Algorithmica (1990) 5: 11-42

Algorithmica

Approximation Algorithms for Partitioning a Rectangle with Interior Points[1]

Teofilo Gonzalez[2] and Si-Qing Zheng[2,3]

Abstract. Let R be a rectangle and let P be a set of points located inside R. Our problem consists of introducing a set of line segments of least total length to partition the interior of R into rectangles. Each rectangle in a valid partition must not contain points from P as interior points. Since this partitioning problem is computationally intractable (NP-hard), we present efficient approximation algorithms for its solution. The solutions generated by our algorithms are guaranteed to be within three times the optimal solution value. Our algorithm also generates solutions within four times the optimal solution value when R is a rectilinear polygon. Our algorithm can be generalized to generate good approximation solutions for the case when R is a rectilinear polygon, there are rectilinear polygonal holes, and the sum of the length of the boundaries is not more than the sum of the length of the edges in an optimal solution.

Key Words. Approximation algorithms, Partition of rectilinear polygons, Polynomial-time complexity.

1. Introduction. Partitioning a polygon is one of the fundamental problems in computational geometry. Traditionally the objective is finding a convex partition with a minimum number of components, and fortunately polynomial-time algorithms for these problems exist [CD], [GJPT], [LLMP], [LLMPL], [S]. Lingas *et al.* [LPRS] investigated the problem of partitioning a rectilinear polygon into rectangles by introducing a set of line segments (edges) of least total length. In VLSI design, the problem of dividing routing regions into channels can be reduced to this new partition problem [R]. Several variations of this partition problem were shown to be NP-hard, but the computational complexity of some other related problems remains unknown [LPRS]. In this paper we present efficient approximation algorithms to solve restricted versions of this partitioning problem. Let us now formally define several variations of the partitioning problem and review some previous results.

A rectilinear *boundary* is a simple polygon with the additional constraint that all of its sides are either parallel or perpendicular to each other. A *hole* is a simple rectilinear polygon located inside the rectilinear boundary whose sides are parallel or perpendicular to the sides of the rectilinear boundary. There can be no holes inside a hole. A single point inside the boundary is called a *degenerate*

[1] A preliminary version of this paper appeared in the *Proceedings of the Symposium on Computational Geometry*, June 1985, pp. 281-287. This research was supported in part by the National Science Foundation under Grant DCR-8503163.

[2] Department of Computer Science, University of California, Santa Barbara, CA 93106, USA.

[3] Professor Si-Qing Zheng is now with the Computer Science Department, Louisiana State University, Baton Rouge, LA 70803-4020, USA.

Received March 10, 1986; revised May 11, 1988. Communicated by David Dobkin.

Table 1.1

Problem	Boundary	Points and holes	Complexity
RP-HF	Rectilinear polygon	Hole Free	$O(n^4)$
RG-NLP	Rectangle	Non-corectilinear points[1]	NP
RP-NLP	Rectilinear polygon	Non-corectilinear points[1]	NP
RG-P	Rectangle	Points	NP-complete
RP-P	Rectilinear polygon	Points	NP-complete
RG-RP	Rectangle	Rectilinear polygons	NP-complete
RP-RP	Rectilinear polygon	Rectilinear polygons	NP-complete
RG-RPP	Rectangle	Rectilinear polygons and points	NP-complete
RP-RPP	Rectilinear polygons	Rectilinear polygons and points	NP-complete

[1] A set of points is said to be non-corectilinear if no two points in the set are located along the same vertical or horizontal line.

hole. A *figure* is a rectilinear boundary which may contain an arbitrary number of nonoverlapping holes. A *rectangular partition* of a figure is a set of line segments lying within its boundary and not crossing any nondegenerate hole so that when added to the figure, the area not enclosed by holes is partitioned into rectangles that do not contain as interior points any of the degenerate holes. The partitioning line segments are called *edges.* For every problem instance I and every set of edges $E(I)$ in a feasible solution we use the function $L(E(I))$ to represent the sum of the length of the edges in $E(I)$. For problem instance I, the rectangular partition $E(I)$ is said to be a minimum edge length partition if $L(E(I)) \leq L(W(I))$, where $W(I)$ is any rectangular partition.

