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@ce.sharif.edu

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
m.t.d.berg@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 present 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 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.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SOG ’11 Paris, France
Copyright 2011 ACM X-XXXXXX-XX-X/XX/XX ...$10.00.

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 decomposition. This can be done in linear time [3]. 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 [5] 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 \( \Omega(\log n) \).

The above concepts can also be studied for rectilinear polygons. We call a decomposition of a simple rectilinear polygon \( P \) into rectangles a \textit{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 \textit{(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 [2] 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 \( \Omega(\log n) \).

The algorithms of Hershberger and Suri [5] and of De Berg and Van Kreveld [2] 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 \textit{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 \textit{et al.} [4] studied the stabbing number of convex decompositions of polytopes. Later, Tóth [8] proved that any subdivision of \( d \)-dimensional space for \( d \geq 2 \) into \( n \) convex cells has stabbing number \( \Omega((\log n / \log \log n)^{d/4-1}) \). He also showed that any partition of \( d \)-dimensional space \( (d \geq 2) \) 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 \( \mathbb{R}^2 \) and \( \mathbb{R}^3 \) (using Steiner points) with stabbing number \( O(\sqrt{n} \cdot \log n) \).

2. OPTIMAL 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 \( \partial P \).

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 \( s \subset P \). 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 \textit{maximal} edges, that is, segments of maximal length that are the union of one or more rectangle edges that are not part of \( \partial P \). 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.

\textbf{Lemma 1.} Any rectilinear simple polygon \( P \) has an optimal partition \( R_{\text{opt}} \) in which all maximal edges are anchored.

\textbf{Proof.} Over all optimal partitions of \( P \), let \( R_{\text{opt}} \) 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 \( R_{\text{opt}} \) 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 \( |S_{\text{left}}(e)| \leq |S_{\text{right}}(e)| \).

Now imagine moving \( e \) horizontally to the right until either (i) \( e \) hits a vertex of \( \partial P \) (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 \( \sigma(R) \leq \sigma(R_{\text{opt}}) \). Since \( R \) either has fewer non-anchored edges than \( R_{\text{opt}} \) or the same number of non-anchored edges and fewer rectangles, this contradicts the choice of \( R_{\text{opt}} \) 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 stays the same set of rectangles or it stays the rectangles from \( S_{\text{left}}(e) \) instead of those from \( S_{\text{right}}(e) \), and \( |S_{\text{left}}(e)| \leq |S_{\text{right}}(e)| \). The claim follows. \( \square \)

}\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure1.png}
\caption{A maximal edge that is not anchored, and the set of rectangles bordering it.}
\end{figure}
A binary space partition, or BSP for short, of a rectilinear polygon $P$ is a rectangular partitioning obtained by the following recursive process. First, the polygon is cut into two subpolygons with an axis-parallel segment, 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 inside $\text{int}(P)$ except possibly 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 partitionings 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 [6]— see also [2]. 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 $\text{OPT}$ 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 $H = \{H_1, \ldots, H_j\}$. Let $\text{OPT}_j$ denote the stabbing number of optimal partition of a histogram $H_j \in H$. We claim that $\text{OPT}_j \leq \text{OPT}$ for each $H_j \in H$. Let us denote the optimal partition of $P$ by $\text{OPT}_P$. When we partition $P$ into a set of histograms as above, a rectangle $r$ in $\text{OPT}_P$ is cut by $\text{base}(H_j)$ or one of the so-called windows connected to $\text{base}(H_j)$. Thus, the part of $r$ in $H_j$ is still a rectangle. This means that the part of $\text{OPT}_P$, which is induced by $H_j$, is still a valid rectangular partition. This implies $\text{OPT}_j \leq \text{OPT}$. It is easy to see that any axis-aligned line in $P$ intersects at most three histogram of $H$. Since we have $\text{OPT}_j \leq \text{OPT}$ for each histogram, finding $\text{OPT}_j$ in polynomial time 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 $\text{OPT}_P$ for $H$ that is an anchored BSP and such that for each rectangle $r \in \text{OPT}_P$ we have

(i) the bottom edge of $r$ is contained in either the top edge of a single rectangle $r' \in \text{OPT}_P$ 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 $\text{OPT}_P$ that is an anchored BSP. We claim that $\text{OPT}_P$ already has property (i) and that we can convert it into a partition having property (ii) as well without changing its stabbing number.

