords immediately above it, and the label sequence $\overline{\Sigma}$ defined by the label sequences of the chords.

$x+1$ and one of $x_{\text{left}}, x_{\text{right}}$ is zero. We merge x_{left} labels from the left into one new label, and x_{right} labels from the right into one, as in Figure 7. In other words, on the left side we replace $\lambda_1, \ldots, \lambda_{x_{\text{left}}}$ by a single new label $\lambda_{x_{\text{left}}}+1$ (and similarly on the right). If there are some labels immediately to the right of $\lambda_{x_{\text{left}}}$ with the same value as $\lambda_{x_{\text{left}}}$, then we include them into the merging process. (We can do this for free, since it reduces the number of labels, without increasing the value of the new label.) If this merging process yields a new label whose value is more than the previous maximum label value, then we simply merge the entire sequence into a single new label. If the value of this label is $k+1$, then we discard the sequence.

After having applied the above steps to every pair $R_1 \in \mathcal{R}(s_1), R_t \in \mathcal{R}(s_t)$, we remove from $\mathcal{R}(s)$ all partitions with a label sequence that is dominated by the sequence of some other partition. How this is done, is explained below. The next lemma shows the correctness of the decision algorithm.

Lemma 6. HistogramPartition(H,k) *returns a partition of H with stabbing number at most k if it exists.*

Proof. We will prove that the algorithm maintains the following invariant. Let R^* be an anchored partition of H of stabbing number at most k with a valid unimodal labeling.

Invariant: After handling the chord s, the set $\mathcal{R}(s)$ contains at least one label sequence Σ that dominates $\Sigma(s, R^*)$.

Let $\Sigma^* := \Sigma(s, R^*)$ and $\Sigma_i^* := \Sigma(s_i, R^*)$. Consider the handling of s in Step 2. If we are in case (i) then the Invariant obviously holds, so now suppose we are in case (ii). Let $s_1, \ldots, s_t$ be the chords immediately above s. By the invariant, each set $\mathcal{R}(s_i)$ contains a label sequence Σ_i dominating Σ_i^*. We argue that this implies that we can generate a label sequence Σ from $\Sigma_1, \ldots, \Sigma_t$ that dominates Σ^*, and that our algorithm actually finds such a label sequence.

The former statement is easy to see. By definition we can apply some merging operations to each Σ_i to turn it into a subsequence of Σ_i^*—in fact, we may also have to increase some labels, but this does not change the argument—and then we can simply "copy" the operations that turn $\Sigma_1^*, \ldots, \Sigma_t^*$ into Σ^*, thus turning $\Sigma_1, \ldots, \Sigma_t$ into a sequence dominating Σ^*. To argue that our algorithm actually finds a dominating sequence Σ—that is, that Step 2(ii) is implemented correctly—we have to argue a bit more carefully.

Suppose that in Step (b) we replace several labels by a single label λ. When these labels are between two other labels with value at least λ, we cannot avoid increasing their values to at least λ, and the best we can do is to replace all of them by a single label λ. Consider the sequence $\overline{\Sigma}$ made in step (b), that was generated from a pair Σ_1, Σ_t such that Σ_1 dominates Σ_1^* and Σ_t dominates Σ_t^*. Recall that for making $\overline{\Sigma}$ we picked, for $1 < i < t$ label sequences $\Sigma_i \in \mathcal{R}(s_i)$ for which the label with maximum value occurs the minimum number of times. Together with the fact that Σ_1 dominates Σ_1^* and Σ_t dominates Σ_t^*, and that, as just explained, we replace labels in an optimal way, this implies that $\overline{\Sigma}$ dominates Σ^*.