Known results about the computational complexity of the decision problem corresponding to our minimum edge rectangular partitioning of rectilinear polygons is displayed in Table 1.1.

Our notation is as follows: by RP-NLP we mean "minimum edge length rectangular partitioning of a Rectilinear Polygon with Non-corectiLinear Points." The other abbreviations in Table 1.1 are self-explanatory. The computational complexity of the problems in the above table is displayed in Figure 1.1, where "problem A → problem B" means that problem A is polynomially reducible to problem B, i.e., problem A is not computationally "harder" than problem B.

The computational complexity of the RG-NLP and RP-NLP has not yet been determined. Since an optimal solution for a given problem instance of RG-NLP

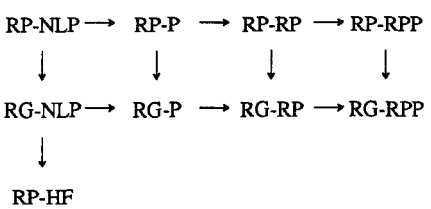


Fig. 1.1

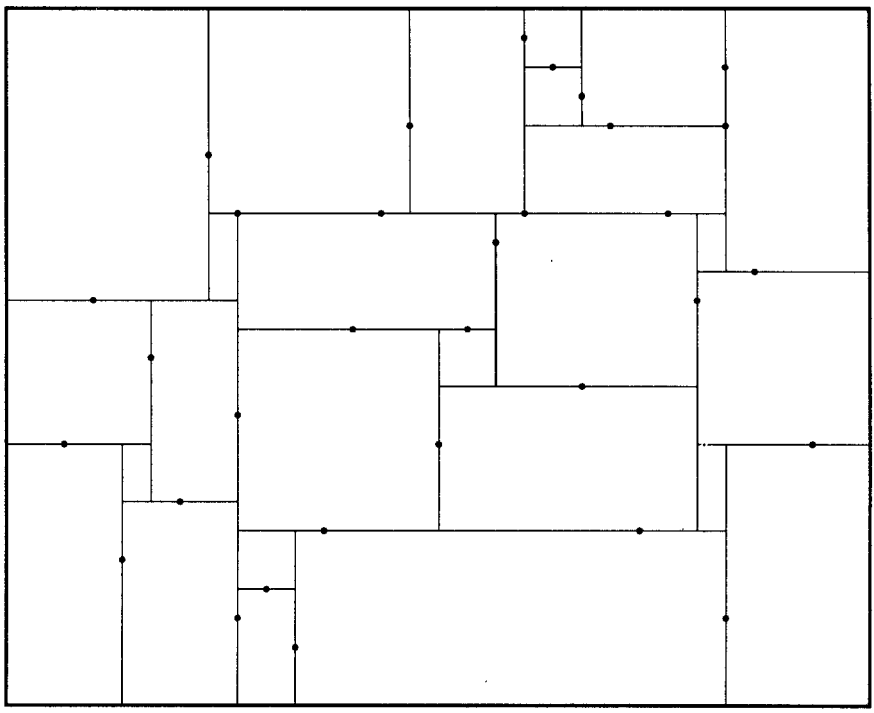


Fig. 1.2. Near-optimal solution.

could be very complex (see Figure 1.2 for an example of a near-optimal solution) and it seems that there is no way to obtain an optimal solution without exhaustive search, we conjecture that both of these problems are NP-complete.