Let $e_{\text{bot}}$ be the bottom edge of some rectangle $r \in \text{OPT}_P$. Because $H$ is a histogram, $e_{\text{bot}}$ 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 $e_{\text{bot}}$ is contained in the union of some top edges of other rectangles. If $e_{\text{bot}}$ 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 $e_{\text{bot}}$. 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 $e_{\text{bot}}$ must be contained in the top edge of a single rectangle, proving that $r$ has property (i).

Next we convert $\text{OPT}_P$ as follows. For each rectangle $r$ that does not yet have property (ii) we push its top edge $e_{\text{top}}$ upward until it hits $\partial H$; see Figure 3. After doing this, the edge has property (ii). Because of property (i), there are no
rectangles whose bottom edge overlaps only partially with $e_{\text{top}}$ 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 having properties (i) and (ii) from Lemma 3 (but not all partitions will be 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 $2 \log_2 n$ [2], 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 chords from top to bottom. Now consider a chord $s_i$ with, say, two chords $s_j$ and $s_t$ immediately above it. Here we say that a chord $s_i$ is immediately above another chord $s_j$, if we can connect $s_i$ to $s_j$ with a vertical segment that does not cross any other chord. One may hope that if we have optimal partitions for $H(s_j)$ and $H(s_t)$ 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 $H(s_t)$. 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.

Labeled partitions and label sequences.

We first introduce some notation and terminology. Let $R$ be any partition of $H$ (having 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 $\lambda_i \leq \cdots \leq \lambda_i$ and $\lambda_i \geq \cdots \geq \lambda_t$. 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.
Figure 5: Illustration for the proof of Lemma 4.

\textbf{MAKEUNIMODAL}(\Sigma)

Let \( \Sigma = \lambda_1, \ldots, \lambda_t \), and let \( \lambda_r \) be a maximum label in the sequence. For each \( i < i' \) set \( \lambda_i := \max_{j \leq i} \lambda_j \), and for each \( i > i' \) set \( \lambda_i := \max_{j \geq i} \lambda_j \).

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

\textbf{Lemma 4.} Any anchored partition of \( H \) of stabbing number at most \( k \) admits a valid unimodal labeling.

\textbf{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 \textbf{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 \( s_j \) with \( i < j \) 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 \( r \) by \( \lambda_r \), and the label after applying \textbf{MAKEUNIMODAL} to \( s_i \) by \( \lambda_r' \). Let \( \Sigma_i := \lambda_1, \ldots, \lambda_i \). Let \( R_{\text{left}}^i \) be the set of rectangles properly intersecting \( s_i \). The rectangles in \( R_{\text{left}}^i \) are consecutive, because the partition is anchored. Let \( R_{\text{int}}^i \) and \( R_{\text{right}}^i \) be the sets of rectangles to the left and right of \( R_{\text{left}}^i \), respectively—see Figure 5. Let \( \Sigma_{\text{int}}^i \) be the subsequence of \( \Sigma_i \) consisting of the labels of the rectangles in \( R_{\text{int}}^i \). Note that \( \Sigma_{\text{int}}^i \) is already unimodal before \( s_i \) is handled, because these rectangles were handled in a previous step.

