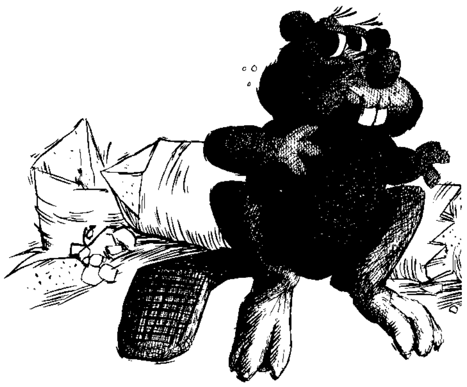

Hole Problems for Rectangles in the Plane

Gregory J.E. Rawlins
Peter Widmayer
Derick Wood

Data Structuring Group
CS-86-49

October, 1986

Hole Problems for Rectangles in the Plane [*]

Gregory J.E. Rawlins [†] Peter Widmayer [‡] Derick Wood [§]

October 9, 1986

Abstract

Given a set of n rectangles with sides parallel to the coordinate axes, we show how to determine whether their union, viewed as a set of disjoint polygons, has a hole. Our algorithm needs no more than $O(n \log n)$ time and $O(n)$ space, which we show to be optimal. However, in practice we also need to know the locations of holes, if there are any. We present an algorithm to determine the locations of all h holes in $O(n \log n + h)$ time and $O(n)$ space, which is again optimal. The algorithm computes a point within each hole, representing the location of the hole. The efficiency of these and several other algorithms follow from some simple combinatorial arguments about sets of rectangles in the plane.

1 Introduction

The problem of gathering information about holes in the union of a set of rectangles with sides parallel to the coordinate axes is of theoretical interest and of importance as a step in the computation of the contour of Boolean combinations of sets of rectangles. Certainly, any algorithm reporting the contour of the union of a set of rectangles (see e.g. [3,2,9]) will also report the contours of all holes. However, such an algorithm uses time $O(n \log n + p)$ for n rectangles and p edges in the contour, where $p = O(n^2)$ in the worst case.

[*]The first and third authors were supported under a Natural Sciences and Engineering Research Council of Canada grant No. A-5692 and the work of the second author by a grant from the Deutsche Forschungsgemeinschaft Wi 810/1-1

[†]Data Structuring Group, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

[‡]Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe, Postfach 6980, 7500 Karlsruhe, West Germany

[§]Data Structuring Group, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

For some questions concerning holes, this may be too much information to be desirable. Specifically, we answer the following simpler problems more efficiently.

Given a set of n rectangles in the plane, with sides parallel to the coordinate axes,

1. Does the union of the rectangles have a hole?

2. Report one point in each hole of the union of the rectangles.

In order to solve Problems (1) and (2) efficiently, we establish a few simple combinatorial properties of a set of rectangles in Section 2. We would certainly like Problem (1) to be decidable in time independent of the number of edges of the holes or even independent of the number of holes. We show in Section 3 that $\Omega(n \log n)$ is a lower bound for Problem (1), and we present an algorithm answering Problem (1) in time $O(n \log n)$.

For Problem (2), the number h of holes is the size of the output. From the lower bound of Problem (1), a lower bound of $\Omega(n \log n + h)$ follows directly for Problem (2). An algorithm to answer Problem (2) in time $O(n \log n + h)$ is presented in Section 4. Finally, we present some final remarks in Section 5.

2 Basic Definitions and Combinatorial Properties

We are given a set $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$ of n rectangles in the plane. Each rectangle R_i, $1 \leq i \leq n$, is described by a quadruple $(x_l, x_r, y_b, y_t)_i$ of coordinates for its left x-, right x-, bottom y- and top y-coordinate. We identify rectangle R_i with the two-dimensional interval $[x_l, x_r] \times [y_b, y_t]$ in the plane. A point $p = (x, y)$ in the plane is said to be *covered* by rectangle R_i if $x_l \leq x \leq x_r$ and $y_b \leq y \leq y_t$ for $R_i = (x_l, x_r, y_b, y_t)$; rectangle R_i is said to *cover* p. Let $\mathcal{U}$ denote the union of all points covered by some rectangle in $\mathcal{R}$. Without loss of generality, assume that $\mathcal{U}$ is a connected set (the connected components of $\mathcal{R}$ can be computed in time $O(n \log n)$, see [1]). If there are several connected components, one can lie inside a hole of another; because we are interested in holes with respect to the complexity of computations, we consider all components separately. $\mathcal{U}$ can be described as a polygon P with sides parallel to the coordinate axes. P divides the plane $\Re^2$ into a number of connected components of maximal size. That is, $\Re^2 - P = \{P_i | 0 \leq i \leq h\}$, for $h \geq 0$, and each P_i is a polygon describing a connected set of points not covered by any rectangle in $\mathcal{R}$, and no two polygons P_i, P_j intersect for $i \neq j$. P_0 is defined to be the polygon containing

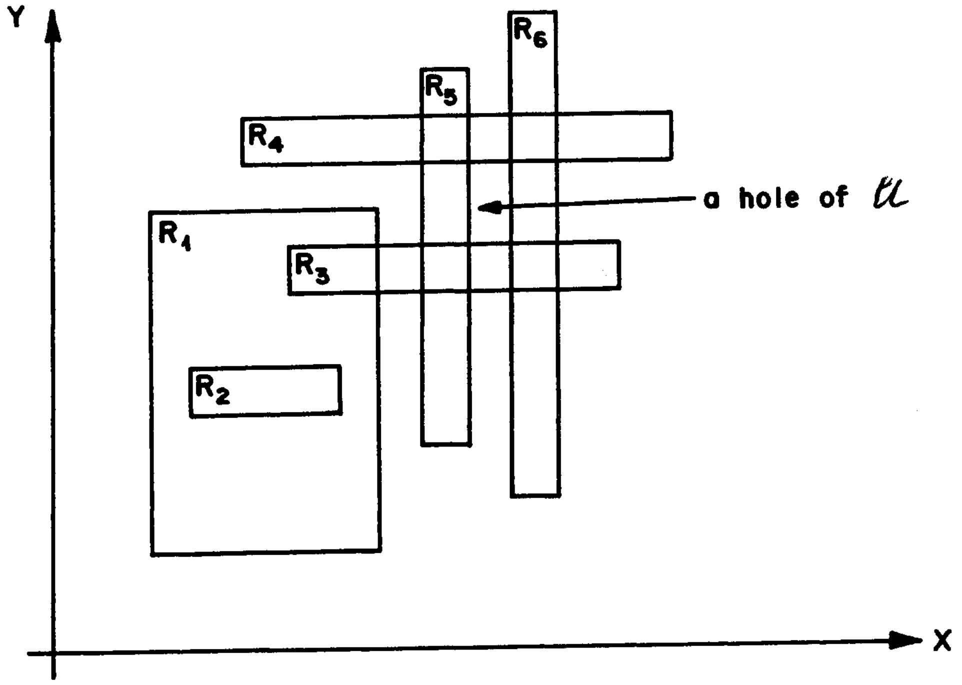

Figure 1: The basic situation.

point (∞, ∞); it is called the *exterior* of $\mathcal{U}$. Polygons $P_1, \ldots P_h$ are called the *holes* of $\mathcal{U}$.