A problem instance is formally defined as $I=(B, P, H)$, where $B=(q_0, q_1, \ldots, q_{m-1})$ is a sequence of corner points (in clockwise order) that defines a rectilinear polygonal boundary, $P=\{p_1, p_2, \ldots, p_n\}$ is a set of points (degenerate holes) inside the boundary, and $H=\{h_1, h_2, \ldots, h_r\}$ is a set of rectilinear holes inside the boundary. Each of these holes is defined by a sequence of points. We shall refer to the polygon defined by B as the *global boundary* and to each of the polygons that defines a hole as the *hole boundary.* Certainly B, P, and H must satisfy a set of constrains so that "I" is a valid problem instance. A line segment is represented by its two endpoints, i.e., $[(x_i, y_i), (x_j, y_j)]$ represents the line segment with endpoints (x_i, y_i) and (x_j, y_j). Let $B(I)$ represent the set of edges that defines the boundary for problem instance I and let $H(I)$ represent the set of edges that defines the boundary of the holes in problem instance I. $L(B(I))$ and $L(H(I))$ represent the sum of the length of the edges in $B(I)$ and $H(I)$, respectively.

Lingas [L] developed a polynomial-time approximation algorithm for all of these partitioning problems, however, this algorithm may generate a solution with an objective function value far from the optimal one. Levcopoulos [Lev1]

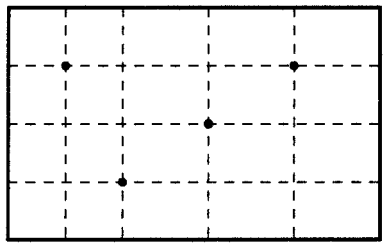


Fig. 1.3. Grid lines (dashed lines are grid lines).

developed an algorithm to solve this problem by using the "thickest-first" heuristic. He claims that his algorithm has an approximation bound of 5 and time complexity $O(n^2)$. Recently, Levcopoulos [Lev2] presented an $O(n \log n)$ algorithm that guarantees a solution with objective function value within a constant times the optimal-solution value. His algorithm uses a fast implementation of the algorithm given by Gonzalez and Zheng [GZ1] for the RG-P problem. Gonzalez and Zheng's algorithm generates solutions within $3+\sqrt{3}$ of the optimal. In this paper we present an algorithm that generates solutions for the RG-P problem that are within three times of the optimal.

In the next section we present an approximation algorithm for the RG-P problem. We show that our algorithm generates solutions such that $L(E_{\text{apx}}(I)) \leq 3L(E_{\text{opt}}(I))$. We also show that the approximation bound for this algorithm is best possible in the sense that for every $\varepsilon > 0$ there are problem instances such that $L(E_{\text{apx}}(I)) > 3L(E_{\text{opt}}(I)) - \varepsilon$. This approximation algorithm together with the techniques in [GZ1] can be used to obtain an approximation algorithm for the RP-P problem.

We define the *grid* induced by a problem instance as the set of horizontal and vertical line segments introduced by extending all sides of the figure and the holes in both directions until either the line segment reaches the outside of the figure or the interior of a hole. Figure 1.3 illustrates the grid induced (by dashed lines) for a problem instance. It can be easily shown that in any optimal solution for the RP-RPP problem, all partitioning edges lie on the induced grid [LPRS]. The solutions generated by our algorithms have this property.

2. Approximation Algorithms for the RG-P Problem. In this section we present our approximation algorithm for the RG-P problem. Our approach consists of transforming the instance of the RG-P problem into an instance of a generalized RP-HF problem (JRP-HF) for which we can find an optimal rectangular partition in polynomial time. Such a partition is our solution to the original problem. We assume that the rectangular boundary is located in the first quadrant of the xy-coordinate system whose *origin* is the lower left corner of the rectangular boundary. The first step of our procedure, the *scanning algorithm*, consists of scanning the points and introducing jogging lines to connect these points directly or indirectly to the boundary. Let $p_1 = (x_1, y_1), p_2 = (x_2, y_2), \ldots, p_n = (x_n, y_n)$ be

the points inside the rectangle and assume that they have been reordered in such a way that

(1) $x_i \leq x_j$ for $1 \leq i < j \leq n$; and
(2) if $x_i = x_j$ then $y_i > y_j$, for $1 \leq i < j \leq n$.

Our algorithm traverses the points in the order $p_1, p_2, \ldots, p_n$.