We claim that the labels in \( \Sigma_{\text{int}}^i \) are not modified when we make \( \Sigma_i \) unimodal. To prove this claim, note that if the label of some rectangle \( r \in R_{\text{int}}^i \) has been changed, then there must be labels \( \lambda_r' \) and \( \lambda_r'' \), with \( \ell' < \ell < \ell'' \), such that \( \lambda_r' < \min(\lambda_r', \lambda_r'') \). We cannot have \( \lambda_r', \lambda_r'' \in \Sigma_{\text{int}}^i \), because that would mean that \( \Sigma_{\text{int}}^i \) was not unimodal before \( s_i \) was handled. Let \( \lambda_{r'} \) and \( \lambda_{r''} \) be the labels of the rectangles below the rectangles in \( R_{\text{left}}^i \) and \( R_{\text{right}}^i \). It follows from the definition of labeling that for any rectangle \( r \in R_{\text{left}}^i \) we have \( \lambda_r < \lambda_{r'} \). Similarly, for any rectangle \( r \in R_{\text{right}}^i \) we have \( \lambda_r < \lambda_{r''} \). Since the labeling below \( s_i \) was unimodal before handling \( s_i \), we have \( \lambda_r' < \lambda_r \) or \( \lambda_r < \lambda_r'' \), or both. If both inequalities hold, then the claim holds since at least one of \( \lambda_r' \) or \( \lambda_r'' \) belongs to one of \( R_{\text{left}}^{i-1} \) or \( R_{\text{right}}^{i-1} \) and is smaller than \( \lambda_r \). Otherwise, assume without loss of generality that \( \lambda_r' < \lambda_r < \lambda_r'' \).

\begin{align*}
\begin{array}{c|c|c|c|c}
1 & 2 & 3 & 4 & 5 \\
\hline
1 & 2 & 3 & 1 & 0 \Rightarrow 1 & 2 & 3 & 1 & 0
\end{array}
\end{align*}

Figure 6: Merging two rectangles.

\( \lambda_r' \leq \lambda_r \) and \( \lambda_r \leq \lambda_r'' \). If \( \lambda_r \in R_{\text{left}}^i \) then since \( \lambda_r < \lambda_r' \) we have \( \lambda_r < \lambda_r \) and the claim holds. When \( \lambda_r \) is not in \( R_{\text{left}}^i \), it should be in \( R_{\text{right}}^i \). Since \( \lambda_r \leq \lambda_r'' \), and from the fact that the labeling below \( s_i \) before modification was unimodal, we can conclude that \( \lambda_r \leq \lambda_r \) and the claim holds.

Since the labels in \( \Sigma_{\text{int}}^i \) are not modified, we clearly have property (ii): the label sequence of any chord \( s_j \) with \( i < j \) is still unimodal. We also have property (i). Indeed, if a rectangle \( r \in R_{\text{right}}^i \) gets a new label, then there is a label \( \lambda_r' \in \Sigma_{\text{int}}^{i-1} \) to the left of it that is at least as large as the new label \( \lambda_r'' \). But this implies that the label of the rectangle below all rectangles in \( R_{\text{left}}^{i-1} \) is larger than this label. Hence, the modification of the label of \( r \) does not make the labeling invalid. A similar argument shows that the modification of the labels of rectangles in \( R_{\text{right}}^{i-1} \) does not make the labeling invalid. \( \square \)

\textbf{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 \( \Sigma(s, R_3) = 1, 1, 3, 1 \) and \( \Sigma(s, R_2) = 2, 3, 1 \), and let \( r_1, \ldots, r_4 \) be the rectangles in \( R_1 \) reaching the chord \( s \). Then we could have \( r_1 \) and \( r_2 \) just before reaching \( s \), that is, we could have \( r_1 \) and \( r_2 \) just after modifying \( s \), such that \( r_1 \) and \( r_2 \) start a new rectangle with label \( "2" \)—see Figure 6. The new subsequence is then \( 2, 3, 1 \). This is a subsequence of \( \Sigma(s, R_2) \), 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 operation zero or more times to \( \Sigma(s, R_1) \):

- Replace a subsequence \( \lambda_1, \ldots, \lambda_j \) of \( \Sigma(s, R_1) \) by the single label \( \max(\lambda_1, \ldots, \lambda_j) + 1 \). Note that we can have \( i = j \); in this case the operation just adds 1 to the label \( \lambda_i \).