Now suppose that the number of labels in $\overline{\Sigma}$ is more than k. If the maximum label value in Σ^* is higher than the maximum label value in $\overline{\Sigma}$, then Step (c) will clearly result in a sequence dominating Σ^*. Otherwise the maximum label values in Σ^* and $\overline{\Sigma}$ are the same. Consider a sequence of replacement operations that turns $\overline{\Sigma}$ into a subsequence of Σ^*; such a sequence of operations exists because $\overline{\Sigma}$ dominates Σ^*. Let j be such that $\lambda_j \in \overline{\Sigma}$ has the maximum value. Let M_{left} (and M_{right}) be the maximum value of any label to the left (resp. right) of λ_j that is involved in a replacement operation. Then simply replacing all x_{left} labels to the left of λ_j whose value is at most M_{left} by a single label $M_{\text{left}}+1$, and replacing all x_{right} labels to the right of λ_j whose value is at most M_{right} by a single label $M_{\text{right}}+1$, leads to a sequence that dominates Σ^* and has at most k labels. Step (c) must consider some combination of merging x'_{left} labels from the left and x'_{right} labels from the right with $x'_{\text{left}} \leqslant x_{\text{left}}$ and $x_{\text{right}} \leqslant x_{\text{right}}$, and this will then result in a sequence dominating Σ^*. $\square$

The following lemma explains an approach for removing all partitions with a label sequence that is dominated by another partition in the set.

Lemma 7. *Let $\mathcal{R}$ denote a set of unimodal sequences, and let n be the total number of elements of the sequences in $\mathcal{R}$. Then we can remove all sequences from $\mathcal{R}$ that are dominated by another sequence in $\mathcal{R}$ in time $O(n \log n)$.*

Proof. Let M_i denote the maximum label in a sequence Σ_i, and let n_i be the number of times it occurs in Σ_i. Define $M_{\min} := \min\{M_i : \Sigma_i \in \mathcal{R}\}$ and $n_{\min} := \min\{n_i : \Sigma_i \in \mathcal{R}\}$. As mentioned above, the non-dominated sequences in $\mathcal{R}$ all have $M_i = M_{\min}$ and $n_i \leqslant n_{\min}+1$. Thus, first we remove all the sequences with $M_i > M_{\min}$ and the sequences with $n_i > n_{\min}+1$. This can be done in $O(n)$ time.

We partition the set of remaining sequences into two subsets: a subset $\mathcal{R}(n_{\min})$ containing all Σ_i with $n_i = n_{\min}$, and a subset $\mathcal{R}(n_{\min}+1)$ containing all Σ_i with $n_i = n_{\min}+1$. Note that a sequence in $\mathcal{R}(n_{\min}+1)$ can never dominate a sequence in $\mathcal{R}(n_{\min})$. Hence, our task is now to (i) remove all sequences from $\mathcal{R}(n_{\min})$ that are dominated by another sequence in $\mathcal{R}(n_{\min})$, and (ii) remove all sequences from $\mathcal{R}(n_{\min}+1)$ that are dominated by another sequence in $\mathcal{R}(n_{\min}+1)$, and (iii) remove all sequences from $\mathcal{R}(n_{\min}+1)$ that are dominated by a sequence in $\mathcal{R}(n_{\min})$.

Task (i) is performed as follows. Each sequence $\Sigma_i \in \mathcal{R}(n_{\min})$ is divided into two subsequences by removing all labels of maximum value from it. We denote the resulting sequences by Σ_i^1 and Σ_i^2—see Figure 8(a). Now consider two sequences $\Sigma_i, \Sigma_j \in \mathcal{R}(n_{\min})$. The crucial observation is

(a) $\Sigma_i = \langle 1,2,4,4,1,1 \rangle$ $\quad$ $\Sigma_j = \langle 1,3,4,4,2 \rangle$
$\Sigma_i^1 = \langle 1,2 \rangle$ $\quad$ $\Sigma_j^1 = \langle 1,3 \rangle$
$\Sigma_i^2 = \langle 1,1 \rangle$ $\quad$ $\Sigma_j^2 = \langle 2 \rangle$

