Stabbing Segments with Rectilinear Objects

Mercè Claverol∗ Delia Garijo† Matias Korman‡ Carlos Seara‡ Rodrigo I. Silveira§

Abstract

Given a set of $n$ line segments in the plane, we say that a region $R \subseteq \mathbb{R}^2$ is a stabber if $R$ contains exactly one endpoint of each segment of the set. In this paper we provide efficient algorithms for determining whether or not a stabber exists for several shapes of stabbers. Specifically, we consider the case in which the stabber can be described as the intersection of isothetic halfplanes (thus the stabbers are halfplanes, strips, quadrants, 3-sided rectangles, or rectangles). We provide efficient algorithms reporting all combinatorially different stabbers of that shape. The algorithms run in $O(n)$ time (for the halfplane case), $O(n \log n)$ time (for strips and quadrants), $O(n^2)$ (for 3-sided rectangles), or $O(n^3)$ time (for rectangles).

1 Introduction

We say that a region $R \subseteq \mathbb{R}^2$ is a stabber for a collection of line segments $S$ if $R$ contains exactly one endpoint of each segment of the set (see Figure 1). We study the problem of computing all the combinatorially different stabbers of a given set of segments, for several stabbers. Specifically, we consider stabbers that can be described as the intersection of isothetic halfplanes. Thus, the shapes we consider are halfplanes, strips, quadrants, 3-sided rectangles, and rectangles.

Perhaps the simplest stabber one can consider is a halfplane, whose boundary is defined by a line. Thus a stabbing halfplane is equivalent to a line that intersects all segments. In this context, Edelsbrunner et al. [14, 15] presented an $\Theta(n \log n)$ time algorithm for solving the problem of constructing a representation of all stabbing lines (with any orientation) of a given set of $n$ segments. Moreover, they also give an $\Omega(n \log n)$ lower bound for the problem. However, the lower bound from [15] does not apply to the decision problem (i.e., determining whether or not there exists a line stabber for a set of segments). An $\Omega(n \log n)$ lower bound for the decision problem, in the fixed order algebraic decision tree model, was later presented by Avis et al. [4].

Whenever no stabbing halfplane exists, it is natural to ask for a stabbing wedge (the stabbing region defined by the intersection of two halfplanes). Claverol et al. [11] studied the problem of reporting all combinatorially different stabbing wedges for $S$. The time and space complexities of their algorithm depends on two parameters of $S$. In the worst case, the algorithm runs in $\Theta(n^3 \log n)$ time and $\Theta(n^2)$ space. The authors of [11] also studied some other stabbers/classifiers such as double-wedges, 2-level trees, and zigzags (see [11] for a table comparing the time and space complexities for these stabbers).

The related problem of deciding whether $S$ can be stabbed by a circle (another stabber with constant complexity) was studied by Claverol [10], using the same criterion used in this paper, i.e., endpoints inside (resp. outside) the stabbing circle are red (resp. blue). She gave an $O(n^2 \sqrt{n \log n})$ time algorithm for computing a minimum-radius stabbing circle (if one exists); for parallel segments she shows how to compute all the $O(n)$ combinatorially different stabbing circles in $O(n^2)$ time and $O(n)$ space.

Very recently, Díaz-Báñez et al. [13] considered a similar stabbing concept: a region $R$ stabs a collection of segments if at least one endpoint of each segment is in $R$. Note that under this model a set of segments can always be stabbed (i.e., the whole plane itself is a stabber). Thus, they search for the stabber of minimum perimeter or area. In their work, they provide a polynomial-time algorithm for solving the

∗Dept. de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya, Spain, merc@ma4.upc.edu.
†Dept. de Matemática Aplicada I, Universidad de Sevilla, Spain, dgarijo@us.es.
‡Dept. de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Spain, \{matias.korman, carlos.seara, rodrigo.silveira\}@upc.edu.
§Dept. de Matemática, Universidade de Aveiro, Portugal, rodrigo.silveira@ua.pt.
problem for the particular case in which segments are pairwise disjoint. In addition, they show that the problem is NP-hard when the segments are allowed to overlap.