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 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 $\Sigma$, let $\text{inc}(\Sigma)$ denote the largest (not necessarily strictly) increasing subsequence of $\Sigma$. Since $\Sigma$ is unimodal, $\text{inc}(\Sigma)$ is the prefix of $\Sigma$ ending at the rightmost occurrence of the maximum value in the sequence. Similarly, let $\text{dec}(\Sigma)$ denote the largest decreasing subsequence of $\Sigma$. Define $\mathcal{S}_{\text{inc}} \subseteq \mathcal{S}$ to be the set of sequences for which $|\text{inc}(\Sigma)|$ is the length of $\text{inc}(\Sigma)$. Hence, the number of label sequences in $\mathcal{S}_{\text{inc}}$ is bounded by the number of different (non-strictly) increasing subsequences that we can have 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 subsequences of length at most $k$ and consisting of the integers $1, \ldots, k$. This is equivalent to the number of subsequences 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 $|\text{inc}(\Sigma)| \leq |\text{dec}(\Sigma)|$ 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 $(\begin{pmatrix}k/2 \cdot \lfloor k/2 \rfloor \end{pmatrix}) = O(2^{3k/2}/\sqrt{k})$. □

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 $\mathcal{R}(s) \equiv \{H(s)\}$. 
   (ii) Otherwise $s$ has one or more chords $s_1, \ldots, s_i$ 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 $\mathcal{R}(s_1), \ldots, \mathcal{R}(s_i)$ 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.

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 label partitions $R_t \in \mathcal{R}(s_1), R_t \in \mathcal{R}(s_i)$ 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 $\Sigma_i$ from each $\mathcal{R}(s_i)$ with $1 < i < t$. Let $\Sigma$ be the label sequence obtained by concatenating the sequences $\Sigma_i$ 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 $\Sigma$ 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 $\Sigma$ is at most $k$, then we put $\Sigma$ into $\mathcal{R}(s)$. Otherwise $\Sigma$ 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 $\Sigma$ 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 $x_{\text{left}} + x_{\text{right}} = x + 1$ and both $x_{\text{left}}, x_{\text{right}}$ are non-zero.
\[ x + 1 \text{ and one of } x_{\text{left}}, x_{\text{right}} \text{ is zero.} \] We merge \( x_{\text{left}} \) labels from the left into one new label, and \( x_{\text{right}} \) labels from the right into one, as in Figure 7. In other words, on the left side we replace \( \lambda_1, \ldots, \lambda_{\text{left}} \) by a single new label \( \lambda_{\text{left}} + 1 \) (and similarly on the right). If there are some labels immediately to the right of \( \lambda_{\text{right}} \), with the same value as \( \lambda_{\text{right}} \), 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 \( R_i \in \mathcal{R}(s_1), R_j \in \mathcal{R}(s_1) \), 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 by induction 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 \( \Sigma \) that dominates \( \Sigma(s, R') \).

Let \( \Sigma' := \Sigma(s, R') \) and \( \Sigma_1 := \Sigma(s, R') \). 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 \( s_1, \ldots, s_l \) be the chords immediately above \( s \). By the invariant, each set \( \mathcal{R}(s_i) \) contains a label sequence \( \Sigma_i \) dominating \( \Sigma_1 \). We argue that this implies that we can generate a label sequence \( \Sigma \) from \( \Sigma_1, \ldots, \Sigma_l \) that dominates \( \Sigma' \), 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 \( \Sigma_i \) to turn it into a subsequence of \( \Sigma_1 \)---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 \( \Sigma_1, \ldots, \Sigma_l \) into \( \Sigma' \), thus turning \( \Sigma_1, \ldots, \Sigma_l \) into a sequence dominating \( \Sigma' \). To argue that our algorithm actually finds a dominating sequence \( \Sigma \)—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 \( \lambda \). When these labels are in between two other labels with value at least \( \lambda \), we cannot avoid increasing their values to at least \( \lambda \), and the best we can do is to replace all of them by a single label \( \lambda \). Consider a sequence \( \Sigma \) made in step (b) for which we picked \( \Sigma_1 \) and \( \Sigma_2 \) dominating \( \Sigma_1 \) and \( \Sigma_2 \). Since for making \( \Sigma \) we picked arbitrary label sequences \( \Sigma_i \in \mathcal{R}(s_i) \) for which the label with maximum value occurs the minimum number of times, based on the above fact \( \Sigma \) dominates \( \Sigma' \). If the number of labels in \( \Sigma \) is at most \( k \) we are done.

