CONGRESSUS NUMERANTIUM

VOLUME 46

MAY, 1985

WINNIPEG, CANADA

A SIMPLE SPACE-OPTIMAL CONTOUR ALGORITHM
FOR A SET OF ISO-RECTANGLES

by

Przemyslaw Prusinkiewicz

Vijay V. Raghavan

Department of Computer Science
University of Regina
Regina, Saskatchewan S4S 0A2
CANADA

ABSTRACT

A new algorithm for finding the contour of a set of iso-rectangles is developed. The algorithm requires $O(n^2)$ time and $O(n)$ space. This resolves the question, raised by Güting [4], of whether there exists an $O(n)$ space algorithm which reports the pieces of the contour in the order of contour-cycles. The algorithm uses a data structure that facilitates a clear separation of the geometrical and topological aspect of the contour problem. A notion of topological equivalence is introduced and applied for avoiding repetitive computations for sets of iso-rectangles belonging to the same equivalence class. A modified definition of a slab is introduced for handling special cases (induced by colinear edges) in a consistent way.

1. INTRODUCTION

A rectangle with two edges parallel to the x-axis and two - to the y axis is called an iso-rectangle. The contour of a set of iso-rectangles is the boundary of the their union. In general this is a collection of disjoint "contour-cycles", each of which is a sequence of alternating horizontal and vertical contour-pieces.

Iso-rectangles play an important role in several practical areas such as computer graphics [9,11], VLSI design [7], and architectural data bases [1]. They also attract considerable theoretical interest (see [2,3] for extensive bibliographies).

This research is supported by a grant from Natural Sciences and Engineering Research Council of Canada.

CONGRESSUS NUMERANTIUM, 46 (1985), pp. 249-270

The problem of finding the contour of a set of iso-rectangles was first mentioned by Shamos [10, p.163], and explicitly stated by Vitanyi and Wood [12]. Vitanyi and Wood remarked that their algorithm for computing the perimeter could be modified for reporting the contour, too. However, they did not pursue this idea.

The first complete solution of the contour problem is due to Lipski and Preparata [6]. Two other, time-optimal algorithms were published by Güting [4,5]. Yet another algorithm was found by the authors, looking for an intuitively simple solution of the contour problem [8]. Time and space complexity of these algorithms are summarized in Table 1.

The algorithm described in this paper requires $O(n^2)$ time and $O(n)$ space. The space requirement is optimal. Like the algorithms of Lipski and Preparata [6], and Güting [4], our

Table 1. Time and space requirements of known algorithms for finding the contour of a set of iso-rectangles.

Algorithm	Time	Space
Vitanyi & Wood [2]	$O(n^2)$	$O(n\log n)$?
Lipski & Preparata [6]	$O(n\log n + p\log(2n^2/p))$	$O(n+p)$
Guting [4]	$O(n\log n+p)$	$O(n+p)$
Guting [5]	$O(n\log n+p)$	$O(n\log n+p)$
Prusinkiewicz & Raghavan [8]	$O(n^2)$	$O(n^2)$

n - number of given rectangles
p - number of edges in the contour

algorithm is based on the line-sweep approach. However, instead of a segment tree or its modifications, it uses a multilinked list of edges as its fundamental data structure. This data structure is the key to the low space requirement. Due to the use of pointers, it is not necessary to store all edges of the contour before reporting them. Loosely speaking, this eliminates term p in the formula $O(n+p)$, describing the space complexity of some previous algorithms (Tab.1).

The paper is organized as follows.

In section 2 the main idea of the algorithm is given. The presentation is made under the assumption that no two edges of the input iso-rectangles are colinear. This assumption is removed in section 3 through modifying the notion of a slab. Since this modification makes the slabs lose their common intuitive interpretation, appropriate formal definitions are provided. In section 4 the notion of topological equivalence between sets of iso-rectangles is introduced. It is shown that the contour of a set of iso-rectangles can be mapped onto the contour of any equivalent set, thus saving computation. Finally, in section 5 analysis of the time and space complexity of the contour algorithm is given.

2. THE ALGORITHM

2.1 Intuition

The algorithm is based on the use of slabs.

Suppose that vertical lines are drawn through vertices of all given rectangles. Since the rectangles are iso-oriented, these lines can also be seen as infinite extensions of vertical edges. Each pair of consecutive lines defines a vertical strip, called a slab. Horizontal edges of rectangles which intersect with a slab are said to be active. Within a slab an active edge will contribute to the overall contour of the union, if it separates an area covered by one or more rectangles from a non-covered area. Such an edge is called relevant to the slab. Thus, an active edge of a rectangle is relevant if and only if it is not contained in any other rectangle within the slab. For example, consider the iso-rectangles shown in Fig.1a. In slab 4 horizontal edges of rectangles 2 and 3 are active and relevant, while in slab 5 horizontal edges of rectangles 2,3 and 4 are active, but only the bottom edge of rectangle 4 and the top edge of rectangle 3 are relevant (Fig. 1b).