For an example illustrating the definitions, see Figure 1. Note that there need not be any holes in $\mathcal{U}$. To simplify the discussion, but with no loss of generality, we assume that the rectangles are in *general position*, that is, no two of the x-values (y-values) in the input data are the same.

Recall that n is the number of rectangles and h is the number of holes in $\mathcal{U}$. Let c or, equivalently, h_0, be the number of edges of the exterior P_0 of $\mathcal{U}$, and let h_i be the number of edges of hole P_i, $1 \leq i \leq h$. Then we know from [3]:

Proposition 2.1 ([3]) *If $\mathcal{U}$ has no holes, then $c \leq 8n - 4$.*

[8] have generalized Proposition 2.1 to allow holes:

Proposition 2.2 ([8]) *$c \leq 8n - 4$, for n and c as defined above.*

Proof: We present a proof of this Proposition that is different from the one in [8], with the aim of using the insights it provides in subsequent proofs.

Any set of $n+1$ rectangles in the plane in one connected component can be obtained by adding a rectangle to a set of n rectangles in a connected component, for $n \geq 1$. For $n = 1$, the claim holds, because $c = 4$. Now suppose our claim holds for some $n \geq 1$. Consider a set of $n+1$ rectangles. Remove a rectangle R_{n+1}. Then the intersection of the boundary of R_{n+1} with the uncovered area $\Re^2 - P$ consists of line segments on the boundary of R_{n+1} (see Figure 2). At most 8 line segments contain a vertex of R_{n+1}; these line segments may increase the number of edges of either the exterior P_0 or of holes (see Figure 2, edges e_1 to e_5). Consider a line segment on the boundary of R_{n+1} obtained from the intersection with P_0, say line segment l_0. Then, either l_0 is an edge of the exterior P_0' of $\mathcal{U}' = \mathcal{U} \cup R_{n+1}$, or l_0 is an edge of a newly created hole P_j, $j > h$ (see Figure 2, edges e_6 to e_{12}). In the former case, the number of edges of P_0 does not increase, because l_0 replaces at least one edge in P_0, that is, at least one edge e_0 of P_0 is not an edge of P_0', because l_0 belongs to P_0', and e_0 either lies in $\mathcal{U}'$ or belongs to a hole in $\mathcal{U}'$. Therefore, c', the number of edges of P_0', satisfies $c' \leq c + 8 \leq 8n - 4 + 8 = 8(n+1) - 4$. □

Corollary 2.3 *For any hole P_i of $\mathcal{U}$,*

$h_i = 0$ *for* $n \leq 3$,

$h_i \leq 4$ *for* $n = 4$, *and*

$h_i \leq 8n - 32$ *for* $n \geq 5$,

Proof: By arguments similar to those for Proposition 2.2. □

Note that all of the stated bounds are tight, that is, there exist examples for arbitrary n achieving the stated bound exactly. For Proposition 2.2, the example can be constructed as in [3] (Figure 3). For Corollary 2.3, Figure 4 provides an example. We have seen so far that the exterior of $\mathcal{U}$, P_0, as well as any hole P_i, $1 \leq i \leq h$, of $\mathcal{U}$, may have a number of edges that is linear in n. The example in Figure 5 shows that the number of holes may be as high as $(\frac{n}{2} - 1)^2$. However, the total number of edges in all holes and the exterior of $\mathcal{U}$ together, is much less than the combination of these figures might suggest.

Lemma 2.4 $e \leq 8n + 4h - 4$, *where* $e = \sum_{i=0}^{h} h_i$ *is the total number of edges in the exterior and in all holes of* $\mathcal{U}$.

Proof: Based on the proof of Proposition 2.2, consider the case where l_0 is not an edge of P_0', but an edge of a hole P_j, $j > h$. Then P_j may have just been created, that is, before adding R_{n+1}, P_j was part of P_0. In this case, l_0 accounts for one new hole. Another possibility is that P_j has been obtained by splitting a hole into two. In general, i edges on the

