

Fast Algorithm for Polygon Decomposition

SURENDRA NAHAR AND SARTAJ SAHNI, FELLOW, IEEE

Abstract—We develop an $O(k \log(k) + n)$ algorithm, where n is the number of vertices in the polygon and k the number of vertical inversions, to decompose rectilinear polygons into rectangles. This algorithm uses horizontal cuts only and reports nonoverlapping rectangles whose union is the original rectilinear polygon. This algorithm has been programmed in Pascal on an Apollo DN320 workstation. Experimentation with rectilinear polygons from VLSI artwork indicates that our algorithm is significantly faster than the plane sweep algorithm and the algorithm proposed in [5].

Key words and phrases: polygon decomposition, computational geometry, time complexity.

I. INTRODUCTION

THE PROBLEM OF decomposing a polygon into basic components has applications in computer graphics, data bases, image processing, VLSI layout, and artwork analysis [5], [15], [13], [3]. The development of efficient algorithms to decompose a polygon has been the focus of much research. For example, Keil [8] develops polynomial time algorithms to decompose a simple polygon (i.e., one with no holes) into convex polygons, spiral polygons, star-shaped polygons, and monotone polygons. These algorithms minimize the number of simpler components without introducing any steiner points. Liu and Ntafos [10] consider the case when Steiner points are allowed. They develop a linear time algorithm to partition a simple monotone polygon into a minimum number of star-shaped polygons.

Asano *et al.* [1] present an $O(n^2)$ algorithm (n being the number of polygon vertices) to decompose a polygon with no holes into a minimum number of trapezoids. When the polygon has holes, this decomposition problem is NP-complete [4], [1]. In [11], an $O(n \log n)$ algorithm to partition a rectilinear polygon into a minimum number of uniformly monotone rectilinear polygons is developed. An $O(n^3)$ algorithm to find a maximum set of independent chords in a circle is used, in [12], to partition simple polygons into a minimum number of uniformly monotone polygons.

In this paper, we are concerned solely with the decom-

position of rectilinear hole-free polygons into a minimum number of rectangles (cf. Fig. 1). Our work is trivially extendable to nonrectilinear polygons and also to polygons with holes.

The rectangles in the decomposition of a polygon are required to be disjoint (or nonoverlapping). When this restriction is removed (i.e., overlapping rectangles are allowed), decomposing a rectilinear polygon with holes into a minimum number of possibly overlapping rectangles is NP-complete. This is seen by observing that every 0, 1 matrix is the digitized version of some rectilinear polygon with holes (the 1's represent polygon interiors, the 0's exteriors). Hence the rectilinear picture compression problem [4] is the same as the polygon decomposition problem. The rectilinear picture compression problem is NP-complete.

Lingas *et al.* [9] use dynamic programming to obtain an $O(n^4)$ algorithm to dissect a rectilinear polygon with no holes into disjoint rectangles with minimum total edge length. They also show this to be NP-complete when holes are present. Ohtsuki [14] develops an $O(n^{5/2})$ algorithm to decompose a rectilinear polygon into a minimum number of rectangles using both horizontal and vertical cuts. This algorithm has been improved to $O(n^{3/2} \log n)$ in [7].

In some applications, only horizontal cuts are permissible. Fig. 2 shows the best decomposition when both horizontal and vertical cuts are permitted, as well as when only horizontal cuts are permitted. Consider any hole-free rectilinear polygon P . Let R_{HV} be the minimum number of rectangles in a decomposition using both horizontal and vertical cuts. Let R_H and R_V , respectively, denote the minimum number when only horizontal or only vertical cuts are permitted. It can be shown that

$$\min \{R_H, R_V\} < \frac{3}{2} * R_{HV}.$$

To see this, draw all possible vertical and horizontal line segments that join two internal corners (see Fig. 3). Let p_H be the number of horizontal line segments drawn, p_V the number of vertical segments, and p the number of lines in a maximum independent set of line segments (two line segments are independent iff they do not intersect). Clearly, $p \leq p_V + p_H$. So, $p \leq 2 * \max \{p_V, p_H\}$. From [14], it follows that

$$R_{HV} = H - p - 1$$

$$R_V = H - p_V - 1$$

$$R_H = H - p_H - 1$$

Manuscript received August 19, 1986; revised August 5, 1987, and November 6, 1987. This work was supported in part by the National Science Foundation under Grants DCR-8305567 and DCR-8420935. The review of this paper was arranged by Associate Editor R. H. J. M. Otten.

S. Nahar was with the Computer Science Department, University of Minnesota, Minneapolis. He is now with AT&T Bell Laboratories, Murray Hill, NJ.

S. Sahni is with the Computer Science Department, University of Minnesota, Minneapolis, MN 55455.

IEEE Log Number 8718828.

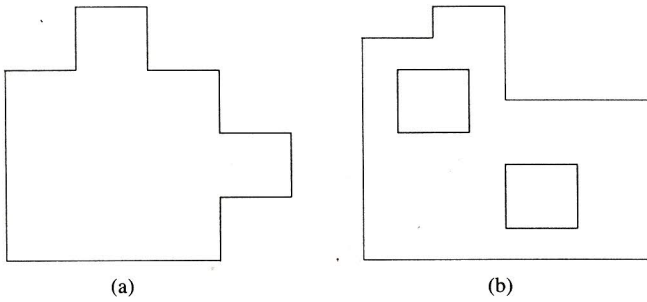


Fig. 1. Rectilinear polygons. (a) No holes (hole-free). (b) With holes.