Vertical segments of the contour lie on the lines limiting

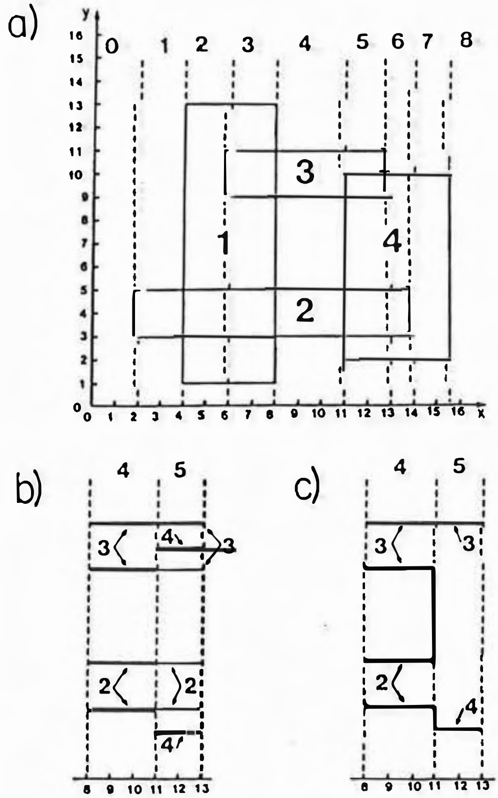


Fig. 1. a) An example of layout of iso-rectangles.
b) Active (——) and relevant (——) edges in slabs 4 and 5.
c) Tying together relevant edges.

the slabs. They tie together relevant horizontal edges of two consecutive slabs. As the contour line cannot self-intersect, the horizontal edges can be tied together in only one way: the lowest one with the second from the bottom, the third with the fourth, and so on.

A vertical segment degenerates to one point if horizontal segments to be tied together are colinear. In particular, this situation arises if an edge is relevant to two or more consecutive slabs. When found, a sequence of colinear contour segments should be replaced by one segment of appropriate length.

In the example of slabs 4 and 5 shown in Fig.1b, vertical segments will connect:

- the bottom edge of rectangle 4 in slab 5 with the bottom edge of rectangle 2 in slab 4;
- the top edge of rectangle 2 in slab 4 with the bottom edge of rectangle 3 in the same slab, and
- the pieces of the top edge of rectangle 3, lying in slabs 4 and 5.

In the last case the connected segments are colinear and should be replaced by one longer segment. The result of these connections is shown in Fig. 1c.

The final contour is a sequence of alternating horizontal pieces of relevant edges and vertical segments connecting them pairwise.

2.2. Input data

Input data consist of the descriptions of a given set of iso-rectangles. Each rectangle is specified as a 5-tuple:

$$R_k=(k,\ x_{lk},\ x_{rk},\ y_{bk},\ y_{tk})$$

where:

- k is an ordering number of the rectangle,
- x_{lk} and x_{rk} are abscissas of its left and right edges,
- y_{bk} and y_{tk} are ordinates of the bottom and the top edge respectively.

For instance, the set of rectangles shown in Fig. 1a is described as follows:

(1,	4,	8,	1,	13)
(2,	2,	14,	3,	5)
(3,	6,	13,	9,	11)
(4,	11,	15.5,	2,	10)

For the sake of clarity, in the description of the algorithm we will assume that no two edges of different rectangles are colinear. This restriction will be removed in section 3.

2.3 Internal representation of edges and contour-cycles

Edges of iso-rectangles can be specified in several ways. We will specify them as pairs (k,s), where k is the number of a rectangle, and the tag $s \in \{\leftarrow, \uparrow, \rightarrow, \downarrow\}$ indicates its bottom, left, top and right edge , respectively. Obviously, given an edge (k,s), the coordinates of its endpoints can be found in a straightforward way by referring to the 5-tuple $R_k = (k, x_{lk}, x_{rk}, y_{bk}, y_{tk})$.

Contour-cycles are represented by circular lists of pairs (k,s), specifying consecutive edges of iso-rectangles which contribute to the overall cycle. By convention, edges of each cycle are listed such that the figure is on the right side while traversing consecutive edges. In other words, a contour-cycle is oriented clockwise if it is an external boundary of a connected component, and counterclockwise - if it is the boundary of a hole. For instance, the interior contour-cycle of the set shown in Fig.1a is specified by the circular list of elements $(2,\rightarrow)$, $(4,\uparrow)$, $(3,\leftarrow)$, $(1,\downarrow)$. Observe that this representation is unambiguous - the intersections of consecutive edges define unambiguously consecutive vertices of the contour-cycle.

The use of pairs in the form (k,s) instead of explicit coordinates helps separate the topology of the contour problem from its geometry. This will be discussed in more detail in section 4.

2.4 Output data

The algorithm reports the edges of the contour immediately after they are determined. In consequence, edges belonging to different cycles may interlace in the order of presentation. Sequences of edges forming cycles are specified by means of pointers, indicating explicitly the next edge of the cycle. For example, see the output of the algorithm for the set of rectangles from Fig.1, given in Table 2 and Fig.2.

Table 2. Output of the algorithm for the set of rectangles from Fig.1a. Assignment of numbers to edges is arbitrary and depends on details of implementation of the algorithm (cf. Fig.5).