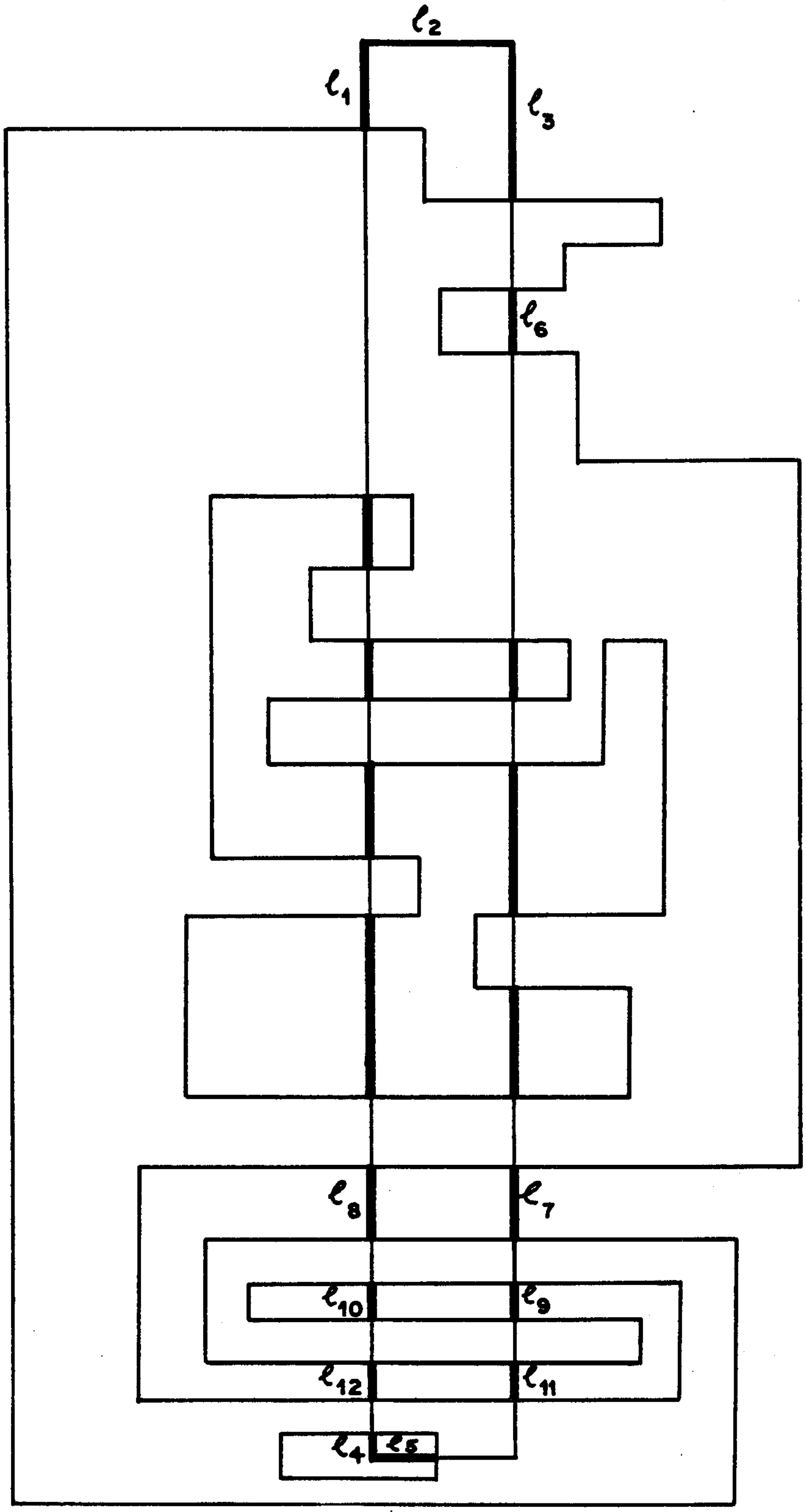

Figure 2: Illustrating the proof of Proposition 2.2.

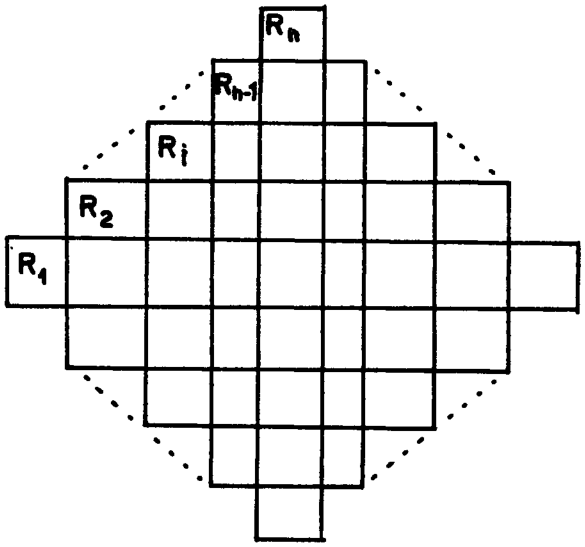

Figure 3: The bound in Proposition 2.2 is tight.

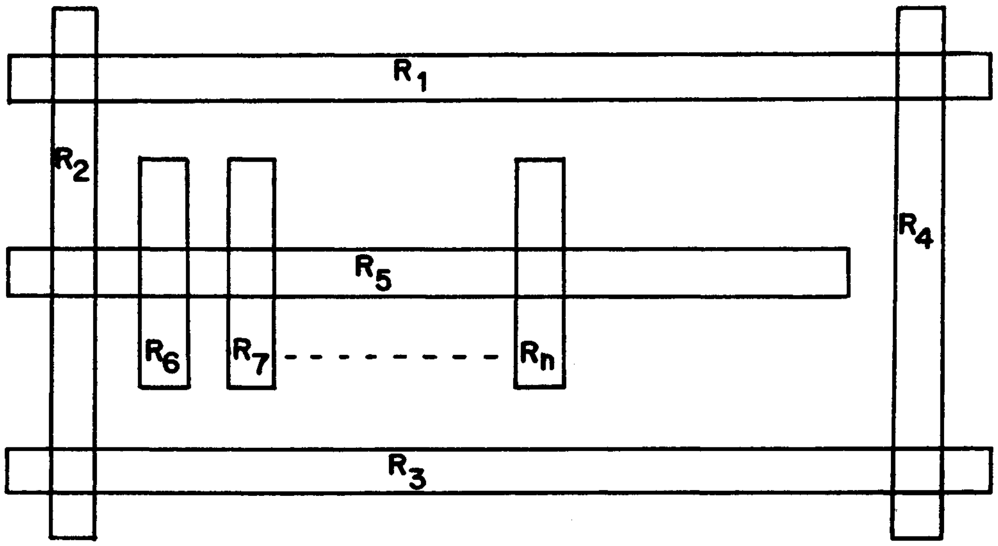

Figure 4: The bounds in Corollary 2.3 are tight.

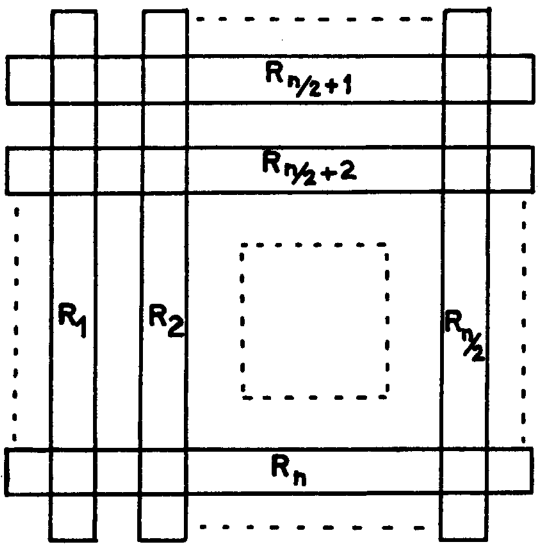

Figure 5: The maximum number of holes.