A generalization of the previous problems that also has been studied is finding color-spanning objects. In this case, the input is a set of \( n \) colored points, with \( k \) colors in total, and the goal is to find an object (rectangle, circle, etc.) that contains at least one point from each color. Our setting, in which we have \( n \) line segments, can be seen as a particular case in which each segment has a different color, thus having \( 2n \) points with \( k = n \) different colors. As before, these color-spanning objects always exist. Thus, the aim is to find one of smallest size. The color-spanning objects studied are strips, axis-parallel rectangles [1, 12], and circles [2]. In all cases, the problem can be solved in roughly \( O(n^2 \log k) \) time.

Our setting, in which the stabber must contain exactly one endpoint from each segment, implies that any stabbing region \( R \) implicitly classifies all endpoints of \( S \) into two sets: the ones inside \( R \) and the ones outside \( R \). Thus, our problem fits into the general framework of separability problems.

Separability problems have been widely investigated and arise in many diverse problems in computational geometry. Several different criteria have been used, depending on what intersection criteria with segments or objects are adopted. Basically, there are three different conditions used in the intersection criteria with segments: either (i) the region must contain exactly one endpoint of each segment, (ii) the region must contain at least one endpoint of each segment, or (iii) the stabbing region must intersect with all segments (but no restriction on the endpoints is given).

Previous work on stabbers focuses mostly on optimization problems such as minimizing the perimeter or area of the stabbers. The criterion is defined by either conditions (ii) or (iii). We cite some relevant previous work on stabbing problems with these criteria that focuses on optimization problems in two dimensions [3, 7, 8, 17, 19, 21–23, 25]. Variants of these problems have also been studied in three dimensions (e.g. [5, 9, 16, 18, 24]).

1.1 Definition and Results

The input consists of a set \( S = \{s_1, s_2, \ldots, s_n\} \) of \( n \) segments. For simplicity in the exposition, we assume that there is no horizontal or vertical segment in \( S \), and that all segments have non-zero length. The modifications needed to make our algorithms handle these special cases are straightforward, albeit rather tedious.

For any \( 1 \leq i \leq n \), let \( p_i \) and \( q_i \) denote the upper and lower endpoint of \( s_i \), respectively. Given a point \( p \in \mathbb{R}^2 \), let \( x(p) \) (resp. \( y(p) \)) denote its \( x \)- (resp. \( y \)-) coordinate.

Any stabbing region \( \mathcal{R} \) implicitly classifies all endpoints of \( S \) into two classes. We say that a point is red if it is in \( \mathcal{R} \), or blue otherwise. Note that we view \( \mathcal{R} \) as a closed region, thus points on its boundary are also considered red.

In the remaining of the paper we present efficient algorithms for stabbers that can be described as the intersection of 2, 3 or 4 isothetic halfplanes. In Section 2 we provide efficient algorithms for the case in which the stabber is formed by at most two halfplanes: they are either strips or wedges. We present a general approach that partitions the plane into three regions: a red region that must be contained in any stabber, a blue region that must be avoided by any stabber, and a gray region for which we do not
have enough information yet. The algorithms are based on iteratively classifying segments and updating the boundaries of these regions. In Section 3 we focus on 3-sided rectangles and proper rectangles. Unfortunately, it is not clear how to extend the approach used for strips and wedges to the case in which the stabber is formed by 3 or 4 halfplanes. Thus, our algorithms for these types of stabbers are based on reusing the one for strips repeatedly in an almost black-box fashion, resulting in relatively high running times.

We note that even though we present our algorithms for stabbers defined by isothetic halfplanes, they all extend to halfplanes that have any two fixed orientations, by applying an appropriate affine transformation.

Note: Parallel to this research, Barba et al. [6] studied a generalization of our problem. The input is a set of \( n \) points partitioned into \( k \) color classes (similar to [1, 2, 12]). However, in addition we are given the number of points of each color that need to be covered. Their goal is to find a color-spanning object containing exactly the given number of points of each color. The objects considered in [6] include discs, squares, and axis-aligned rectangles. The last variant includes, as a particular case, the problem studied here for isothetic rectangles (i.e., we solve the particular case in which \( k = n/2 \) and exactly one point from each color must be covered). The algorithm presented in [6] can be used to solve this problem in \( O(n^2k) = O(n^3) \) time, as does our algorithm in Section 3.

2 Stabbing With One or Two Halfplanes

In this section we look for stabbers that can be described as the intersection of at most two halfplanes. That is, our aim is to obtain a halfplane, strip, or quadrant that contains exactly one endpoint from each segment. Note that a stabbing object is not always certified to exist.