Edge Number	Edge Specification	Next Edge	Edge Number	Edge Specification	Next Edge
1	(2,←)	2	7	(1,→)	13
2	(2,↑)	3	13	(1,↓)	14
4	(1,←)	5	15	(4,←)	16
5	(1,↑)	1	16	(4,↑)	8
3	(2,→)	6	10	(2,→)	17
6	(1,↑)	7	17	(4,↑)	12
8	(2,←)	9	14	(3,→)	18
9	(1,↓)	4	18	(3,↓)	19
12	(3,←)	11	19	(4,→)	20
11	(1,↓)	10	20	(4,↓)	15

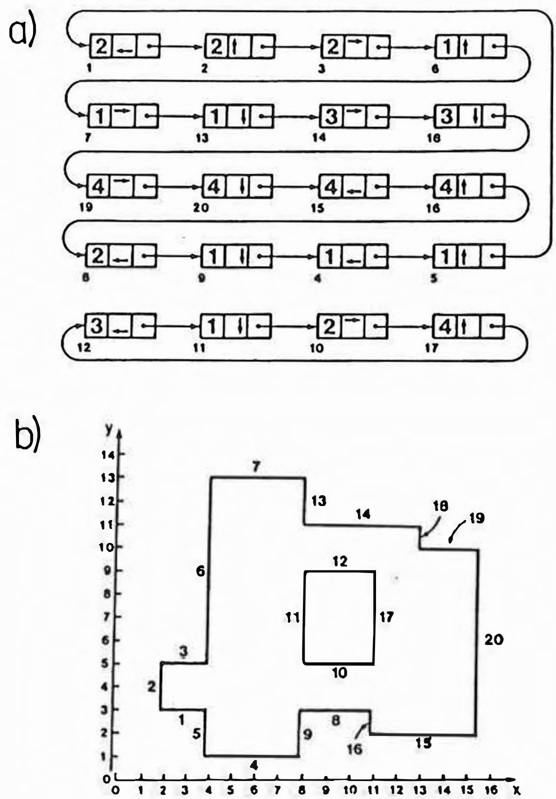


Fig. 2. Output of the algorithm (Tab. 2) interpreted as: a) list of edges, and b) plot of the contour. Edge numbers correspond to Tab. 2.

2.5 Preprocessing

The purpose of preprocessing is to establish a list of consecutive slabs. Each slab is specified by a vertical edge of a rectangle, colinear with the left edge of the slab. For instance, slab 4 in Fig. 1a is specified as (1,) and slab 5 as (4,). Obviously, finding the list of consecutive slabs is equivalent to sorting the set of vertical edges of the given rectangles by their abscissas.

2.6. Finding active edges in a slab

In this step, the bottom-up list of active edges in a slab i is computed recursively as follows:

1) Initially, in slab 0 (preceding the leftmost edge of a given set of iso-rectangles) the list of active edges is empty.
2) Transition from a slab i to the slab i+1 implies either one of the actions:
   - insertion of new active edges (k,$\leftarrow$) and (k,$\rightarrow$) - if the slab i+1 is specified by the left edge of the rectangle k: (k,$\uparrow$), or
   - deletion of edges (k,$\leftarrow$) and (k,$\rightarrow$) which are no longer active - if the slab i+1 is specified by the right edge of the rectangle k: (k,$\downarrow$).

The edges must be inserted into their proper places to keep the list of active edges in the bottom-up order (see. Fig.3).

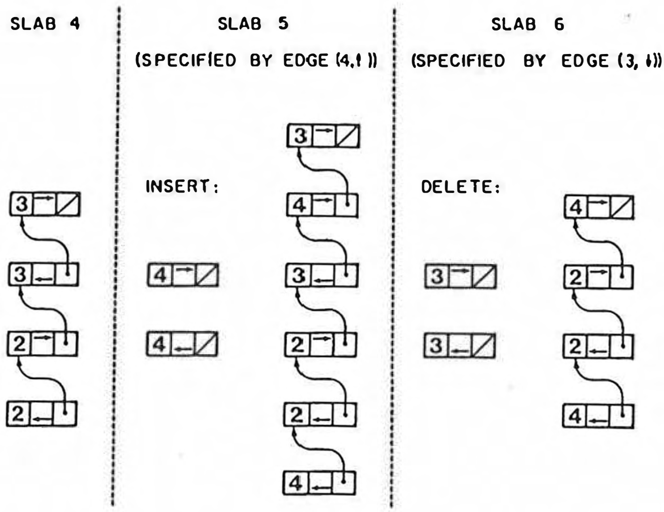


Fig. 3. Finding active edges in slabs 5 and 6 from Fig. 1a.

2.7. Finding relevant edges in a slab

Consecutive active edges divide a slab into segments. The number of rectangles including a given segment is called the invisibility order [cf. 9,11] and is denoted by I. The invisibility order of each segment of a slab is computed recursively, by scanning the slab bottom-up.

1) Initially (i.e. below the lowest active edge) I=O.

2) - Traversing of a bottom active edge $(k,\leftarrow)$ of a rectangle increases I by 1;

- Traversing of a top active edge $(k,\rightarrow)$ decreases I by 1.

After the invisibility orders are known, the relevant edges can be easily selected, since they separate segments with I=O from segments with I=1 (or segments with I=1 from segments with I=O). For example, see Fig.4.