(b) $\Sigma_i = \langle 1,2,4,4,2,1 \rangle$ $\quad$ $\Sigma_j = \langle 1,1,4,4,4,3 \rangle$
$\Sigma_i^1 = \langle 1,2 \rangle$ $\quad$ $\Sigma_j^{1,1} = \langle 1,1,4 \rangle$ $\quad$ $\Sigma_j^{2,1} = \langle 3 \rangle$
$\Sigma_i^2 = \langle 2,1 \rangle$ $\quad$ $\Sigma_j^{1,2} = \langle 1,1 \rangle$ $\quad$ $\Sigma_j^{2,2} = \langle 4,3 \rangle$

Figure 8: (a) Two sequences Σ_i and Σ_j in which $n_i = n_j$ and the four subsequences made from them. (b) Two sequences Σ_i and Σ_j in which $n_i = n_j + 1$ and the six subsequences made from them.

that Σ_i dominates Σ_j if and only if Σ_i^1 dominates Σ_j^1 and Σ_i^2 dominates Σ_j^2. Let $\mathcal{R}^1(n_{\min}) := \{\Sigma_i^1 : \Sigma_i \in \mathcal{R}(n_{\min})\}$ and $\mathcal{R}^2(n_{\min}) := \{\Sigma_i^2 : \Sigma_i \in \mathcal{R}(n_{\min})\}$, where we make sure all sequences in $\mathcal{R}^1(n_{\min})$ and $\mathcal{R}^1(n_{\min})$ have length exactly k by adding extra zeros; for $\mathcal{R}^1(n_{\min})$ these are added at the beginning of the sequence and for and $\mathcal{R}^1(n_{\min})$ they are added at the end. Note that all sequences in $\mathcal{R}^1(n_{\min})$ are non-decreasing, and all sequences in $\mathcal{R}^2(n_{\min})$ are non-increasing. For two sequences $\Sigma_i^1, \Sigma_j^1 \in \mathcal{R}^1(n_{\min})$ we write $\Sigma_i^1 \prec \Sigma_j^1$ if Σ_i^1 dominates Σ_j^1. Since the sequence in $\mathcal{R}^1(n_{\min})$ are non-decreasing this is equivalent to the following. Let $\Sigma_i^1 = \lambda_1, \dots, \lambda_k$ and $\Sigma_j^1 = \lambda'_1, \dots, \lambda'_k$. We have $\Sigma_i^1 \prec \Sigma_j^1$ if there is an index m such that $\lambda_m < \lambda'_m$ and for all $k < l \leqslant k$ we have $\lambda_l = \lambda'_l$. Note that $\prec$ defines a total order on $\mathcal{R}^1(n_{\min})$.

For two sequences $\Sigma_i^2, \Sigma_j^2 \in \mathcal{R}^2(n_{\min})$ we also write $\Sigma_i^2 \prec \Sigma_j^2$ if Σ_i^2 dominates Σ_j^2. Again, $\prec$ defines a total order on $\mathcal{R}^2(n_{\min})$. We now remove the dominated sequences from $\mathcal{R}(n_{\min})$ as follows.

Sort $\mathcal{R}^1(n_{\min})$ according to $\prec$, and sort $\mathcal{R}^2(n_{\min})$ according to $\prec$. Using RadixSort this can be done in $O(nk) = O(n \log n)$ time [5]. Now suppose that the first sequence in $\mathcal{R}^1(n_{\min})$ is Σ_i^1. Find the corresponding sequence Σ_i^2 in $\mathcal{R}^2(n_{\min})$; by maintaining cross-pointers this can be done in $O(1)$ time. Remove all the sequences Σ_j^2 from $\mathcal{R}^2(n_{\min})$ such that $\Sigma_i^2 \prec \Sigma_j^2$, and remove the corresponding sequences from Σ_j^1 from $\mathcal{R}^1(n_{\min})$. Since for the removed sequences we have $\Sigma_i^1 \prec \Sigma_j^1$ and $\Sigma_i^2 \prec \Sigma_j^2$, we also have that Σ_i dominates Σ_j. Hence, we remove Σ_j from $\mathcal{R}(n_{\min})$. Now remove Σ_i^1 and Σ_i^2 from $\mathcal{R}^1(n_{\min})$ and $\mathcal{R}^2(n_{\min})$, and repeat the process: take the next sequence from $\mathcal{R}^1(n_{\min})$, locate its corresponding sequence in in $\mathcal{R}^2(n_{\min})$, and remove the dominated sequences, and so on. The whole process, including the sorting, takes $O(n \log n)$ time, and it removes all dominated sequences from $\mathcal{R}(n_{\min})$.