2.1 Stabbing Halfplanes

As a warm-up, we give a simple algorithm for determining if a stabbing horizontal halfplane exists. That is, a horizontal line such that one of the (closed) halfplanes defined by the line contains exactly one endpoint from each segment. Observe that such a stabbing halfplane can be perturbed so that it does not have an endpoint of a segment on its boundary. In this case, the complement of a stabbing halfplane is also a stabber. Thus, we are effectively looking for a horizontal line that intersects all segments.

Although the algorithm is straightforward, we explain it for completeness, since it will be used in the upcoming sections. Since we are dealing with horizontal stabbers, the problem becomes essentially 1-dimensional, and it will ease our presentation to state the problem in that way. All segments can be projected onto the \( y \)-axis, becoming intervals. Let \( \mathcal{S} = \{s_1, \ldots, s_n\} \) be the set of projected segments. If we consider the set of projected segments, the question is simply whether all the intervals in \( \mathcal{S} \) have a point in common.

Consider the values \( y_b = \max_{s_i \in \mathcal{S}} \{y(q_i)\} \) and \( y_t = \min_{s_i \in \mathcal{S}} \{y(p_i)\} \), which correspond to the \( y \)-coordinates of the highest bottom endpoint and the lowest top endpoint, respectively. Then any horizontal line \( y := u \) stabbing \( \mathcal{S} \) must have its \( y \)-coordinate between those two values, namely \( y_b \leq u \leq y_t \). Therefore such a line exists if and only if \( y_b \leq y_t \). This simple observation directly leads to a straightforward linear-time algorithm.

Observation 1 Given a set \( \mathcal{S} \) of segments, we can determine whether a horizontal stabbing halfplane exists (and if so, report it) in \( O(n) \) time.

2.2 Stabbing Strips

We now consider the case in which the stabber is formed by a horizontal strip. Note that the existence of a horizontal stabbing halfplane directly implies the existence of a horizontal stabbing strip, but the reverse is not true.
Figure 2: Computing a horizontal strip stabber. Left: result after classifying $s_i$ and $s_b$. Right: result after cascading; The red region has grown, and only one segment remains unclassified. In both images, the segments of $U$ are shown dotted, those of $W$ are dashed, and those of $C$ are depicted with a solid line.

As before, we can ignore the $x$-coordinates of the endpoints, project the points to the $y$-axis, and work with the set $S$ instead. The endpoints of the classified segments can be seen in the projection onto the $y$-axis as a set of blue and red points, see Figure 2 (left). Clearly, there is a separating horizontal strip for them if and only if the red points appear contiguously on the $y$-axis. In particular, the points must appear on the $y$-axis in three contiguous groups, from top to bottom, first a blue group, then a red group, and then another blue group. We refer to the two groups of the blue points as the top and bottom blue points, respectively. We denote the intervals of the $y$-axis spanned by them by $B_t$ and $B_b$, respectively. The interval of the $y$-axis spanned by the group of red points is denoted by $R$.

Moreover, since all points above $B_t$ and below $B_b$ must be necessarily blue, we extend $B_t$ and $B_b$ from $+\infty$ and until $-\infty$, respectively. In this way the $y$-axis is partitioned into three colored intervals and two uncolored (gray) intervals separating them.

Our algorithm relies on the following observations.

**Observation 2** Let $s_i$ and $s_j$ be two segments in $S$ such that $\pi_i$ and $\pi_j$ are disjoint. Then, there is a unique way to classify the endpoints of $s_i$ and $s_j$ with a horizontal strip.

**Proof.** Since $\pi_i$ and $\pi_j$ do not overlap, one of them is above the other one. Assume without loss of generality that $s_i$ is above $s_j$. That is, $y(q_i) > y(p_j)$. Then any horizontal stabbing strip separating the endpoints of $s_i$ and $s_j$ must contain in its interior the bottom endpoint of $s_i$ and the top endpoint of $s_j$. □

**Observation 3** Let $S$ be a set of segments without a stabbing halfplane, and let $s_b$ and $s_t$ be the segments in $S$ with highest bottom endpoint and lowest top endpoint, respectively. Then $\pi_b$ and $\pi_t$ are disjoint.