Now consider the number of labels in \( \Sigma \) is more than \( k \). The number of labels in \( \Sigma' \) is at most \( k \) and \( \Sigma \) dominates \( \Sigma' \). Thus, there should be at least two subsequences \( \lambda_1, \ldots, \lambda_l \) in \( \Sigma \) and \( \lambda_1, \ldots, \lambda_m \) in \( \Sigma' \) having the maximum value labels, such that \( \lambda_i < \lambda_j \) and/or \( \lambda_j < \lambda_m \) and all the other labels are equal to each other one by one. Let us consider such a subsequence of \( \Sigma' \) with maximum length. The algorithm in Case (c) tries to merge the least possible number of labels from the sides and replace them with a new label larger than their maximum. It tries all the possibilities, and it is easy to see that merging "in the middle" is never better than merging from the sides. Thus, it finds a valid sequence which dominates \( \lambda_1, \ldots, \lambda_m \) and \( \Sigma' \). Thus the lemma holds.

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 us denote the maximum label in a sequence \( \Sigma_i \) by \( M_i \) and the number of times it occurs by \( n_i \). As mentioned above the sequences in \( \mathcal{R} \) should have the same value \( M_i \), and \( n_i \) should differ by at most one in them. Thus, first we remove all the sequences with maximum value greater than the minimum \( M_i \). Then, we remove the sequences for which \( n_i \) differs more than 1 from the minimum value of \( n_i \). The two steps can be done in \( O(n) \) time.

Let \( \Sigma_i \) and \( \Sigma_j \) be two subsequences whose maximum values are the same, and whose maximum values occur the same number of times. If we take the labels with maximum value from \( \Sigma_i \), it will be divided to two subsequences. Let us denote these subsequences by \( \Sigma_i(1) \) and \( \Sigma_i(2) \). Similarly we can make \( \Sigma_j(1) \) and \( \Sigma_j(2) \) from \( \Sigma_j \). Then \( \Sigma_i \) dominates \( \Sigma_j \) if and only if \( \Sigma_i(1) \) dominates \( \Sigma_j(1) \) and \( \Sigma_i(2) \) dominates \( \Sigma_j(2) \).

Similarly for the case in which the maximum value occurs in \( \Sigma_i \) once more than \( \Sigma_j \). By picking \( n_i \) labels with maximum value from \( \Sigma_i \) we get four different subsequences. Let us denote these four subsequences by \( \Sigma_i(1), \Sigma_i(2), \Sigma_i(3), \Sigma_i(4) \) and \( \Sigma_j(1), \Sigma_j(2), \Sigma_j(3), \Sigma_j(4) \). Since the maximum occurs in one of the sequences \( \Sigma_i(1), \Sigma_i(2), \Sigma_i(3), \Sigma_i(4) \) or one of the sequences \( \Sigma_j(1), \Sigma_j(2), \Sigma_j(3), \Sigma_j(4) \), \( \Sigma_i \) does not dominate \( \Sigma_j \). However, \( \Sigma_j \) dominates \( \Sigma_i \) if and only if \( \Sigma_i(1) \) dominates both \( \Sigma_i(1) \) and \( \Sigma_i(2) \) and \( \Sigma_i(2) \) dominates both \( \Sigma_i(2) \) and \( \Sigma_i(2) \).

We use the above facts to remove all the non-dominating sequences from \( \mathcal{R} \). Let us denote the minimum value of \( n_i \) for all sequences \( \Sigma_i \) in \( \mathcal{R} \) by \( m \). We call the subsequences made by picking \( m \) labels with maximum value from a sequence \( \Sigma_i \), the paired subsequences of \( \Sigma_i \). Note that the values of labels in one of the paired subsequences are strictly increasing and in another one are strictly decreasing. Now consider all \( O(n) \) subsequences made by picking \( m \) labels with maximum value from the sequences in \( \mathcal{R} \). Put all the increasing subsequences in a set, called \( \mathcal{R}(1) \), and all decreasing subsequences in another set, called \( \mathcal{R}(2) \). Then sort \( \mathcal{R}(1) \) and \( \mathcal{R}(2) \) such that a subsequence \( \Sigma_i \) dominates another subsequence \( \Sigma_j \) if and only if it comes before \( \Sigma_j \) in the order. For two decreasing subsequences \( \Sigma_i = \lambda_1, \ldots, \lambda_k \) and \( \Sigma_j = \lambda', 1, \ldots, \lambda_k' \) we put \( \Sigma_i \) before \( \Sigma_j \) in the order if there are two labels \( \lambda_l, \lambda_l' \) in \( \Sigma_i \) and \( \lambda_l', \lambda_l \) in \( \Sigma_j \) such that \( \lambda_l < \lambda_l' \) and for all \( l < k \) we have \( \lambda_l = \lambda_l' \). Similarly we can define orders between two increasing sequences.