> Let $(X, Y)_i = \{(x, y) \mid ((x, y)$ is a point on the boundary or a point in one of the line segments introduced during the previous iterations) and $x \leq x_i\}$.

During the ith iteration point p_i is connected directly or indirectly to the boundary. Let $(x, y) \in (X, Y)_i$ be a closest point (with respect to the L_1 metric) to p_i. Point p_i is connected to (x, y) by a shortest path (L_1 metric). The path consists of at most one vertical line segment and at most one horizontal line segment. If $y \geq y_i$, then the horizontal line segment (if present) must include p_i; otherwise the vertical line segment (if present) must include p_i. At the end of this section we explain why it is important for our algorithm to connect points this way. Let $C(I)$ be the set of line segments introduced by the above scanning algorithm. Figure 2.1 shows the set $C(I)$ for a problem instance with an optimal solution of the form given in Figure 1.2. The algorithm introduces at each iteration either a *vertical*

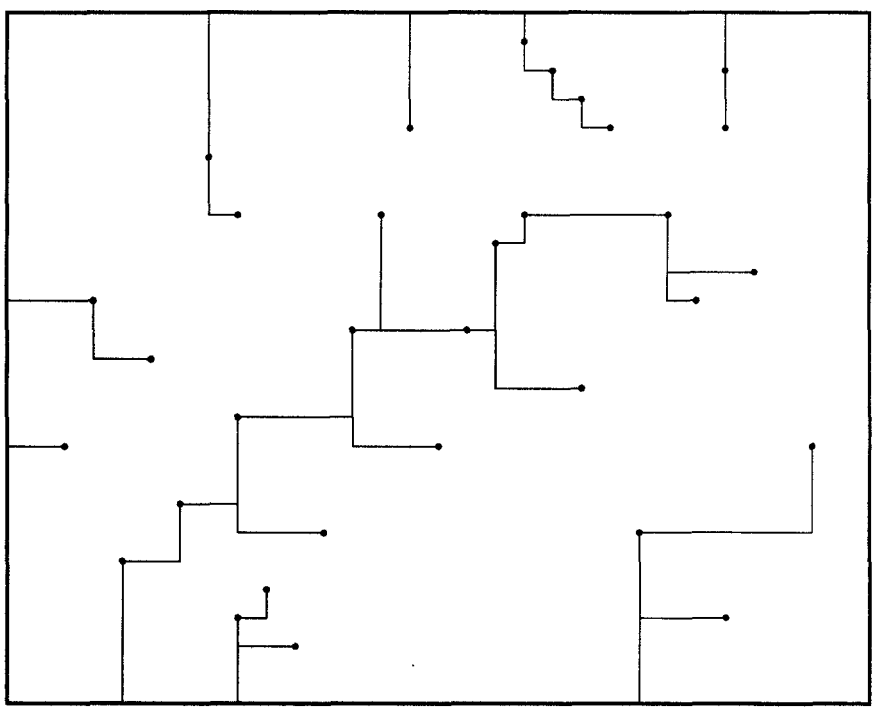


Fig. 2.1. $C(I)$.

line segment (if $x = x_i$ and $y \neq y_i$), a *horizontal line segment* (if $x \neq x_i$ and $y = y_i$), a *right corner segment* (if $x \neq x_i$ and $y > y_i$), a *left corner segment* (if $x \neq x_i$ and $y < y_i$), or an *empty segment* (if $x = x_i$ and $y = y_i$). The *corner point* in a right corner segment or in a left corner segment is the point located at the intersection of the line segments that form the corner segment.

The problem instance displayed in Figure 2.1 is an instance of the JRP-HF problem. In such a problem we do not have holes nor points. We only have a rectangular boundary with jogging lines connected to it. These jogging lines should be viewed as boundaries. Each "side" of a jogging line is part of the boundary. The $O(n^4)$ algorithm for the RP-HF problem given in [LPRS] can be trivially modified to find an optimal partition for the JRP-HF problem. The main idea behind the modification is to treat the internal line segments as two sided boundaries.