**Proof.** Let $y_b$ and $y_t$ be the values defined in Section 2.1. It follows from the definition of $s_b$ and $s_t$ that $y(q_b) = y_b$ and $y(p_t) = y_t$. Since $S$ has no horizontal stabbing line, we have $y(q_b) = y_b > y_t = y(p_t)$. In particular $\pi_b$ and $\pi_t$ cannot overlap. □

**Corollary 1** Let $S$ be a set of segments $S$ without a stabbing halfplane, and let $s_b$ and $s_t$ be the segments in $S$ with highest bottom endpoint and lowest top endpoint, respectively. Then there is a unique way to classify $s_b$ and $s_t$ with a horizontal strip.

With the previous observations in place, we can present our algorithm. We virtually partition $S$ into three sets $C$, $W$, and $U$. A segment is in $C$ if it has been already classified, in $W$ if it is waiting for
being classified (there is enough information to classify it, but it has not been done yet), or in $U$ if its classification is still unknown. The algorithm is initialized with $C = W = \emptyset$, $U = S$.

**Regions** In addition to the three sets of segments, the algorithm maintains red and blue candidate regions that are guaranteed to be contained or avoided in any solution, respectively. Technically speaking, we maintain the projection of the regions on the $y$-axis, but it is essentially the same thing. The regions are represented by intervals $B_i$, $R$, and $B_b$. We define the blue region as $B_i \cup B_b$ and the red region as $R$. The complement of the union of the blue and red regions is called the gray region. Note that the gray region, like the blue one, consists of two disjoint components. During the execution of the algorithm the regions will become updated as new segments become classified.

The algorithm starts by computing $s_b$ and $s_t$. If a stabbing halfplane exists, we can find a stabbing strip and report it. Otherwise, by Corollary 1 we know how to classify both segments (that is, we can classify $q_b$ and $p_t$ as red, and $p_b$ and $q_t$ as blue). Thus, we move them $U$ to $W$. Next we further classify segments through a *Cascading procedure*. We explain this procedure in rather general terms because it will also be used in the upcoming sections.

**Cascading procedure** The procedure iteratively classifies all segments in group $W$ based on the red and blue regions. This is an iterative process in the sense that the classification of one segment can make the blue or red region grow, making other segments move from $U$ to $W$.

We classify each segment in $W$, in any order, assigning the corresponding colors and moving it from $W$ to $C$. Then, based on the color assignment, the red or blue areas are modified accordingly (one of them may grow). Note that after the red or blue region grows, other segments can change from $U$ to $W$.

This classification process is repeated until either (i) a contradiction appears (the red region is forced to overlap with the blue region), or (ii) group $W$ is empty.

**Lemma 1** If the cascading procedure finishes without finding a contradiction, each remaining segment in $U$ has an endpoint in each of the connected components of the gray region.

**Proof.** If the cascading procedure finishes without finding a contradiction, then all segments still in $U$ must have both endpoints in the gray region (otherwise, list $W$ would not be empty). Assume, for the sake of contradiction, that there exists a segment $\bar{s}_i \in U$ whose both endpoints lie in the same gray component. Without loss of generality, assume it is the lower one. Recall that, by construction, the red region contains the interval $[y_b, y_t]$. In particular, we have $y(p_t) < y_b$, giving a contradiction with the definition of $y_t$. \hfill \square

**Corollary 2** A solution exists if and only if the cascading procedure finishes without finding a contradiction.

**Proof.** From the above observations, it is clear that if the cascading procedure finds a contradiction, then there will not be a solution. Likewise, if no contradiction is found, we can find a solution by extending the bottom boundary of $B_i$ until it covers the upper gray component and extending the bottom boundary of $R$ until it covers the lower gray component. \hfill \square

**Theorem 2** Determining whether a stabbing strip exists for a set of $n$ segments can be done in $O(n \log n)$ time and $O(n)$ space.

**Proof.** The correctness follows from the previous observations, thus we focus on the running time. In order to implement the cascading procedure efficiently, we must be able to quickly find if there is a segment that has an endpoint in $B_i$, $B_b$, or $R$. Since the red and blue regions are simply intervals, this can be done by maintaining a balanced binary tree (sorted by $y$-coordinate) with the endpoints of all unclassified segments. In this way, a segment with an endpoint in, say, $B_i$ can be found in $O(\log n)$ time by querying the tree with the endpoints of $B_i$. Every time an endpoint inside $B_i$ is found, it is removed from the tree, and moved into $W$. When classifying a segment, then the limits of $B_i$, $B_b$, or $R$
must also be updated accordingly. However, since these intervals have constant complexity, they can be
updated in constant time. Since no interval is classified more than once, the total time for the cascading
operation is bounded by \( O(n \log n) \).

\[ \square \]

**Theorem 3** All the combinatorially different stabbing horizontal strips of a given set \( S \) of \( n \) segments
can be computed in \( O(n \log n) \) time.