Suppose that the first subsequence in \( \mathcal{R}(1) \) is \( \Sigma_1(1) \). Find its paired subsequence \( \Sigma(2) \) in \( \mathcal{R}(2) \). Remove all the subsequences \( \Sigma_1(1) \in \mathcal{R}(1) \) and \( \Sigma_1(2) \in \mathcal{R}(2) \) such that \( \Sigma_1(1) < \Sigma_1(2) \).
$\Sigma(1)$ and $\Sigma(2) \leq \Sigma(2)$ in the sorted orders. Repeat this for all the remaining subsequences in $R(1)$ one by one. Finally, keep all the sequences in $R$ which still have subsequences in $R(1)$ and $R(2)$ and remove all the other ones from it. The final set $R$ is the set of non-dominating sequences. Sorting of $R(1)$ and $R(2)$ takes $O(n \log n)$. Finding a paired subsequence can be done in constant time and we need to process $O(n)$ subsequences in $R(1)$.

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

**Lemma 8.** Algorithm HistogramPartition runs in $O(n^7 \log^2 n \log \log n)$ time.

**Proof.** By Lemma 5 the number of sequences in each $R(s_i)$ is at most $O(2^{8k/2}/\sqrt{k})$. The fact that $k \leq 2 \log_2 n$ then implies $|R(s_i)| = O(n^{k}/\sqrt{\log n})$. Thus there are $O(n^k/\log n)$ different pairs of $R_1 \in R(s_1)$ and $R_2 \in R(s_2)$ in Step 2.

Step (a) takes time linear in the total length of all subsequences, which is $O(n^k/\sqrt{\log n})$, but we can re-use this result for each combination $R_1, R_2$. 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 $\Sigma = \{\lambda_1, \ldots, \lambda_k\}$ by $m$. If $m > k$ we simply merge the entire sequence. Now suppose that $m < k$, and the labels $\lambda_1, \ldots, \lambda_n$ all have the maximum value. We need to find two labels $\lambda_i$ ($s < i$) and $\lambda_h$ ($h > j$) such that $h - s + 1 = k$. When there are more than one label with the same value as $\lambda_i$, pick the rightmost one. Similarly when there are more than one label with the same value as $\lambda_h$, pick the leftmost one. Merge all the labels to the left and including $\lambda_i$, and merge all the labels to the right and including $\lambda_h$. Repeat this for $O(k)$ different possible values of $h$ and $s$.

Thus the total running time needed for a single pair $R_1 \in R(s_1)$ and $R_2 \in R(s_2)$ is $O(k) = O(\log(n))$. Multiplying by the number of pairs, the time becomes $O(n^k)$. We have now generated $O(n^k)$ 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 $O(n^k \log^2 n)$.

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 $O(n^7 \log^2 n \log \log n)$. $\square$

### 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 $\mathcal{VP}(s)$, is the simple polygon consisting of all points of $P$ that see $s$. The number of vertices of $\mathcal{VP}(s)$ is denoted by $|\mathcal{VP}(s)|$. Let $Opt(P)$ denote the minimum stabbing number of any Steiner triangulation of $P$, and let $V_{\text{max}}$ be the maximum complexity of any weak visibility polygon $\mathcal{VP}(s)$, where $s$ is either an edge of $P$ or a chord of $P$ (that is, a segment whose endpoints lie on $\partial P$ and that otherwise lies in $\text{int}(P)$). We have the following lemma.