2.8. Forming contour-cycles

Contour cycles are represented as circular lists of horizontal relevant edges connected by appropriate vertical edges. Computation of contour-cycles poses the following problems:

- Identification of pairs of relevant edges to be connected.
- Detection of situations where two edges to be connected are, in fact, different segments of the same edge split among two or more consecutive slabs. These segments should be replaced by one segment of the appropriate length.
- Creation of vertical segments which will connect remaining pairs of relevant edges.

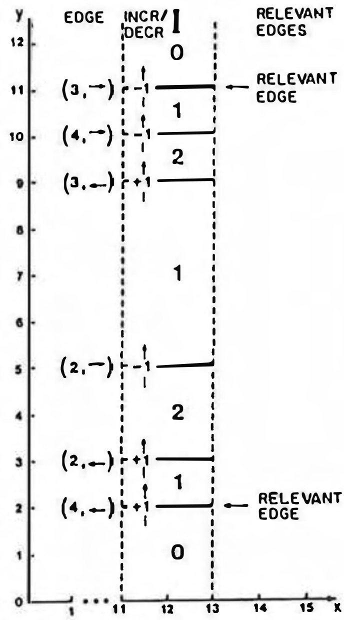


Fig. 4. Calculating invisibility orders and finding relevant edges in slab 5 from Fig. 1a.

- Determination of the order of the edges within a contour-cycle.

The algorithm forms contour-cycles by systematic examination of pairs of consecutive slabs. Given slabs i-1 and i, this involves the following steps:

1) Find the two lowest relevant edges, not previously considered, in the merged slabs i-1 and i.
2) If the two edges are colinear (i.e. they represent two different segments of the same edge (k,s)), replace them

by one longer segment running through two or more slabs.

3) If the two horizontal edges to be connected are not colinear, join them by the vertical edge specifying slab i. Determine the order in which the three edges will be listed as parts of a contour-cycle using the reasoning below:

An edge of a union of iso-rectangles is an edge of some rectangle or a segment of it. Thus, the union must lie on the same side of this edge as the rectangle. Suppose that edges of input iso-rectangles are vectors oriented such that given a vector, the rectangle is on its right side. Then the edges of each contour-cycle must be listed in such an order, that all edges are scanned conforming to their orientation when the cycle is traversed (cf. section 2.3). This provides the criterion for edge linking. For instance, consider Fig.1a and 1c. The edges: $(4,\leftarrow)$ in slab 5 and $(2,\leftarrow)$ in slab 4 are oriented from right to left, and the edge $(4,\uparrow)$ specifying slab 5 is oriented bottom-up. Hence, in a contour-cycle these three edges must be listed in the order $(4,\leftarrow)$ - $(4,\uparrow)$ - $(2,\leftarrow)$ and not: $(2,\leftarrow)$ - $(4,\uparrow)$ - $(4,\leftarrow)$. Likewise, the edges $(2,\rightarrow)$, $(3,\rightarrow)$ and $(4,\uparrow)$ from the same slabs 4 and 5 must be ordered $(2,\rightarrow)$ - $(4,\uparrow)$ - $(3,\rightarrow)$ and not $(3,\rightarrow)$ - $(4,\uparrow)$ - $(2,\rightarrow)$.

4) Repeat steps 1 through 3 until all relevant edges in slabs i-1 and i are considered.

2.9 Overall structure of the algorithm

A complete algorithm for reporting the contour of a set of iso-rectangles is composed of the following steps:

1. Preprocessing: Given n iso-rectangles, establish the ordered list of 2n+1 slabs numbered from 0 (the leftmost slab) to 2n (the rightmost one).
2. Set initial conditions: Create the empty list of active edges in slab 0.

3. Main body of the algorithm: For i=1 until 2n do:

{

- Find the list of active edges in slab i;
- Find the list of relevant edges in slab i;
- For all pairs of relevant edges in merged slabs i-1 and i, considered in the bottom-up order, do:

  If both edges are colinear

  then replace them by one segment of appropriate length

  else

  {

  - Connect the edges with the vertical segment specifying slab i. Observe their order within the contour cycle they belong to;
  - Report the edges to be added to the contour along with pointers establishing their order.

  }

}

2.10. An example of implementation

A straightforward implementation of the algorithm is based on the use of linked lists. The following lists are maintained:

- consecutive vertical edges of input iso-rectangles,
- active edges in the current slab (slab i),
- relevant edges in the current slab (slab i),
- relevant edges in the previous slab (slab i-1).

Relevant edges are represented by atoms with two pointers. One pointer indicates the next relevant edge within a slab. Another pointer is used to link consecutive contour-edges within a contour-cycle. This data structure is exemplified in Fig. 5.

3. COLINEAR EDGES

3.1 General discussion

In section 2.2 we assumed that no two edges of the input iso-rectangles were colinear. Hence, it was possible to linearly order all vertical edges in ascending sequence of their abscissae and the horizontal edges - in ascending sequence of their ordinates. Both orderings are important to the algorithm. The ordering of vertical edges has two implications:

- slabs can be defined as strips (of non-zero width) between extensions of consecutive edges (cf. section 2.1);
- transition from a slab i to the slab i+1 involves either insertion or deletion of exactly two active edges (cf. section 2.6).