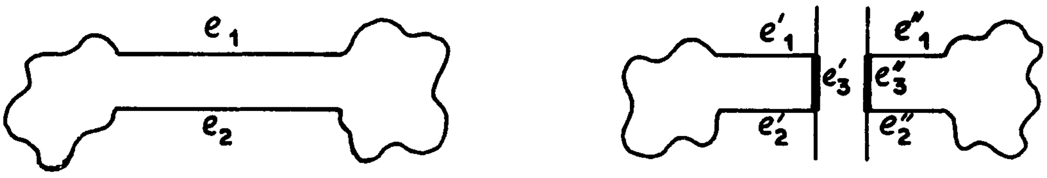

Figure 6: Splitting a hole (before and after).

boundary of R_{n+1} may create i new holes from an old one, that is, add $i-1$ new holes. For $i=2$, the increase in the number of edges by creating a new hole is at most 4 (see Figure 6). For $i>2$, the increase in the number of edges per new hole is less. Therefore, we get $e' \leq e+8+4h'$, where h' is the additional number of holes created by R_{n+1}. Altogether, $e' \leq e+8+4h' \leq 8n+4h-4+8+4h' = 8(n+1)+4(h+h')-4$. □

Next, we show that this bound is tight , that is, for any given n and h such that h holes can be formed by a set of n rectangles, there is a set of n rectangles forming h holes and having a total of e edges in the exterior and in all holes of $\mathcal{U}$. To this end, we first need to know the maximum possible number h of holes for a given n.

Lemma 2.5 *The number of holes in any arrangement of n rectangles is at most $(\lceil \frac{n}{2} \rceil - 1)(\lfloor \frac{n}{2} \rfloor - 1)$, for $n \geq 1$.*

Proof: By induction on n. Let $h(n)$ denote the number of holes in an

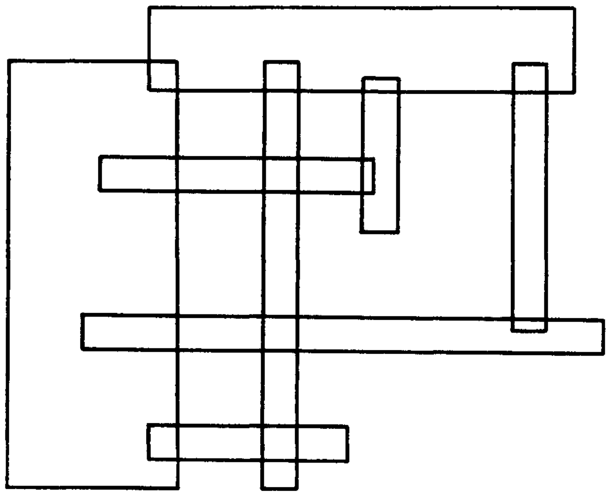

Figure 7: Leftmost and topmost holes and rectangles.

arrangement of n rectangles. We perform induction on odd and even n separately.

Consider a hole with leftmost left boundary, and a hole with topmost top boundary. Any rectangle bounding the former hole to the left is called a leftmost rectangle; any rectangle bounding the latter hole from the top is called a topmost one. Let $d(n)$ denote the decrease in the number of holes of an arrangement of n rectangles, when a leftmost and a topmost rectangle are removed. That is, $d(n)$ is at most the number of holes with a left or a top hole edge from one of the two rectangles being removed. The edges of these holes are, hence, aligned along the two rectangles, in a purely linear fashion (see Figure 7). Therefore, to separate two holes from each other, a rectangle is needed. This rectangle cannot separate any other two of the holes under consideration. As there are just $n - 2$ rectangles left, these can form at most $n-3$ holes in the described way. Therefore, $d(n) \leq n-3$. Now consider induction on even numbers of rectangles, that is, $h(2n)$, for $n \geq 2$. The claim $h(2n) \leq (n-1)^2$ holds for $2n = 4$. Assume the claim holds for all even values up to some value $2n \geq 4$. We show that the claim then holds for $2(n+1)$. To this end, assume the contrary, that is, $h(2(n+1)) > n^2$. Because $d(2(n+1)) \leq 2n - 1$ is the number of holes we have to subtract when removing two rectangles, we get $h(2n) \geq h(2(n+1)) - d(2(n+1)) > n^2 - 2n + 1 = (n-1)^2$, a contradiction to the inductive assumption that $h(2n) \leq (n-1)^2$.

For odd numbers of rectangles, the argument is similar. The claim $h(2n+1) \leq n(n-1)$ holds for $2n+1 = 5$ (we could also start with $2n+1 = 3$ and no holes). Assume the claim holds for all odd values up to some $2n+1 \geq 5$. To see that the claim also holds for $2(n+1)+1$, assume the contrary, that is, assume that $h(2(n+1)+1) > (n+1)n$. By removing the two rectangles to the left and on the top we get $h(2n+1) \geq h(2(n+1)+1) - d(2(n+1)+1) > (n+1)n - 2n = n(n-1)$, a contradiction. □

The example in Figure 5 shows how the upper bound given in Lemma 2.5 can be achieved for even n; for odd n, just add one rectangle in either of the two ways. Any desired number h of holes can be achieved by shrinking rectangles R_i, $1 \leq i \leq \lfloor \frac{n}{2} \rfloor$, in such a way that enough holes are connected to the exterior, and the desired number of holes remains (see Figure 8 for an example with 6 rectangles). If the corner points of all rectangles belong to the boundary of a hole or the exterior, that is, are not covered by some other rectangle, then the bound of Lemma 2.4 is realized exactly. Hence, for any n and h as described, this bound is tight.

For $k \geq 1$ connected components of the set $\mathcal{R}$ we obtain:

Corollary 2.6 *For all $n \geq 1$ and $k \geq 1$:*

1. $c \leq 8n - 4k$;
2. $h_i = 0$ *for* $n - k + 1 \leq 3$;
   $h_i \leq 4$ *for* $n - k + 1 = 4$;
   $h_i \leq 8n - 8k - 24$ *for* $n - k + 1 \geq 5$;
3. $e \leq 8n + 4h - 4k$.