Algorithm TRANS
Step 1: Perform the scanning procedure to construct an instance, I', of the JRP-HF problem.
Step 2: Use a modified version of the $O(n^4)$ algorithm, for the RP-HF problem, that appears in [LPRS] to find an optimal solution to I'.
End of algorithm TRANS

THEOREM 2.1. *The time complexity for algorithm* TRANS *is* $O(n^4)$.

PROOF. First let us show that step 1 of algorithm TRANS takes $O(n^2)$ time. Clearly, the scanning procedure introduces at most two line segments for each point p_i. Hence, when computing the closest point to p_i we only need to find a shortest path (L_1 metric) from p_i to at most $2(i-1)+3$ different line segments (at most $2(i-1)$ line segments from the previous $i-1$ points and three other line segments for the boundaries) and then finding the smallest of these distances. Finding the closest point (L_1 metric) from p_i to a line segment can be easily performed in constant time, and finding the smallest of these distances can be done in $O(n)$ time. Hence, the overall time complexity for step 1 is $O(n^2)$. After modifying the algorithm in [LPRS] for step 2, its time complexity remains the same $O(m^4)$, where m is the number of sides of the "generalized" rectilinear polygon. The "generalized" rectilinear polygon that we construct has $O(n)$ sides, since the line segments connected to boundaries of the rectangle are treated as sides of the "generalized" rectilinear polygon. Hence, the overall time complexity for the algorithm is $O(n^4)$. □

Let $E_{apx}(I) = C(I) \cup E_{opt}(I')$ be the solution generated by algorithm TRANS. Let $E(I) = C(I) \cup E_{opt}(I) \cup D(I)$, where $E_{opt}(I)$ is an optimal solution for I, $C(I)$ is the set of line segments introduced by the scanning algorithm, and $D(I)$ (to be defined later) is a set of line segments that transforms $C(I) \cup E_{opt}(I)$ into a rectangular partition. Since $(E_{opt}(I) \cup D(I)) - C(I)$ is a solution to the JRP-HF problem and $E_{opt}(I')$ is an optimal solution to the JRP-HF problem, we

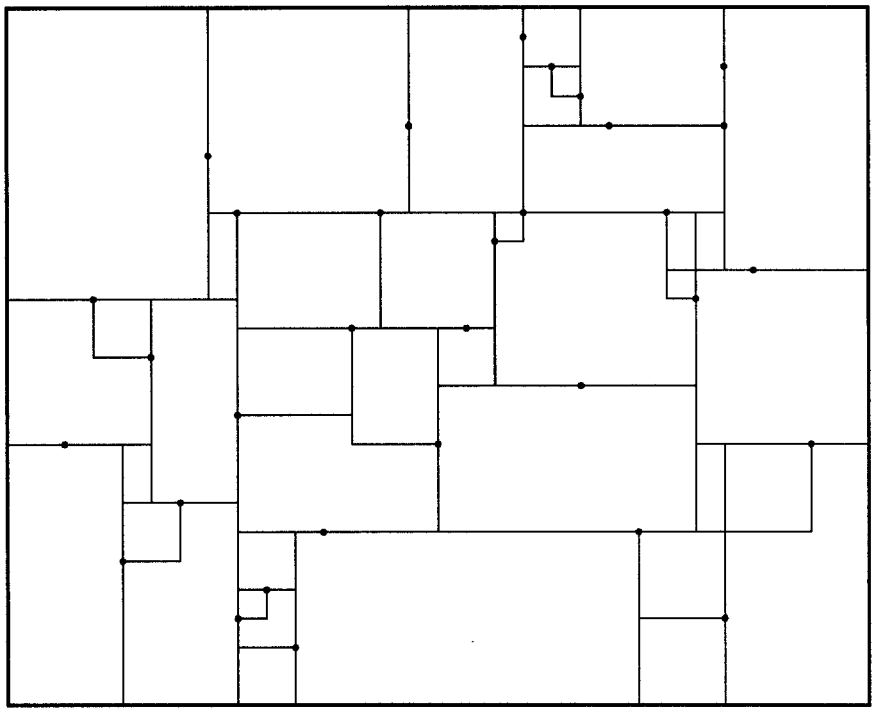


Fig. 2.2. $E_{\text{opt}}(I) \cup C(I)$.