**Proof.** The only new operations that the algorithm introduces are searching for the indices \( t \) and \( b \),
and execute further cascading. Since we keep the elements of \( U \) in balanced binary trees, the former
operation can be done in logarithmic time. As in the decision version, each segment can only trigger
one cascade operation, hence the total running time is also bounded by \( O(n \log n) \).

\[ \square \]

### 2.3 Stabbing Quadrants

The previous approach essentially partitioned the plane into three regions: the red region is guaranteed
to be included in any solution, the blue region must be avoided by any solution, and the gray region
where the endpoints of the unclassified segments remain. If the cascading finishes without finding a
contradiction, we conclude that each unclassified segment must have an endpoint in each of the connected
components of the gray regions. Thus, we can enlarge the red region to contain one of the connected
components of the gray region to obtain a solution.

We now extend this same approach for quadrants (i.e. a wedge formed by a horizontal and a vertical
ray from a common point, the apex of the quadrant). There are four types of quadrants. Without loss
of generality, we concentrate on the bottom-right type. Thus throughout this section, quadrant, unless
otherwise stated, refers to a bottom-right quadrant. Other types can be handled analogously.

For a segment \( s = (p, q) \), let \( Q(s) \) denote its bottom-right quadrant; that is, the quadrant with apex at \((\max\{x(p), x(q)\}, \min\{y(p), y(q)\})\). See Figure 3(left).

**Observation 4** Any quadrant classifying a segment \( s \) must contain \( Q(s) \).

We can now define a similar notation for a set of segments \( S \). We define the bottom-right quadrant of
\( S \), denoted \( Q(S) \), as the (inclusionwise) smallest quadrant that contains \( \cup_{s \in S} Q(s) \), see Figure 3(center).
The following corollary will be crucial for our algorithm.

**Corollary 3** Any stabbing quadrant of $S$ must contain $Q(S)$.

**Regions** The partition into regions is as follows: the red region is given by a quadrant $R$. Corollary 3 gives us a first candidate, thus we start with $R = Q(S)$. Note that after the cascading procedure, $R$ might grow, but it will always contain $Q(S)$. At any point in the execution, let $a = (x_R, y_R)$ denote the apex of $R$. Any blue point $b$ of a classified segment forbids the stabber to include $b$ or any point above and to the right of $b$ (i.e., in the top-left quadrant of $b$). Moreover, if $b$ satisfies $y(b) \leq y_R$ or $x(b) \geq x_R$ a whole halfplane will be forbidden (see Figure 3(right)). The union of such regions forms a staircase polygonal line. Initially, we take the blue region defined by a point at $(-\infty, -\infty)$. We say that a point $p = (x, y)$ is in the gray region if it is not in a red or blue region, and satisfies either $x \geq x_R$ or $y \leq y_R$, see Figure 3(right). As before, observe that the gray region is the union of two connected components (which we call right, and down components). Note that, for this case, there are regions in the plane (which we call white) that do not belong to neither red, blue, or gray regions. However, our first observation is that no endpoint of an unclassified segment can lie in the white region.

**Observation 5** A segment $s \in S$ containing an endpoint in the white region must contain its other endpoint in the red region.

**Proof.** This fact follows from the definition of $Q(s)$: assume that there exists a segment $s \in S$ that has an endpoint in the white region and an endpoint in the right gray component. On particular, the $y$-coordinate of both endpoints is larger than $y_R$. Thus, the quadrant $Q(s)$ is not contained in $R$. However, recall that, by construction, we have $Q(s) \subseteq Q(S) \subseteq R$; obtaining a contradiction. Similar contradictions are found when the other endpoint of $s$ is in the other gray component, or in the white or blue regions. □

Initially, we place all segments that have one endpoint inside $Q(S)$ in $W$, and all the rest in $U$. After initializing, we apply the cascading procedure. As with the strip case, if the cascading operation finds a contradiction, we can conclude that a stabbing quadrant does not exist. Otherwise, we have a very strong characterization of the remaining unclassified segments.

**Lemma 4** If the cascading procedure finishes without finding a contradiction, each remaining segment in $U$ has an endpoint in each of the gray components.