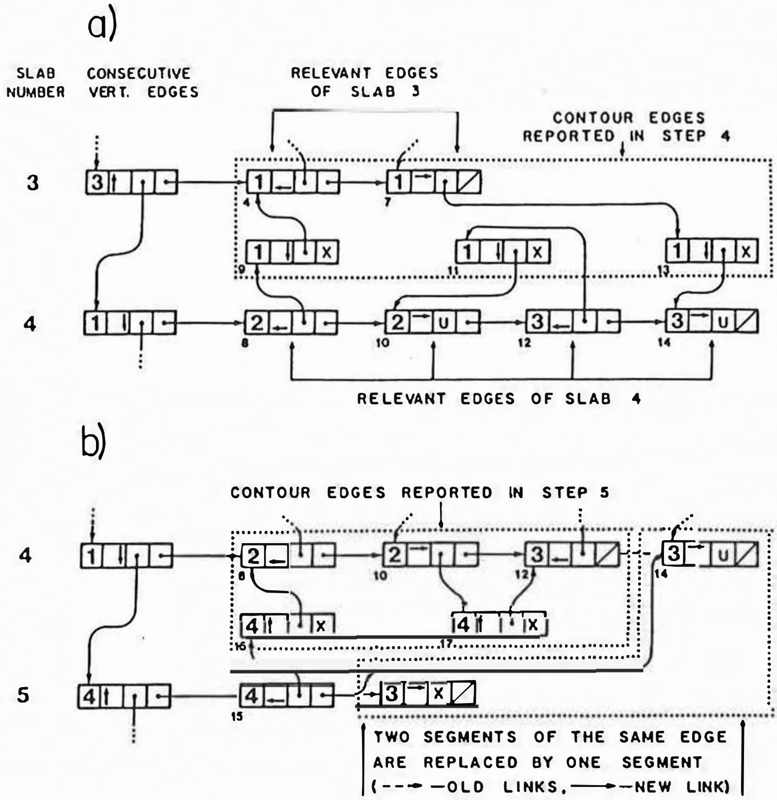


Fig. 5. Finding the contour of the set of iso-rectangles of Fig. 1a. a) Data structure after step i=4. b) Data structure after step i=5. Lists of active edges not shown - cf. Fig. 3. x - link irrelevant. u - link not determined yet. Contour edges are identified by the same numbers as shown in Tab. 2 and Fig. 2. These numbers are assigned to edges while they are connected into cycles, in the order of consideration.

The ordering of horizontal edges has other implications:

- consecutive active edges divide a slab into segments (cf. section 2.7);

- transition between consecutive segments of a slab either increments or decrements the order of invisibility I by exactly one (cf. section 2.7);
- relevant edges to be connected within a contour cycle (the two lowest edges not yet considered) can be determined unambiguously (cf. section 2.8).

The assumption of non-colinearity of edges can be removed using one of two approaches:

1) by modifying the algorithm to handle edges which are not linearly ordered;
2) by replacing increasing values of abcissae or ordinates with another ordering relation which induces an appropriate linear ordering of edges, even if they are colinear.

In this paper the second approach is employed. Intuitively, the main idea is that even colinear vertical edges define separate slabs (perhaps of zero width). Likewise, all active edges within a slab induce separate slab segments (perhaps of zero height).

Colinear edges must be ordered in such a way that:

α) left edges ($\uparrow$) precede right edges ($\downarrow$), and

β) bottom edges ($\leftarrow$) precede top edges ($\rightarrow$).

This will ensure that iso-rectangles which share only an edge or a vertex will be correctly treated as connected components (cf. Fig.6).

The ordering of colinear edges of the same type (for instance,$\uparrow$) is less important. This ordering may affect the way in which the contour is represented, but does not affect the contour itself. One possibility is to order colinear edges of the same type by their rectangle's number.

The use of relations imposing a linear order on any set of horizontal or vertical edges makes it possible to remove the restriction of non-colinearity of edges without essentially modifying the algorithm given in section 2. However, the notions of a slab, an active edge, and a relevant edge lose their straightforward geometrical interpretations (cf. section 2.1) and require new definitions.

3.2 Formal definitions

An iso-rectangle in the plane is defined as a 5-tuple $R_k = (k, x_{lk}, x_{rk}, y_{bk}, y_{tk})$, where: k is the number of the rectangle, x_{lk}, x_{rk} are abscissae of its left and right edges, and y_{bk}, y_{tk} are ordinates of its bottom and top edges respectively (cf. section 2.2). A set S of n iso-rectangles $R_1,\ldots,R_n$ can be alternatively specified by a function:

$$\varphi: \{1,\ldots,n\} \times \{\vdash, \dashv, \perp, \top\} \to (-\infty,+\infty)$$

defined for all $k \in \{1,\ldots,n\}$ as follows:

$$\varphi(k,\vdash) = x_{lk} \qquad \varphi(k,\perp) = y_{bk}$$

$$\varphi(k,\dashv) = x_{rk} \qquad \varphi(k,\top) = y_{tk}$$

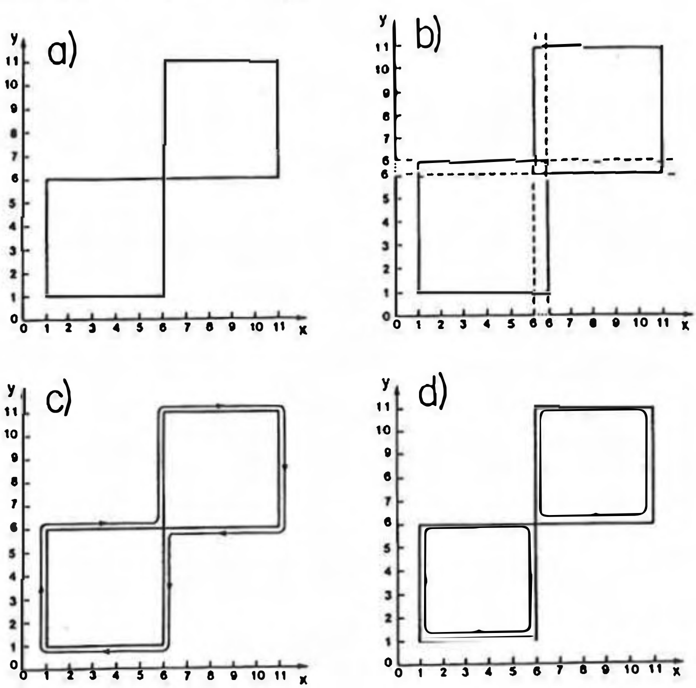


Fig. 6. Example of orderings of colinear edges. a) Two rectangles with colinear edges. b) Ordering of colinear edges following assumptions α and β . c) Contour resulting from this ordering - the rectangles are properly interpreted as being connected. d) Ordering following assumptions opposite to α and β results in an improper contour - the rectangles are interpreted as unconnected.

In the set of vertical edges $X = \{1,\dots,n\} \times \{\uparrow, \downarrow\}$ we define an ordering relations $\prec_x$ as follows:

$\varphi(k_1,v_1) < \varphi(k_2,v_2) \Rightarrow (k_1,v_1) \prec_x (k_2,v_2)$

$\varphi(k_1,\uparrow) = \varphi(k_2,\downarrow) \Rightarrow (k_1,\uparrow) \prec_x (k_2,\downarrow)$

$\varphi(k_1,v) = \varphi(k_2,v)$ & $k_1<k_2 \Rightarrow (k_1,v) \prec_x (k_2,v)$

where $k_1,k_2 \in \{1,\dots,n\}$; $v_1,v_2,v \in \{\uparrow,\downarrow\}$. Likewise, in the set of horizontal edges $Y = \{1,\dots,n\} \times \{\leftarrow,\rightarrow\}$ we define an ordering relation $\prec_y$:

$\varphi(k_1,h_1) < \varphi(k_2,h_2) \Rightarrow (k_1,h_1) \prec_y (k_2,h_2)$

$\varphi(k_1,\leftarrow) = \varphi(k_2,\rightarrow) \Rightarrow (k_1,\leftarrow) \prec_y (k_2,\rightarrow)$

$\varphi(k_1,h) = \varphi(k_2,h)$ & $k_1<k_2 \Rightarrow (k_1,h) \prec_y (k_2,h)$

where $k_1,k_2 \in \{1,\dots,n\}$; $h_1,h_2,h \in \{\leftarrow,\rightarrow\}$.

Let f: $X \to \{1,\dots,2n\}$ assign an ordering number (called the slab number) to each element of the list X sorted by the relation $\prec_x$. A horizontal edge (k,h) is said to be active in slab $i \in \{1,\dots,2n\}$ if and only if $f((k,\uparrow)) \le i$ and $f((k,\downarrow)) > i$. A list of all active edges in a slab i is denoted by Y_i. Let g: $Y_i \to \{1,\dots,2n_i\}$ assign an ordering number (called the segment number) to each element of the list Y_i, sorted by the relation $\prec_y$. The invisibility order $I(j)$ of a segment $j \in \{0,1,\dots,2n_i\}$ is defined recursively as follows:

$$I(0) = 0$$

$$I(j) = \begin{cases} I(j-1)+1 & \text{if } g^{-1}(j)=(k,\leftarrow) \\ I(j-1)-1 & \text{if } g^{-1}(j)=(k,\rightarrow) \end{cases} \qquad j=1,\dots,2n_i$$

An edge j is relevant if $I(j-1)=0$ or $I(j)=0$. The notions introduced above are illustrated in Fig.7. Notice, however, that although in Fig.7b colinear edges are split (e.g. $(1,\rightarrow)$ and $(2,\rightarrow)$), they should be considered as colinear while forming contour cycles (section 2.8).