**Lemma 9.** $Opt(P) \geq \log(V_{\text{max}} - 2) - \log \log(V_{\text{max}} - 2)$

**Proof.** Let $s$ be a segment such that $|\mathcal{VP}(s)| = V_{\text{max}}$ and let $T_{\text{opt}}$ be an optimal Steiner triangulation of $P$. We may assume without loss of generality that $s$ is horizontal and it does not pass through vertices of triangles. We denote the set of triangles crossed by $s$ from right to left by $D(s) = \{\delta_1, \delta_2, \ldots, \delta_h\}$; see Fig. 8(a). If $k > \log(V_{\text{max}} - 2)$ we are done, so suppose that $k \leq \log(V_{\text{max}} - 2)$. The proof is based on constructing a rooted binary tree $B$ whose nodes correspond to edges of the triangles in $T_{\text{opt}}$. We show that $B$ has at least $(V_{\text{max}} - 2)/k$ leaves; the height of $B$ is thus at least $\log((V_{\text{max}} - 2)/k) = \log(V_{\text{max}} - 2) - \log k \leq \log(V_{\text{max}} - 2) - \log \log(V_{\text{max}} - 2)$.

By construction, the height of $B$ 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 $\mathcal{VP}(s)$, except for the two edges where $s$ touches $\partial P$, we define a directed segment witness $\mathcal{W}(e)$ inside $P$ that connects $s$ to $e$. This witness $\mathcal{W}(e)$ is a witness of the fact that $e$ is visible from $s$. Note that we have at least $V_{\text{max}} - 2$ witness edges. We partition the set of witness edges into $k$ subsets, as follows. For each triangle $\delta_i$, let $\operatorname{exit}(\delta_i)$ denote the edge of $\delta_i$ that does not intersect $s$. We put each witness $w := \mathcal{W}(e)$ into a subset associated with the first exit edge that it intersects. Let $W_{i, r}$ denote the set of witnesses associated with $\mathcal{W}(e)$, and let $\operatorname{exit}(\delta_i)$ be the exit edge with the maximum number of witnesses associated with it. Note that $|W_{i, r}| \geq (V_{\text{max}} - 2)/k$.

We now construct the tree $B$, whose nodes will correspond to edges of triangles in $T_{\text{opt}}$, iteratively as follows. The root of $B$ is the edge $\operatorname{exit}(\delta_i)$. Consider the witnesses $w \in W_{i, r}$ in arbitrary order. For each $w := \mathcal{W}(e)$, 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 $B$ starting at the root in such a way that the node visited in $B$ 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 $\nu$ of $B$ that we are in (and which corresponds to $e'$) does not

---

**Figure 8:** (a) The weak visibility polygon of $s$. (b) A part of the binary tree $B$. The node at the root is the exit edge $\mathcal{W}(e)$. At some point this may no longer be possible: then we step from an edge $e'$ to an edge $e''$ while the node $\nu$ of $B$ 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 $\mu$, which corresponds to $e''$—see Fig. 8(b). When we step from $e''$ to $e'''$, we create a child for $\mu$ which corresponds to $e'''$. Thus, effectively we create a path hanging from $\nu$ that corresponds to the nodes intersected after $e'$. It is important to note that an edge can in fact appear multiple times in $B$. When we enter a triangle through a given edge, we can obviously only leave it through two other triangles. (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 $B$ is a binary tree. Moreover, the path from the root of $B$ to any leaf corresponds to the sequence of triangle edges intersected by some witness. Hence, the height of $B$ is indeed a lower bound on the stabbing number of $T_{opt}$.

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 $VP(e')$. Note that $P' \setminus VP(e')$ consists of several subpolygons, each separated from $VP(e')$ by a chord which is called the window of the subpolygon. We recur on each subpolygon with its window as designated edge. This weak-visibility-polygon decomposition can be found in $O(n)$ time in total. Note that any line segment $s \subset P$ intersects at most three of the weak visibility polygons.

The method of Hershberger and Suri [5], makes a Steiner triangulation with stabbing number $(\log n)$ 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 $O(\log V_{\max}) = O(\log n)$.

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 9. 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 $\Delta$, and let $n_\Delta$ be the number of Steiner vertices on the boundary of $\Delta$. We move each vertex slightly outward, so $\Delta$ 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. Because $n_\Delta = O(V_{\max})$ for any diamond $\Delta$, and each of the visibility polygons has at most $V_{\max}$ vertices, 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 $O(\log n)$ 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 been able to come up with a very efficient randomized 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