Task (ii), removing all sequences from $\mathcal{R}(n_{\min}+1)$ that are dominated by another sequence in $\mathcal{R}(n_{\min}+1)$, can be done in the same way, so it remains to do task (iii). This is done as follows. Again, we divide each sequence $\Sigma_i \in \mathcal{R}(n_{\min})$ into two subsequences by removing all labels of maximum value from it. Now consider the sequences $\Sigma_j \in \mathcal{R}(n_{\min}+1)$. We also remove $n_{\min}$ labels of maximum value from them, thus dividing them into two. However, we can do this in two ways, either adding the remaining label of maximum value to the left sequence or two the right sequence. Thus we can obtain four different sequences, which we denote by $\Sigma_j^{1,1}$, $\Sigma_j^{1,2}$, $\Sigma_j^{2,1}$ and $\Sigma_j^{2,2}$—see Figure 8(b). We now have: $\Sigma_i \in \mathcal{R}(n_{\min})$ dominates $\Sigma_j \in \mathcal{R}(n_{\min}+1)$ if and only if either Σ_i^1 dominates $\Sigma_j^{1,1}$ and Σ_i^2 dominates $\Sigma_j^{2,1}$, or Σ_i^1 dominates $\Sigma_j^{1,2}$ and Σ_i^2 dominates $\Sigma_j^{2,2}$. Thus filtering out the sequences from $\mathcal{R}(n_{\min}+1)$ that are dominated by a sequence from $\mathcal{R}(n_{\min})$ can be done in $O(n \log n)$, using a similar strategy as before. □

The next lemma explains the running time of the algorithm. It follows from the above lemmas and the fact that $k \leqslant 2\log_2 n$.

Lemma 8. *Algorithm* HistogramPartition *runs in* $O(n^7 \log n \log\log n)$ *time.*

Proof. By Lemma 5 the number of sequences in each $R(s_i)$ is at most $O(2^{3k/2}/\sqrt{k})$. The fact that $k \leqslant 2\log_2 n$ then implies $|R(s_i)| = O(n^3/\sqrt{\log n})$. Thus there are $O(n^6/\log n)$ different pairs of $R_1 \in \mathcal{R}(s_1)$ and $R_t \in \mathcal{R}(s_t)$ in Step 2.

Step (a) takes time linear in the total length of all subsequences, which is $O(n^4/\sqrt{\log n})$, but we can re-use this result for each combination R_1, R_t. The concatenated sequence with which we start in Step (b) has length at most n. We need to do $O(k)$ different combinations of merging-from-the-left/merging-from-the-right as follows. Let us denote the number of times the maximum value occurs for the resulting sequence $\overline{\Sigma} = \{\lambda_1, \dots, \lambda_h\}$ by m. If $m \geqslant k$ we simply merge the entire sequence. Now suppose that $m < k$, and the labels $\lambda_i, \dots, \lambda_j$ all have the maximum value. We need to find two labels λ_s $(s < i)$ and λ_h $(h > j)$ such that $h - s + 1 = k$. When there is more than one label with the same value as λ_s pick the rightmost one. Similarly when there is more than one label with the same value as λ_h pick the leftmost one. Merge all the labels to the left of and including λ_s, and merge all the labels to the right of and including λ_h. Repeat this for $O(k)$ different possible values of h and s.