**Proof.** As with the strip case, the cascading procedure cannot finish if a segment in $U$ has an endpoint in either a red or blue region. As done in the proof of Observation 5, if both endpoints lie in the same gray components, its quadrant $Q(s)$ is not contained in $R$, giving a contradiction. □

The above result implies that if no contradiction is found during the cascading procedure, then we can extend $R$ until it contains one of the two 3-sided rectangles and obtain a valid solution. This can be done because, by construction, each of the connected components of the grey region form a 3-sided rectangle that shares a corner with the apex of $R$. Note that we can use the same approach as in the strip case, and report all combinatorially different quadrants by repeatedly applying the cascading procedure to the segments whose endpoints to $R$ are closest.

**Theorem 5** All the combinatorially different stabbing quadrants of a given set $S$ of $n$ segments can be computed in $O(n \log n)$ time.

**Proof.** Proof of correctness is analogous to the strip case, thus we argue about the running time. The red and gray regions have constant complexity, thus they can be updated in constant time. The boundary of the blue region can be determined by $\Omega(n)$ points. However, notice that the points in the staircase are sorted in both the $x$-coordinates and the $y$-coordinates. Thus, we can insert a point in the blue region in $O(\log n)$ time if the points in the blue region are stored in a sorted fashion. Once we have
inserted a new point, we check if this increase makes the red and blue regions to intersect. Since we only need to compare the upper left quadrant defined by the new point with the red region, this can be done in constant time. That is, updating the red, blue and gray regions can be done in $O(\log n)$ time.

Now we need an efficient method to update the segments in $W$ whenever the red or blue region changes. For that purpose, we can use any data structure suitable for range searching within a quadrant. For example, one based on priority search trees [20] will suffice: given a bottomless rectangular query region, after $O(n \log n)$ preprocessing time and using $O(n)$ space, we can report all segments containing an endpoint in the query region (or report that the region is empty) in $O(\log n + k)$ time, where $k$ is the number of segments within that region. Note that the structure is dynamic and can handle both deletions and insertions in $O(\log n)$ time.

Observe that the red and blue regions increase by a quadrant (or a bottomless rectangle) each time, thus by querying the appropriate region, we can find the new segments in, delete them from $U$, and insert them into $W$. Once a segment is reported, we also remove it from the search tree so it is never reported again. Thus, all queries will be computed in total $O(n \log n)$ time and $O(n)$ space. Finally, observe that we can report all stabbing quadrants using the same approach as in the strip case, thus the theorem follows.

## 3 Stabbing With Three or More Halfplanes

In this section we present algorithms for the remaining two orthogonal shapes that reuse the algorithms in the previous section. Unfortunately, it is not clear how to extend the previous red-blue-gray region approach to the case in which the stabber contains three or more halfplanes (that is, 3-sided rectangles, and proper rectangles).

### 3.1 3-Sided Rectangles

We now consider the case in which the stabber is defined as the intersection of three halfplanes. As in the quadrant case, it suffices to consider those of fixed orientation. Thus, throughout this section, a 3-rectangle refers to a bottomless rectangle (thus, it is missing the lower boundary edge to become an actual rectangle).

Given any stabbing 3-rectangle, consider the smallest (inclusionwise) stabbing 3-rectangle $R$ contained in it. It is easy to see that $R$ will contain a point $v$ in the upper boundary segment (otherwise we can further shrink it). If $v$ is known, the problem becomes one-dimensional, and can be handled as in the strip case; the only difference is that a few segments can already be classified (i.e., the segments that have an endpoint with higher $y$-coordinate than $y(v)$). Since $v$ is not known, it suffices to consider all $O(n)$ candidates for $v$ and solve each problem instance using Theorem 2 (where we look for a vertical strip). This gives an algorithm that runs in $O(n^2 \log n)$ time, and uses $O(n)$ space. In the following, we show how to reduce the running time by a logarithmic factor by reusing information between different candidates of $v$.

The main observation lies in the fact that the cascading procedure makes many deletions, but no insertions are ever executed. Thus, instead of using a binary tree, we use an alternative data structure: we explicitly store the endpoints of the segments in $S$ using two different orderings. In the first one we sort the segments in $S$ by the $x$-coordinate of the right endpoint. In the second sorting we sort by the $x$-coordinate of the left endpoint instead.