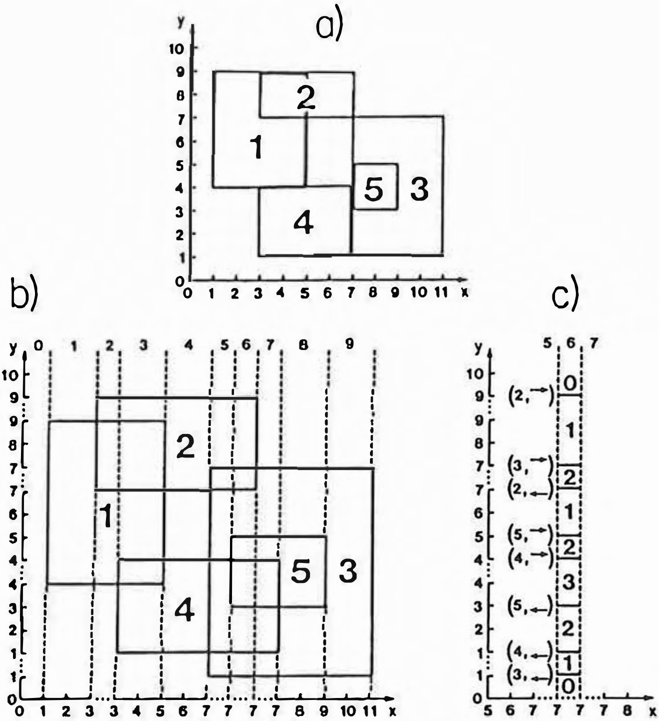


Fig. 7. Using relations $\prec_x$ and $\prec_y$. a) An example of layout of iso-rectangles with colinear edges. b) Intuition of slabs, active edges and slab segments. c) Invisibility orders in slab 6.

4. TOPOLOGY AND GEOMETRY OF THE CONTOUR PROBLEM

The described algorithm refers to the coordinates of input rectangles twice:

- when lists of edges (expressed as pairs (k,s)), ordered by the relations $\prec_x$ and $\prec_y$ are established, and
- when colinearity of contour-edges is checked.

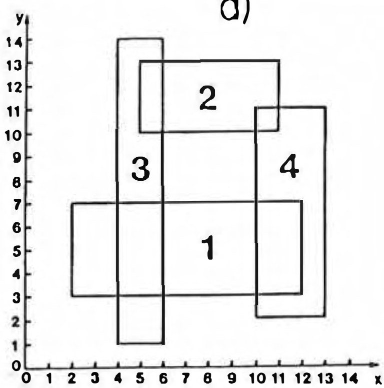
Therefore, any two sets of iso-rectangles which:

- exhibit the same ordering of edges, and
- have the same pairs of colinear edges,

will have identical solutions expressed in terms of lists of the (k,s) pairs. Such sets of iso-rectangles will be called topologically equivalent. The notion of topological equivalence can obviously be extended to situations, where corresponding rectangles from the sets under consideration (denoted by Γ and Δ)

have different numbers and are related to each other by a bijection $\varkappa : \Gamma \longleftrightarrow \Delta$. In this case a circular list of edges (k_1,s_1), $(k_2,s_2),\dots,(k_N,s_N)$ is a contour-cycle of Γ iff the list $(\varkappa(k_1),s_1)$, $(\varkappa(k_2),s_2),\dots,(\varkappa(k_N),s_N)$ is a contour-cycle of Δ. Consequently, if the contour of a set of iso-rectangles Γ is known, and the set Δ is topologically equivalent to Γ, the contour of Δ can be found as a mapping of the contour of Γ, without repeating all computations (Fig.8).

a)



b)

k (Fig. 1a)	$\varkappa$(k) (Fig. 8a)
1	3
2	1
3	2
4	4

c)

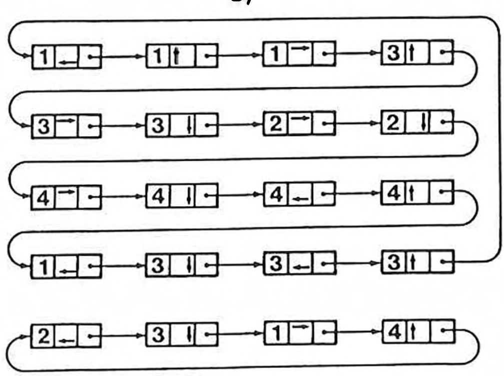


Fig. 8. An example of topological equivalence. a) Set of iso-rectangles topologically equivalent to the set in Fig. 1a. b) Correspondence between iso-rectangles in Fig. 1a and in Fig. 8a. c) Contour of the set in Fig. 8a is a mapping of the contour shown in Fig. 2.

5. ANALYSIS OF THE COMPLEXITY OF THE ALGORITHM

5.1 Time complexity

In the standard way, let us adopt the real random-access machine as the model of computation (cf. [10]). Under this assumption, the worst-case time required to compute the contour of n iso-rectangles can be calculated as follows (cf. section 2.9):

- The time necessary to establish the slabs is determined by sorting the edges. Hence, it is of order $O(n\log n)$.
- Let us consider computations performed by the algorithm in a slab $i \in \{1, \ldots, 2n\}$. In order to establish the list of active edges in slab i, two edges must be inserted to or deleted from the list of active edges of slab i-1. This requires $O(n)$ time. Next, the list of relevant edges is found by a single scan of the active edges. Since the maximum number of active edges is 2n, this scan requires $O(n)$ time. Finally, relevant edges of slabs i-1 and i are connected into fragments of contour-cycles. This can be viewed as the merging of two ordered lists of lengths less than or equal to 2n, followed by a single scan of the merged list. The required time is of order $O(n)+O(n)=O(n)$. Thus, the total time needed to process one slab is of order $O(n)+O(n)+O(n)=O(n)$.
- As the number of slabs is equal to $2n=O(n)$ (no computations for slab 0 are required), the total time necessary to consider all slabs is of order $O(n) \cdot O(n)=O(n^2)$.

Hence, the worst case time necessary to complete the algorithm is of order $O(n\log n)+O(n^2)=O(n^2)$.