Thus the total running time needed for a single pair $R_1 \in \mathcal{R}(s_1)$ and $R_t \in \mathcal{R}(s_t)$ is $O(k) = O(\log n)$. Multiplying by the number of pairs, the time becomes $O(n^6)$.

We have now generated $O(n^6)$ sequences with length at most k, from which we have to select the non-dominated ones. Using the approach described in Lemma 7 we can find all dominating sequences in $O(n^6 \log n)$. This is the time that we spend for each chord s_i. We have at most n chords and we need to test $\log\log n$ different values for k. This makes the total running time to be $O(n^7 \log n \log\log n)$. □

3. APPROXIMATING OPTIMAL STEINER TRIANGULATIONS

In this section we give a $O(1)$-approximation algorithm for the problem of finding a Steiner triangulation with minimum stabbing number of a given n-vertex polygon.

We say that a point $p \in P$ *sees* a point $q \in P$ if the line segment pq lies completely inside P, and we say that a point p sees a segment s—or, equivalently, that the segment s sees p—if p sees at least one point of s. The *weak visibility polygon* of a segment s inside P, denoted by $\text{VP}(s)$, is the simple polygon consisting of all points of P that see s. The number of vertices of $\text{VP}(s)$ is denoted by $|\text{VP}(s)|$. Let $\text{OPT}(P)$ denote the minimum stabbing number of any Steiner triangulation of P, and let $V_{\max}$ be the maximum complexity of any weak visibility polygon $\text{VP}(s)$, where s is either an edge of P or a chord of P (that is, a segment whose endpoints lie on ∂P and that otherwise lies in $\text{int}(P)$). We have the following lemma.

Lemma 9. $\text{OPT}(P) \geqslant \log(V_{\max} - 2) - \log\log(V_{\max} - 2)$.

Proof. Let s be a segment such that $|\text{VP}(s)| = V_{\max}$ and let T_{opt} be an optimal Steiner triangulation of P. We may assume without loss of generality that s does not pass through vertices of triangles. We denote the set of triangles crossed by s from right to left by $\mathcal{D}(s) := \{\delta_1, \delta_2, \dots, \delta_k\}$; see Fig. 9(a). If $k > \log(V_{\max} - 2)$ we are done, so suppose that $k \leqslant \log(V_{\max} - 2)$. The proof is based on constructing a rooted binary tree $\mathcal{B}$ whose nodes correspond to edges of the triangles in T_{opt}. We show that $\mathcal{B}$ has at least $(V_{\max} - 2)/k$ leaves; the height of $\mathcal{B}$ is thus at least $\log((V_{\max} - 2)/k) = \log(V_{\max} - 2) - \log k \leqslant \log(V_{\max} - 2) - \log\log(V_{\max} - 2)$.

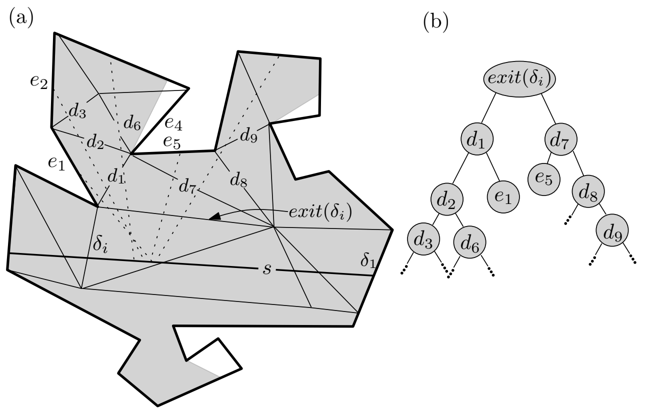


Figure 9: (a) The weak visibility polygon of s. (b) A part of the binary tree $\mathcal{B}$.

By construction, the height of $\mathcal{B}$ will be equal to the number of triangles stabbed by some segment in P, implying the claim of the lemma. The details are as follows.