Proof: For k components having $n_j \geq 1$ rectangles in component j, $1 \leq j \leq k$, we observe the following:

1. $c \leq \sum_{j=1}^{k}(8n_j - 4) = 8\sum_{j=1}^{k} n_j - 4k = 8n - 4k$.

2. The number of holes is maximized if all components apart from one consist of only one rectangle, that is, without loss of generality, $n_1 = n-k+1$, $n_2 = n_3 = \ldots = n_k = 1$. Substituting $n-k+1$ in the bound of Corollary 2.3 yields the above bound.

3. Let $h(j)$ be the number of holes in component j, $1 \leq j \leq k$; then $e \leq \sum_{j=1}^{k}(8n_j - 4h(j) - 4) = 8\sum_{j=1}^{k} n_j - 4\sum_{j=1}^{k} h(j) - 4k$, hence $e \leq 8n - 4h - 4k$.

□

Finally, to ensure that our algorithm satisfies the claimed bounds on time and space, we need the following:

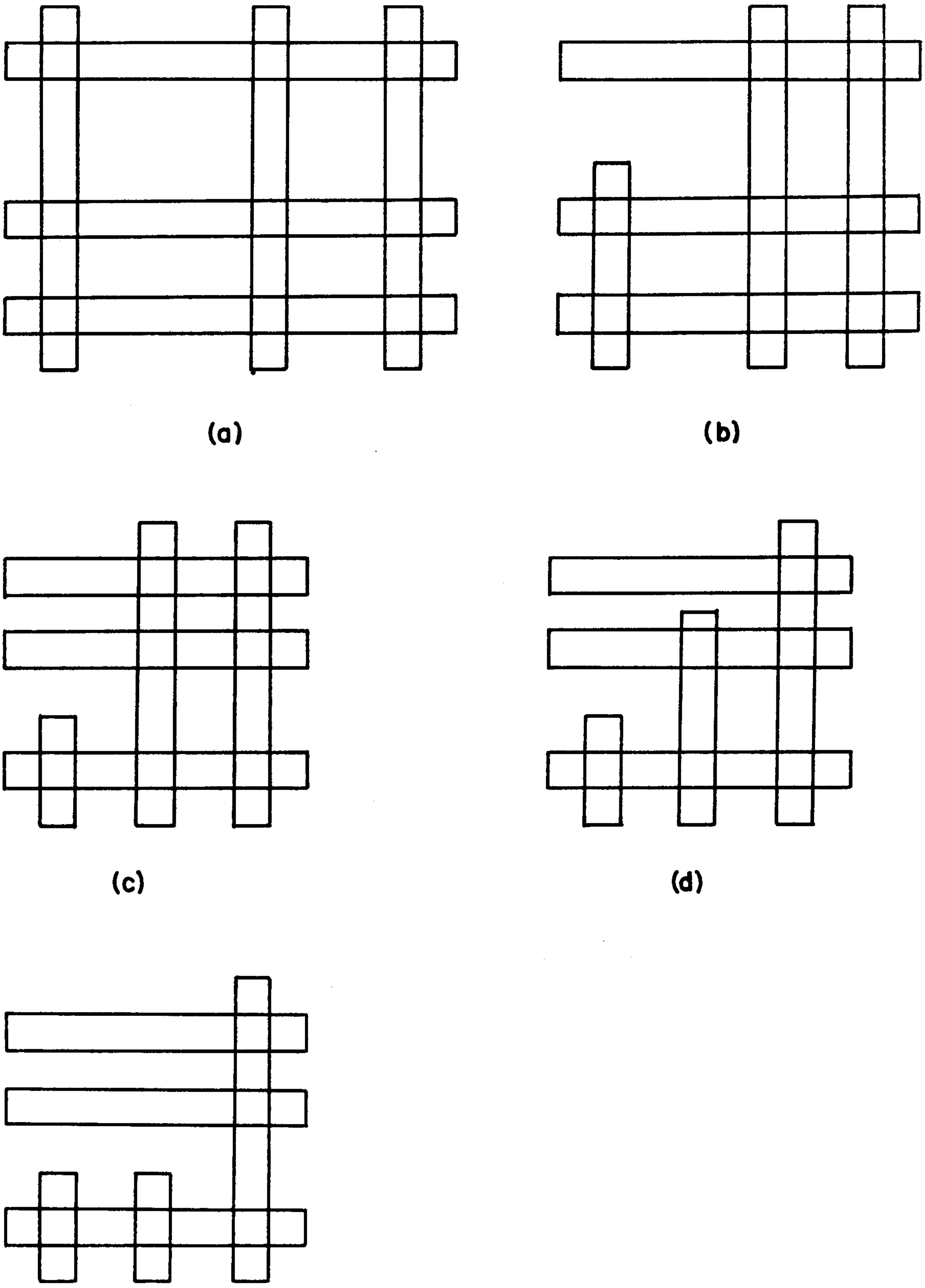

Figure 8: Achieving all possible numbers of holes by shrinking.

Lemma 2.7 *For any vertical or horizontal line l, let $I \subseteq \{0, \ldots, h\}$ be the set of indices of those polygons P_i (holes and exterior of $\mathcal{U}$) that intersect l, that is, $I = \{i | P_i \cap l \neq \emptyset\}$. Then $\sum_{i \in I} h_i \leq 16n - 4k$.*

Proof: In the given set $\mathcal{R}$ of n rectangles, line l intersects all rectangles in a subset $\mathcal{R}' \subseteq \mathcal{R}$. Let each rectangle in $\mathcal{R}'$ be cut into two pieces at l. Then we get two sets of rectangles: $\mathcal{R}_l$ to the left of l and $\mathcal{R}_r$ to the right of l. All edges of the exterior and of holes of $\mathcal{R}$ that have been intersected by l belong (partially) to the exterior of at least one of $\mathcal{R}_l$, $\mathcal{R}_r$. Let k, c, n with subscript l (r) be defined with respect to set $\mathcal{R}_l$ ($\mathcal{R}_r$). Then $k_l + k_r \geq k$, and $n_l + n_r \leq 2n$. By Corollary 2.6 (2.3) we get

$$\sum_{i \in I} h_i \leq c_l + c_r \leq 8n_l - 4k_l + 8n_r - 4k_r \leq 16n - 4k$$

□

Note that this bound is not tight in general, but it is sufficient to establish the runtime and space consumption arguments for the algorithms to be presented. The essence of Lemma 2.7 has been stated and proved already in [8]; we include it here for the sake of completeness.