**Theorem 6** All combinatorially different stabbing 3-sided rectangles of a given set $S$ of $n$ segments can be computed in $O(n^2)$ time and $O(n)$ space.

**Proof.** We store the two sorted lists explicitly in doubly linked lists. In addition, for each entry in either list, we store a pointer to the corresponding position in the other list. It is straightforward to see that this data structure can be created in $O(n \log n)$ time. Moreover, with this data structure we can, given a region (red or blue) that has grown due to a cascading procedure, find the segments of $U$ that must move to $W$ in time proportional to the number of segments to be moved. This is because these segments become moved in order of $x$-coordinate. Moreover, these segments can also be removed from
the structure in the same time (since each endpoint of a segment is linked to its partner in the other ordering).

We now scan all candidates for $v$ in a top-to-bottom fashion. Given a candidate $v$, we first create a working copy of the sorted lists. Then, we scan the segments in $S$: any segment that crosses the line $\{y = y(v)\}$ is moved to $W$, and is removed from the sorted lists. For the other segments we do not have enough information on how to classify it, thus they stay in $U$. The cascading procedure is executed as usual until either (i) a vertical stabbing strip is found for the candidate $v$. This corresponds to a strip together with the halfplane $\{y \leq y(v)\}$, which together correspond to a stabbing 3-rectangle (and thus is reported), or (ii) a contradiction with the regions is found. In such a case we know that our candidate for $v$ was wrong. Thus, we must destroy the working copy of the data structure, and consider the next candidate for $v$.

Thus, after $O(n \log n)$-time preprocessing, the cascading procedure takes linear time per candidate $v$. Since the number of candidates is bounded by $O(n)$, the running time follows. □

3.2 Stabbing Rectangles

Finally, we mention how to extend the algorithm for 3-sided rectangles for the proper rectangle. As before, given any stabbing rectangle, consider the smallest (inclusionwise) stabbing rectangle $R$ contained in it. Now $R$ must contain one endpoint on each side, and in particular, an endpoint $v$ of $S$ must be on its upper boundary segment, and an endpoint $v'$ on its lower boundary segment.

A straightforward algorithm simply consists in executing an instance of the 3-sided rectangle problem for each candidate for $v'$. As in the previous case, we have $O(n)$ candidates for $v'$, thus we obtain the following result.

**Theorem 7** All the combinatorially different stabbing rectangles of a given set $S$ of $n$ segments can be computed in $O(n^3)$ time.

4 Conclusions and open problems

The main contribution of this paper is a general algorithm for classifying the endpoints of segments based on partitioning the plane into red, blue, and gray regions that are updated as points become classified. We showed how to easily apply such algorithms for the case of horizontal strip and quadrant stabbers. A natural question is whether the same approach can be used for other shapes. For the case of 3-sided rectangles and proper rectangles, that would probably result in faster algorithms. Moreover, we could also consider slightly more general shapes such as staircases, orthogonally convex shapes, or the natural extensions of these shapes to higher dimensions.

Another intriguing question is whether there are output-sensitive algorithms for reporting all the different 3-sided rectangles and proper rectangles. This comes together with the question of how many combinatorially different solutions there can be for these two problems. For the strips and quadrants it is easy to see that there can be up to $\Theta(n)$ combinatorially different solutions (fixing the $x$ or $y$ coordinate of a halfplane uniquely determines the second one, and there are $\Theta(n)$ possible choices). However, for the case of rectangles this is not so straightforward, except for an immediate $O(n^3)$ upper bound.

Finally, in all our algorithms it takes the same asymptotic running time to decide whether there is one solution as reporting them all. Is it possible to improve the running times for the decision versions of the problems? This was shown not to be possible for line stabbers [4, 15], but it is unclear if this is also the case in our context. In other words, is it possible to prove lower bounds for any of the stabbers considered in this paper?

**Acknowledgments.** M. C., M. K., C. S., and R.I. S were partially supported by ESF EUROCORES programme EuroGIGA, CRP ComPoSe: grant EUI-EURC-2011-4306, and by project MINECO MTM2012-30951. D. G. was supported by project PAI FQM-0164. M. K. was supported by the Secretary for Universities and Research of the Ministry of Economy and Knowledge of the Government of Catalonia and the European Union. R.I. S. was also partially funded by the FP7 Marie Curie Actions Individual Fellowship PIEF-GA-2009-251235 and by FCT through grant SFRH/BPD/88455/2012.
References