5.2 Space complexity

The memory size required by the algorithm can be calculated as follows (cf. section 2.10):

- The list of consecutive vertical edges of input iso-rectangles requires $2n=O(n)$ space.
- A list of active edges in the current slab (i) is at most

twice the total number of iso-rectangles. Hence, it can be stored in O(n) space.

- Given a slab, the relevant edges form a subset of its active edges. Thus, lists of relevant edges in the current (i) and the previous (i-1) slab require O(n) space each.

In total, the space necessary to implement the algorithm is of order $4 \cdot O(n) = O(n)$.

O(n) is indeed the lower bound on the space necessary to solve the contour problem for a set of iso-rectangles. This follows from the observation, that no contour edges can be reported before all input rectangles are considered (cf. Fig.9). The space required to mererly store n input rectangles is already of order O(n).

6. CONCLUSIONS

The paper presents a new solution of the problem of finding the contour of a set of iso-rectangles. The algorithm requires $O(n^2)$ time and O(n) space. This space requirement is optimal. By the use of pointers the algorithm specifies, how contour edges

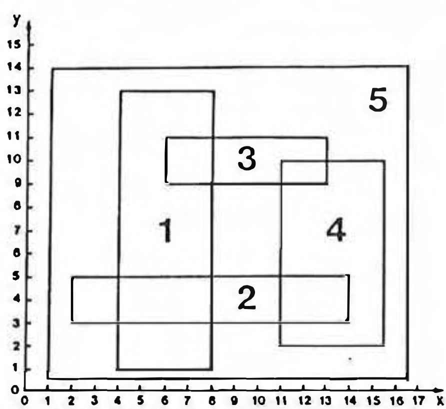


Fig. 9. All input iso-rectangles must be considered before any edge segment is reported: addition of rectangle 5 to the set in Fig. 1a completely changes the contour.

are connected into cycles. Thus, although information on several contour-cycles may interlace at the output of the algorithm, full information on the ordering of edges within contour-cycles is given. In this sense, the paper provides a positive answer to Güting's question, whether there exists an O(n) space algorithm which reports the pieces of the contour in the order of contour-cycles [4].

The algorithm employs orderings of edges of input rectangles rather than their coordinates. For the purpose of handling special cases, related to colinear edges, two ordering relations $\prec_x$ and $\prec_y$ are introduced. They replace the "natural" orderings of edges in the sequences of increasing abscissas or ordinates. Abstraction from coordinates leads to the notion of topological equivalence of sets of iso-rectangles. Two sets are equivalent if they show the same orderings of corresponding edges. A solution of the contour problem for a set of iso-rectangles can be directly mapped to any equivalent set.

The paper leaves some open questions.

1) Does there exist a solution of the contour problem which is both time and space optimal?

2) Does there exist an O(n) space algorithm which reports contour-cycles without interlacing edges from different cycles?

3) Several problems are related to the notion of topological equivalence. For instance,

   - How difficult is it (in terms of time and space complexity) to find whether two sets of iso-rectangles are equivalent?
   - Topological equivalence is a sufficient, but not necessary condition for contour mapping between two sets of iso-rectangles. Does there exist a nontrivial sufficient and necessary condition? If so, how difficult is it to verify whether this condition is satisfied?

REFERENCES

1. Eastman, C.M., Lividini, Y. Spatial search. Report 55, Institute of Physical Planning, Carnegie-Mellon University, Pittsburgh, Pa. 1975.

2. Edelsbrunner, H. Intersection problems in computational geometry. Report F 93, Institute fur Informationsverarbeitung, Technical University of Graz, Graz, Austria, 1982.

3. Edelsrunner, H., v. Leeuwen, J. Multidimensional data structures and algorithms. A bibliography. Report F 104, Institute fur Informationsverarbeitung, Technical University of Graz, Graz, Austria, 1983.

4. Gueting, R.H. An optimal contour algorithm for iso-oriented rectangles. Report 82-CS-04, Unit for Computer Science, McMaster University, Hamilton, Ont., 1982.

5. Güting, R.H. Optimal divide-and-conquer to compute measure and contour for a set of iso-rectangles. Report 141, Abteilung Informatik, University of Dortmund, Dortmund, Germany, 1982.

6. Lipski, W., Preparata, F.P. Finding the contour of a union of iso-oriented rectangles. Journal of Algorithms 1 (1980), 235-246.

7. Head, C., Conway, L. Introduction to VLSI systems. Addison-Wesley, Reading, Mass., 1980.

8. Prusinkiewicz, P., Raghavan, V. V. A simple solution of the contour problem for a set of iso-rectangles. Report CS-83-02, Department of Computer Science, University of Regina, Regina, Sask., 1983.

9. Prusinkiewicz, P., Stepien, C. Selected topics in computer graphics. Technical University of Warsaw Press, Warsaw, Poland, 1982.

10. Shamos, M.I. Computational geometry. Ph.D. Thesis, Yale University, 1977.

11. Sutherland, T.E., Sproull, R.F., Schumaker, R.A. A characterization of ten hidden-surface algorithms. Computing Surveys 6,1 (1974) 1-55.

12. Vitanyi, P.M.B., Wood, D. Computing the perimeter of a set of rectangles. Report 81-CS-04, Unit for Computer Science, McMaster University, Hamilton, Ont., 1981.