3 The Hole Detection Problem

Let us present a lower bound for the hole detection problem first and then describe an optimal solution.

3.1 A Lower Bound

Lemma 3.1 *Given a set $\mathcal{R}$ of n rectangles in the plane, $\Omega(n \log n)$ time is required to determine whether their union $\mathcal{U}$ has a hole. This holds even if the rectangles are given in sorted order according to one of the coordinates.*

Proof: By a reduction from the element uniqueness problem for positive integers.

Given a set $X = \{x_1, \ldots, n_n\}$ of positive integers, define a set of $3n + 1$ rectangles in the plane as follows. Compute $x_{max} = \max\{x_i | 1 \leq i \leq n\}$ in time $O(n)$ by a simple linear scan. Then $\mathcal{R} = \{R_0, \ldots, R_{3n}\}$, where

$R_0 = (-\frac{1}{4}, \frac{1}{4}, -\frac{1}{4}, x_{max} + \frac{1}{4})$, and

for $i = 1, \ldots, n$,

$R_{3i-2} = (-\frac{1}{4}, i + \frac{1}{4}, x_i, x_i + \frac{1}{4})$,

$R_{3i-1} = (i - \frac{1}{4}, i + \frac{1}{4}, x_i - \frac{1}{2}, x_i + \frac{1}{4})$, and

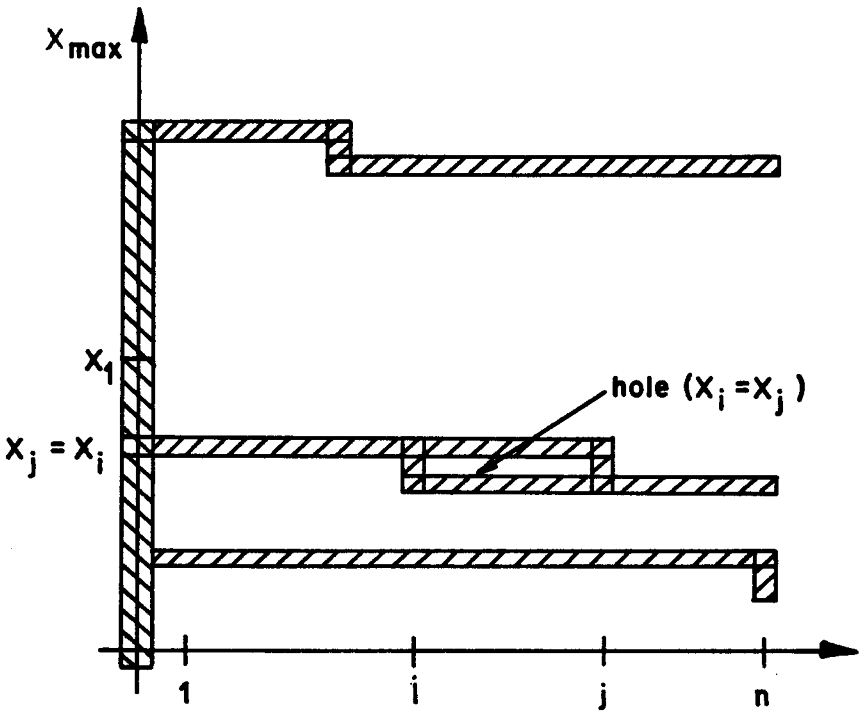

Figure 9: A lower bound for hole detection.

$R_{3i} = (i - \frac{1}{4}, n + \frac{1}{4}, x_i - \frac{1}{2}, x_i - \frac{1}{4})$.

For an illustration see Figure 9. Note that we have deliberately violated the assumption that the rectangles are in general position, in order not to obscure the basic idea of the reduction. The reduction can be changed slightly to shift the rectangles into general position. The set of rectangles has the following properties:

- all rectangles form one connected component;
- there is a hole in $\mathcal{U}$ if and only if the elements of X are not unique.

It is immediate from the construction that Lemma 3.1 holds even if the rectangles are given in sorted order according to one of the coordinates.

This completes the proof of Lemma 3.1. □

3.2 An Algorithm for Hole Detection

Our general approach is to sweep the plane from left to right with a vertical sweep line l. The line stops at each left and right vertical edge of a rectangle.

be determined that e_1 will not contribute to a hole. At sweep position x_3, it is still not known whether e_4 will contribute to a hole.

An uncovered area between two tentative inner edges is called a *tentative hole.* Any hole has to start in our plane sweep as a tentative hole. A tentative hole splits into several branches, when a left vertical edge of a rectangle falls entirely in between the two active edges of the hole. In our example, R_5's left vertical edge falls entirely in between e_3 and e_4. At a right vertical edge of a rectangle R_i, two areas meet that have been separated currently by R_i. If two tentative holes meet, it is recognized that they actually form one tentative hole. If a tentative hole meets the outside P_0 of $\mathcal{U}$, it is recognized that the tentative hole belongs to P_0 rather than forming a hole. A hole is detected only when a left vertical edge blocks all branches of a tentative hole.

Note that the tentative inner edges come in pairs of active edges with adjacent intersections with the sweep line. All of the above situations and transitions can be properly determined by keeping track of all pairs of tentative inner edges, of outer edges, and of hidden edges, in the way outlined above. Furthermore, in order to decide whether all branches of a tentative hole have been blocked, it is helpful to keep track of a set of branches for each tentative hole. Each tentative hole will start out with one branch. Later on, a branch may be added, whenever the tentative hole splits; a branch may be deleted when it ends; two tentative holes may be combined into one when two of their tentative branches meet; and, finally, a tentative hole is recognized to be a hole, when all of its branches end.

Then, more formally, the algorithm works as follows.