know that $L(E(I)) \geq L(E_{\text{apx}}(I))$. If we show that $L(E(I)) \leq 3L(E_{\text{opt}}(I))$, then $L(E_{\text{apx}}(I)) \leq 3L(E_{\text{opt}}(I))$. Therefore, to prove our claim we need to show that there exists a $D(I)$ such that $E(I) = C(I) \cup E_{\text{opt}}(I) \cup D(I)$ is a rectangular partition and $L(E(I)) \leq 3L(E_{\text{opt}}(I))$.

Let $E_{\text{opt}}(I)$ be any optimal rectangular partition and let R be any rectangle in $E_{\text{opt}}(I)$. In what follows we define the names for the different line segments from $C(I)$ that may appear in R and then we characterize the interior of R in terms of the different components that appear inside them. A *left joint* (LJ) is obtained by introducing a left corner segment (including its corner point) inside R that intersects the left and top sides of R (see Figure 2.3(c)). A *right joint* (RJ) is obtained by introducing a right corner segment (including its corner point) inside R that intersects the right and top sides of R (see Figure 2.3(b)). If there is a left joint there could also be a right semijoint (Figure 2.3(i)). A *right semijoint* (RS) is obtained by introducing a right corner segment (including its corner point) inside R that intersects the corner point of the left joint and the right side of R. We say that a line segment is *inside the right joint* if it is located inside the rectangle formed by the right joint and the sides of R. We say that a line segment is *inside the right semijoint* if it is located inside the rectangle formed by the right semijoint, the vertical portion of the left joint, and the sides of R. A *vertical cut* (VC) is obtained by introducing a vertical line segment that intersects the top and bottom sides of R and does not overlap with the left nor the right sides of