For each edge e of the visibility polygon $\text{VP}(s)$, except for the two edges where s touches ∂P, we define a directed segment witness(e) inside P that connects s to e. Thus witness(e) is a witness of the fact that e is visible from s. Note that we have at least $V_{\max} - 2$ witness edges. We partition the set of witness edges into k subsets, as follows. For each triangle δ_i, let exit(δ_i) denote the edge of δ_i that does not intersect s. We put each witness $w :=$ witness(e) into a subset associated with the first exit edge that it intersects. The first exit edge is the first edge which we encounter, when we traverse w from s to e. Let W_i denote the set of witnesses associated with exit(δ_i) and let exit(δ_{i^*}) be the exit edge with the maximum number of witnesses associated with it. Note that $|W_{i^*}| \geqslant (V_{\max} - 2)/k$.

We now construct the tree $\mathcal{B}$, whose nodes will correspond to edges of triangles in T_{opt}, iteratively as follows. The root of $\mathcal{B}$ is the edge exit(δ_{i^*}). Consider the witnesses $w \in W_{i^*}$ in arbitrary order. For each $w :=$ witness(e), we expand the tree as follows. Traverse w from s to e, visiting the triangle edges intersected by w in order. At the same time, traverse the current tree $\mathcal{B}$ starting at the root in such a way that the node visited in $\mathcal{B}$ corresponds to the triangle edge being crossed by w. At some point this may no longer be possible: then we step from an edge e' to an edge e'' while the node ν of $\mathcal{B}$ that we are in (and which corresponds to e') does not have a child corresponding to e''. In this case we create a new child μ , which corresponds to e''—see Fig. 9(b). When we step from e'' to e''', we create a child for μ which corresponds to e''' and so on. Thus, we create a path hanging from ν that corresponds to the nodes intersected after e'. It is important to note that an edge can appear multiple times in $\mathcal{B}$. When we enter a triangle through a given edge, we can obviously only leave it through two other edges. (Here we use the assumption that we have a Steiner triangulation and not an arbitrary decomposition into triangles, that is, that there are no vertices in the interior of any triangle edge.) Hence $\mathcal{B}$ is a binary tree. Moreover, the path from the root of $\mathcal{B}$ to any leaf corresponds to the sequence of triangle edges intersected by some witness. Hence, the height of $\mathcal{B}$ is indeed a lower bound on the stabbing number of T_{opt}. □

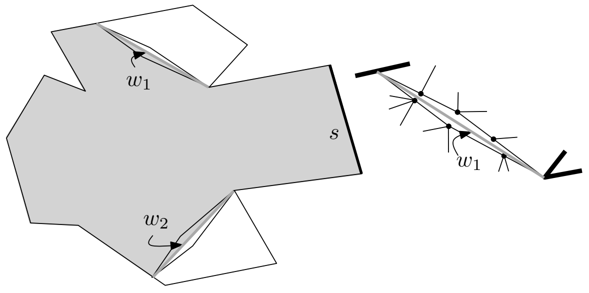


Figure 10: We need to replace each of the windows by a diamond like polygon, and then triangulate them separately

Next we give an algorithm that computes a Steiner triangulation for P with stabbing number $O(\log V_{\max})$. By Lemma 9 this is an $O(1)$-approximation for finding an optimal Steiner triangulation of a simple polygon P. The algorithm consists of two stages.

In the first stage we recursively compute a decomposition of P into weak visibility polygons, in a standard way as follows. In a generic step of the algorithm we get a subpolygon $P' \subset P$ and a designated edge e' of P'. Initially $P' = P$ and e' is an arbitrary edge of P. We then compute $\text{VP}(e')$. Note that $P' \setminus \text{VP}(e')$ consists of several subpolygons, each separated from $\text{VP}(e')$ by a chord which is called the *window* of the subpolygon. We recurse on each subpolygon with its window as designated edge. This weak-visibility-polygon decomposition can be found in $O(n)$ time in total [8]. Note that any line segment $s \subset P$ intersects at most three of the weak visibility polygons.