Algorithm *Detect Hole*
{Given a set $\mathcal{R}$ of rectangles, detect whether there is a hole in the contour of their union.}

```
1. Initialize all sweep-line data structures to be empty, and sort all x-
   values (vertical edges).

2. For all vertical edges $e$ in ascending $x$-order do:

   if $e$ is a left vertical edge, then
       terminate all branches of tentative holes in the $y$-range of $e$;
       if the set of branches for some tentative hole becomes empty,
              then report that a hole has been detected, and stop;
       update all (at most 2) branches that are intersected by (but not
       contained in) $e$'s $y$-range;
       if $e$ falls within one branch of a tentative hole,
              then split this branch into two;
```

```
        terminate all contour edges that lie within $e$'s $y$-range;
        add the top and bottom edges of $e$'s rectangle to the set of active
        edges, and to the contour if they lie outside the area covered by the
        rectangles so far;
    else {$e$ is a right vertical edge}
        if there are no edges in $e$'s $y$-range, to the right of $e$, that
        belong to the contour, then
                combine the two areas adjacent to $e$ above and below into one,
                by creating a new tentative hole from two old ones, by
                combining all branches, or by adding all branches of a tentative
                hole to the contour,
        else {there are contour or hole edges to the right of $e$}
                update the contour or holes above and below $e$ accordingly;
                start a new hole for every pair of horizontal edges in $e$'s
                $y$-range that have uncovered area between them.

  3. Report that there is no hole.
end
```

3.3 An Efficient Implementation

In essence, to detect the existence of a hole in the union of a set of rectangles, the algorithm computes the contour implicitly and additionally keeps track of "topological" information about candidate holes. An optimal sweep-line algorithm for computing the contour, see [8], uses as the central data structure the visibility tree; see [9].

For any given sweep line position, the active horizontal edges are represented by points on the sweep line. Some of these points are visible, others may be hidden. When a vertical edge is encountered, points may change their status from visible to hidden (at a left edge) or from hidden to visible (at a right edge). Also, points representing horizontal edges have to be inserted into or deleted from the visibility tree. The visibility tree allows for the insertion or deletion of a point representing a horizontal edge in time $O(\log n)$, where n is the number of points to be stored in the tree. For a given interval representing a vertical edge of a rectangle, the visibility tree enables the visibility status of all affected edges to be updated in time $O(\log n)$, and it enables these edges to be reported in constant time per edge. Furthermore, it uses only linear space, because each active point is stored only once.

The central data structure in our implementation of the hole detection algorithm is the visibility tree. During the line sweep, we maintain a visibility tree, together with information on the holes to which the visible points belong. To this end, we store the set of active visible edges belonging to a tentative hole in a doubly linked circular list, with one list item for each edge. This is done for each hole, and also for the outer contour. From each active visible point in the visibility tree, there is a pointer to the corresponding list item. All updates of the lists are performed by following pointers from the visibility tree to list items. Actions need only be taken when edges become visible or become hidden. A starting hole or the starting contour lead to a new list containing two items, one for each horizontal edge bounding the hole or contour. When a branch of a hole ends, the two corresponding items are removed from their list. Splitting a hole into two implies that two edges are added to the corresponding list; forming the union of two holes is realized by linking two circular lists (giving one circular list). Note that access to the circular lists is provided through the visibility tree by pointers to list items. As soon as a list becomes empty for a tentative hole and it was not the only one in existence, a hole has been detected. With this implementation, we get:

Theorem 3.2 *Given a set of n rectangles in the plane, it can be determined, in $O(n \log n)$ time and with $O(n)$ space, whether their union has a hole.*

Proof: The presented algorithm correctly determines whether there is a hole by sweeping a line from left to right until the right boundary of a hole is first encountered. Therefore, no hole lies entirely to the left of the current sweep line position. With Lemma 2.7, the number of edges of all started holes and the contour is therefore $O(n)$. Hence, $O(n)$ operations have been performed on the visibility tree, each at a cost of $O(\log n)$ time. Altogether, $O(n)$ edge pieces have changed their visibility status, with each of the involved updates of the associated lists performed in constant time. This establishes the claimed time bound. The space requirement follows from that of the visibility tree. □

4 Computing all holes

A lower bound for computing some constant size information about each hole, say, a point within each hole, follows immediately from the lower bound for hole detection:

Corollary 4.1 *Given a set of n rectangles in the plane, $\Omega(n \log n + h)$ time is required to compute a point within each hole in their union, where h is the number of holes.*

Proof: Follows from Lemma 3.1 and the fact that h is the number of points to be reported. □

An algorithm to report a point inside each hole can be derived simply from the algorithm presented in Section 3 to detect the existence of a hole. Instead of halting after the end of a hole has been detected, we report a point just to the left of the edge terminating a hole, and continue doing so for each hole until all edges have been processed. Thus, we obtain:

Theorem 4.2 *Given a set of n rectangles in the plane, a point within each hole in their union can be computed in time $O(n \log n + h)$ and space $O(n)$, where h is the number of holes.*

Proof: The number of operations to be carried out on the visibility tree is $O(n)$, for the $2n$ left and right vertical edges of rectangles, at a cost of $O(\log n)$ time each. For every change in the visibility status of (a piece of) an edge, constant additional cost to update the lists suffices. From Lemma 2.4 we know that there are $O(n + h)$ of these changes, establishing the claimed time bound. The space bound follows from that of the visibility tree. □

5 Concluding Remarks

In this paper we have proved some basic combinatorial results about the number of holes in the union of a set of rectangles. We have also provided time– and space–optimal algorithms to decide whether there is a hole and to compute a point withir each hole.

Clearly, these questions can be posed in other frameworks, for example, what happens in d-d'.iensions, what happens with other shapes, for example, squares and circles, and what about the searching problem — given n rectangles and a query point that lies in a hole, provide the hole's contour. These questions and related ones are currently under investigation, some initial results are reported in [5].

Initially, the reason we investigated holes was to provide an alternative algorithm to solve the contour problem for a set of restricted-oriented polygons (see [6] for an introduction to restricted-oriented geometry). Of course, using the general algorithm of [4] we can solve this problem in $O(n \log n + k \log n)$ time, where the polygons have a total of n edges and there are k edge intersections. Because restricted-oriented polygons appear to be closer in their behavior to orthogonal polygons we expected to obtain an $O(n \log n + p)$ time algorithm, where p is the number of edges in their union. Unfortunately, so far we have been unable to find such an algorithm. For this reason, we considered a different approach stemming from the following observation.

> Given a point in a hole, we can extend it vertically upwards until it first hits an edge. This must contain an edge of the contour of the hole. Follow the edge to the right, say, until a second edge is met. This must also belong to the contour. We continue walking around the edges bordering the hole, in this manner, until we return to the starting edge.