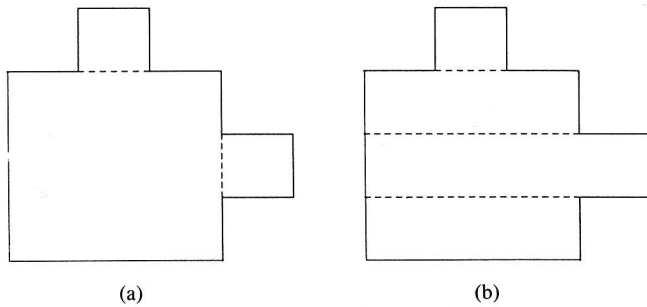


Fig. 2. Rectilinear polygon decomposition. (a) Horizontal and vertical cuts. (b) Only horizontal cuts.

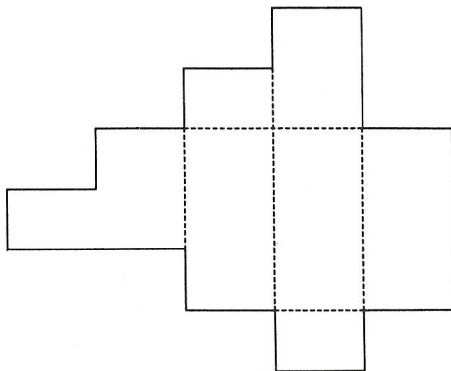


Fig. 3. All possible horizontal and vertical line segments that join two internal corners.

where H is the number of horizontal segments in the boundary of the original polygon. Hence,

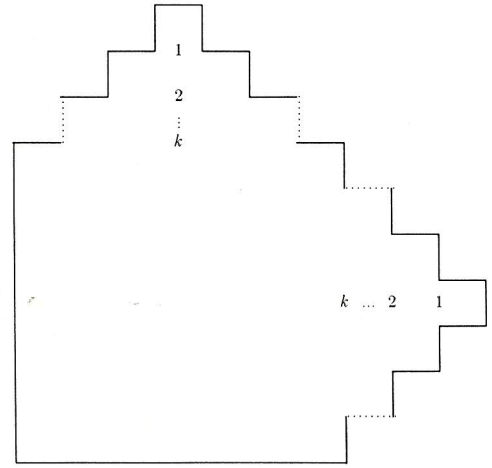
$$\begin{aligned} \min \{R_H, R_V\} &= H - \max \{p_V, p_H\} - 1 \\ &\leq H - \frac{p}{2} - 1 \leq R_{HV} + \frac{p}{2}. \end{aligned}$$

Also, since p segments create $p + 1$ polygons, $p < R_{HV}$. So,

$$\min \{R_H, R_V\} < \frac{3}{2} * R_{HV}.$$

Fig. 4 gives an example that approaches this bound asymptotically.

One application of horizontal cut decomposition is corner stitching [15]. This results in fast algorithms for interactive VLSI layout editing. An algorithm for horizontal



$$\begin{aligned} H &= 4 * k + 2, p = 2 * k \\ \frac{\min \{R_H, R_V\}}{R_{HV}} &= \frac{(4 * k + 2) - k - 1}{(4 * k + 2) - 2 * k - 1} = \frac{3}{2} - \frac{1}{(4 * k + 2)} = \frac{3}{2} \text{ as } k \rightarrow \infty \end{aligned}$$

Fig. 4. An example where asymptotic bound is reached.

cut decomposition is given in [5]. This is reproduced in Fig. 5. This algorithm has a worst-case complexity of $O(n^2)$, though it is expected to do better than this on many polygons. An $O(n \log n)$ algorithm for horizontal cut decomposition is trivially obtained by performing a plane sweep using a vertical (or horizontal) scan line.

In this paper, we develop a new algorithm to decompose a hole-free rectilinear polygon into a minimum number of rectangles using only horizontal cuts. Our new algorithm takes advantage of the fact that polygons that arise in practice have a small number of vertical and horizontal inversions (defined below) relative to the number of vertices. The number of vertical inversions k_V of a hole-free rectilinear polygon is obtained by traversing the vertices of the polygon in anticlockwise order. This traversal begins at a vertex with least y coordinate. Of the several vertices that have this y value, the one with least x is chosen. This vertex, denoted σ , is called the start or σ vertex.

Consider the sequence of distinct y values encountered in the traversal. This sequence is initially increasing, then decreasing, then increasing, then decreasing, etc. The tail end of this sequence is always decreasing. Each time the sequence changes from increasing to decreasing, there is a vertical inversion.

Example: For the polygon of Fig. 6(a), the y value sequence is $y_1 < y_2 < y_3 < \dots < y_{14} > y_{15} \dots > y_{20}$. There is one vertical inversion in the sequence. Fig. 6(b) shows a polygon with two vertical inversions and Fig. 6(c) shows one with three vertical inversions. \square

The number of horizontal inversions is defined in an analogous manner. This time, the anticlockwise traversal begins at a vertex with least x value. Of the several vertices that have this x value, the one with the lowest y value is used. This start vertex is called the τ vertex. We consider the sequence of distinct x values encountered during the traversal. This sequence is initially increasing, then decreasing, then increasing, then decreasing, etc. The tail