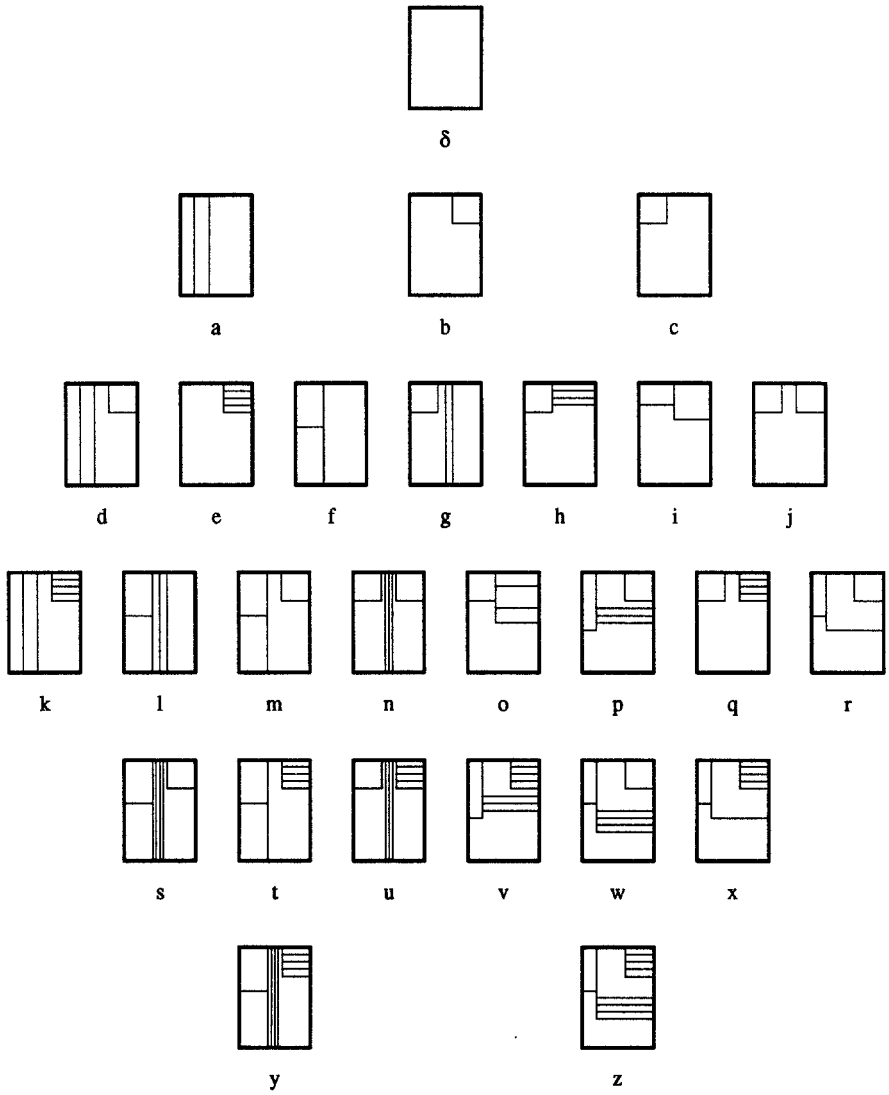


Fig. 2.3

R (see Figure 2.3(a)). A *partial vertical cut* (PVC) is a vertical line segment that intersects the bottom side of R and the corner point of the left joint (see Figure 2.3(f)). A *horizontal cut* (HC) is a horizontal line segment that intersects the left and the right sides of R and does not overlap with the top nor the bottom sides of R (see Figure 2.4Δ). A *partial horizontal cut* (PHC) is a horizontal line segment that intersects the right side of R and either a partial vertical cut, a vertical cut, the vertical portion of the left joint, the vertical portion of the right semijoint, or the vertical portion of the right joint; and does not overlap with the horizontal portion of the right joint, the horizontal portion of the right semijoint, the top