To implement this algorithm for orthogonal polygons we can use a simplified form of the layered segment tree of [7]. Each search for an edge requires $O(\log n)$ time, therefore this leads to an $O(n \log n + p \log n)$ time algorithm for the contour. This is worse than the best algorithms known for this problem, see [2,9], since they only require $O(n \log n + p)$ time. However, this technique can be applied to the restricted-oriented case, with the same time bounds, and, in this case, it is better than the general algorithm. At present however, we are unable to find points in the holes within the same time bound so a contour algorithm for restricted-oriented polygons with this time complexity is still elusive.

References

[1] H. Edelsbrunner, J. van Leeuwen, Th. Ottmann, and D. Wood. Computing the connected components of simple rectilinear geometrical objects in d-space. *RAIRO Informatique théoretique*, 18:171–183, 1984.

[2] R.H. Güting. An optimal contour algorithm for iso-oriented rectangles. *Journal of Algorithms*, 5:303–326, 1984.

[3] W. Lipski and F.P. Preparata. Finding the contour of the union of a set of iso-oriented rectangles. *Journal of Algorithms*, 1:235–246, 1980.

[4] Th. Ottmann, P. Widmayer, and D. Wood. A fast algorithm for the Boolean masking problem. *Computer Vision, Graphics, and Image Processing*, 30:249–268, 1985.

[5] G.J.E. Rawlins, P. Widmayer, and D. Wood. *Circles, Squares, and Holes.* Technical Report, University of Waterloo, Department of Computer Science, 1986.

[6] G.J.E. Rawlins and D. Wood. *Restricted-Orientation Convexity.* Technical Report, University of Waterloo, Department of Computer Science, 1986.

[7] V.K. Vaishanavi and D. Wood. Rectilinear line segment intersection, layered segment trees, and dynamization. *Journal of Algorithms*, 3:160–176, 1982.

[8] P. Widmayer and D. Wood. *Time- and Space-Optimal Contour Computation for a Set of Rectangles.* Technical Report, University of Waterloo, Department of Computer Science, 1986.

[9] D. Wood. The contour problem for rectilinear polygons. *Information Processing Letters*, 19:229–236, 1984.

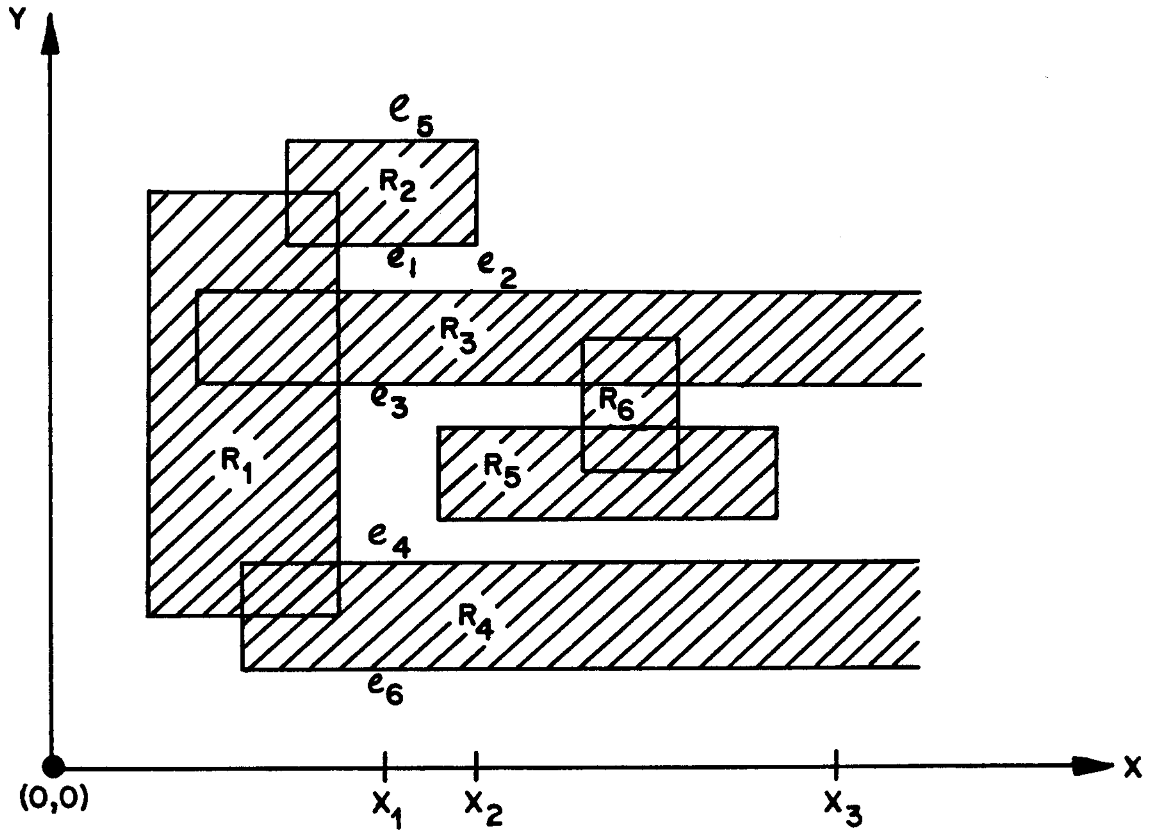

Figure 10: Hole detection.

A data structure L keeps track of the horizontal edges of rectangles currently intersected by the sweep line; the *active* edges. Some of these edges separate a covered area from an area that is not covered, that is, they are on the boundary of $\mathcal{U}$. Whenever l intersects at least one rectangle there will be at least two edges that partially belong to the outside boundary of $\mathcal{U}$, that is, P_0, and there may be edges that partially belong to holes P_i, $i > 0$. All of these edges are said to be *visible*. In addition, there may be active edges that do not currently belong to the boundary of $\mathcal{U}$; these are said to be *hidden*. Of the visible edges, some are known to belong to the outside boundary of $\mathcal{U}$, namely, at least the topmost and the bottommost ones; these edges are called *outer* edges. There may be other visible active edges that may or may not belong to a hole, as far as this can be judged from the current state of the plane sweep; these edges are called *tentative inner* edges with respect to the state of the plane sweep.

For an illustration of these concepts, see Figure 10. At sweep position x_1, e_1, e_2, e_3, and e_4 are all tentative inner edges, whereas e_5 and e_6 are outer edges. At sweep position x_2, e_1 becomes an outer edge, because it can