The method of Hershberger and Suri [6] makes a Steiner triangulation with stabbing number $O(\log n)$ for a polygon with n vertices. Using their method in the second stage we compute a Steiner triangulation with stabbing number $O(\log V_{\max})$ of each weak visibility polygon VP. This produces a decomposition of each weak visible polygon into triangles with stabbing number $O(\log V_{\max})$. Based on Lemma 9 we have $\text{OPT} = O(\log V_{\max})$, so this is a $O(1)$-factor approximation.

However, there is one problem: the decomposition is not necessarily a Steiner triangulation, because the Steiner triangulation of some polygon VP may introduce Steiner vertices on a window that are not used on the other side of the window. Hence, we adapt the algorithm as follows. Before applying the method of Hershberger and Suri to the visibility polygons, we first replace each window by a thin diamond, as shown in Figure 10. The decomposition of the visibility polygons can now add Steiner vertices on the edges of the diamonds, so the diamonds must be further decomposed.

Consider a diamond Δ, and let n_Δ be the number of Steiner vertices on the boundary of Δ. We move each vertex slightly outwards, so Δ becomes a convex polygon (with $n_\Delta + 4$ vertices) none of whose edges is collinear. Since we now have a convex polygon, we can easily compute a non-Steiner triangulation (in linear time) whose stabbing number is $O(\log n_\Delta)$. Note that any segment inside P can intersect at most three visibility polygons and at most two diamonds. Since the stabbing number of each visibility polygon is $O(\log V_{\max})$, each edge of the diamond has $O(\log V_{\max})$

vertices on it. This implies that $n_\Delta = O(\log V_{\max})$ for any diamond Δ. We get the following theorem.

THEOREM 2. *Let P be a simple polygon with n vertices. Then we can compute a Steiner triangulation of P with stabbing number $O(\text{OPT})$ in $O(n)$ time, where* OPT *is the minimum stabbing number of any Steiner triangulation of P.*

4. CONCLUDING REMARKS

We have studied the problem of finding a decomposition with minimum stabbing number for a simple polygon. We gave a 3-approximation algorithm for the rectilinear version of the problem (which was based on an optimal algorithm for histograms) and we gave an $O(1)$-approximation algorithm for the non-rectilinear case. We have not been able to come up with an exact polynomial-time algorithm, but the problems are not known to be NP-complete either. Establishing the computational complexity of the problem is thus the first open problem. Another interesting open problem is to study the case of polygons with holes.

5. REFERENCES

[1] P. K. Agarwal, B. Aronov, and S. Suri. Stabbing triangulations by lines in 3D. In *Proceeding 11th Symposium Computational Geometry*, pages 267–276, 1995.

[2] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete Computational Geometry*, 6(5):485–524, 1991.

[3] B. Chazelle, H. Edelsbrunner, and L. J. Guibas, The complexity of cutting complexes. *Discrete Computational Geometry*, 4(2):139–181, 1989.

[4] M. de Berg and M. van Kreveld. Rectilinear decompositions with low stabbing number. *Information Processing Letters*, 52(4):215–221, 1994.

[5] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein. *Introduction to algorithms, second edition*. The MIT Press, Cambridge, Massachusetts, 2001.

[6] J. Hershberger and S. Suri. A pedestrian approach to ray shooting: shoot a ray, take a walk. *Journal of Algorithms*, 18(3):403-431, 1995.

[7] C. Levcopoulos. *Heuristics for minimum decompositions of polygons*. Linkoping Studies in Science and Technology, Dissertations, No. 55, 1987.

[8] S. Suri. A linear time algorithm for minimum link paths inside a simple polygon. *Computer Vision, Graphics, and Image Processing*, 35(1):99–110, 1986.

[9] C.D. Tóth. Axis-aligned subdivisions with low stabbing numbers. *SIAM Journal of Discrete Mathematics* 22(3):1187–1204, 2008.

[10] C.D. Tóth. Stabbing numbers of convex subdivisions, *Periodica Mathematica Hungarica*, 57(1):217–225, 2008.