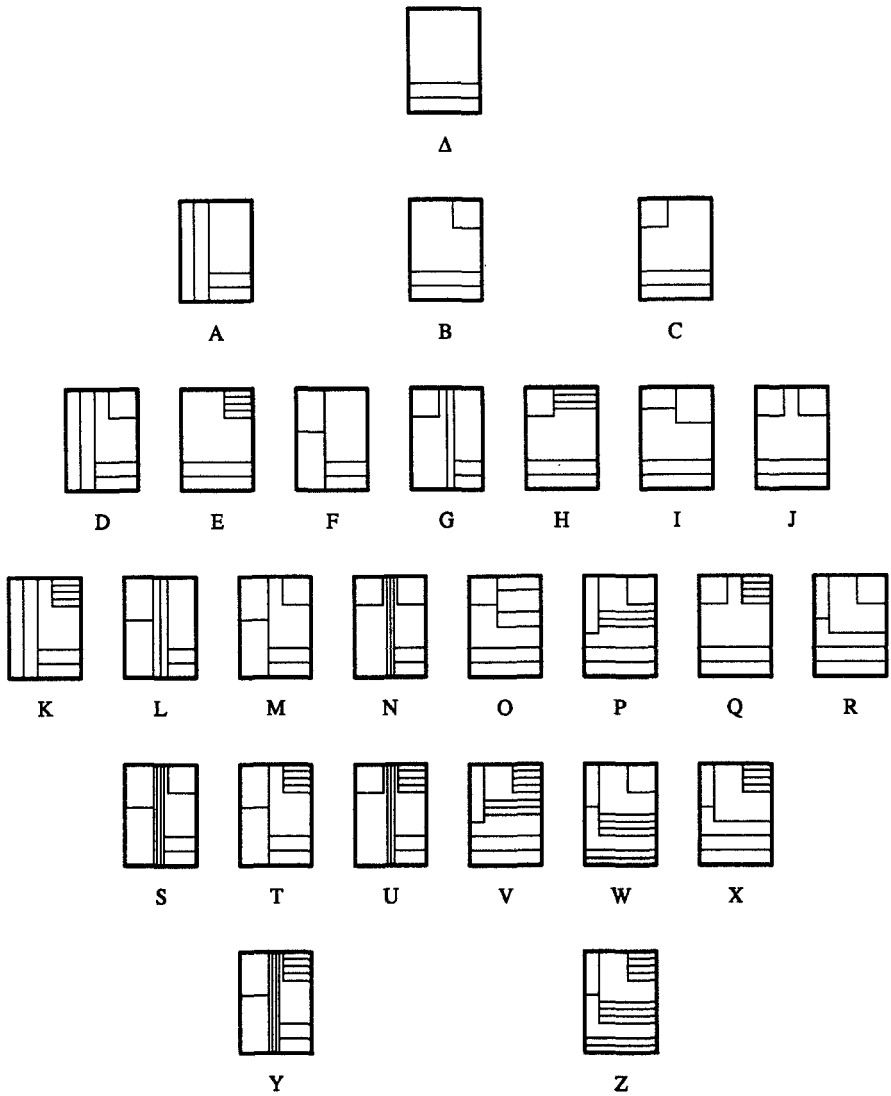


Fig. 2.4

side of R, or the bottom side of R. A partial horizontal cut that intersects the vertical portion of the right joint is called a *Partial Horizontal Cut located inside the Right Joint* (PHC-RJ). Figure 2.3(e) shows a rectangle with a right joint and with partial horizontal cuts inside the right joint. The remaining partial horizontal cuts, those not inside the right joint, will be named differently depending on the existence of a PVC and/or VC segments. In rectangles with a partial vertical cut and/or vertical cuts, a partial horizontal cut that intersects the left joint, the partial vertical cut, or a vertical cut is called a PHC-NRJV (see Figure 2.4(F)). In rectangles without a partial vertical cut nor a vertical cut, a partial horizontal

Table 2.1

Type (type)	LJ	PVC	VC	RS	PHC-NRJNV	RJ	PHC-RJ	PHC-NRJV	HC
δ (Δ)									(*)
a (A)			* (*)					(*)	
b (B)						* (*)			(*)
c (C)	* (*)								(*)
d (D)			* (*)			* (*)		(*)	
e (E)						* (*)	* (*)		(*)
f (F)	* (*)	* (*)						(*)	
g (G)	* (*)		* (*)					(*)	
h (H)	* (*)				* (*)				(*)
i (I)	* (*)			* (*)					(*)
j (J)	* (*)					* (*)			(*)
k (K)			* (*)			* (*)	* (*)	(*)	
l (L)	* (*)	* (*)	* (*)					(*)	
m (M)	* (*)	* (*)				* (*)		(*)	
n (N)	* (*)		* (*)			* (*)		(*)	
o (O)	* (*)			* (*)	* (*)				(*)
p (P)	* (*)				* (*)	* (*)			(*)
q (Q)	* (*)					* (*)	* (*)		(*)
r (R)	* (*)			* (*)		* (*)			(*)
s (S)	* (*)	* (*)	* (*)			* (*)		(*)	
t (T)	* (*)	* (*)				* (*)	* (*)	(*)	
u (U)	* (*)		* (*)			* (*)	* (*)	(*)	
v (V)	* (*)				* (*)	* (*)	* (*)		(*)
w (W)	* (*)			* (*)	* (*)	* (*)			(*)
x (X)	* (*)			* (*)		* (*)	* (*)		(*)
y (Y)	* (*)	* (*)	* (*)			* (*)	* (*)	(*)	
z (Z)	* (*)			* (*)	* (*)	* (*)	* (*)		(*)

cut that intersects the left joint or the right semijoint is called a PHC-NRJNV (see Figure 2.3(o) and (h)). Rectangle R may contain in it any of the following *components*: LJ, PVC, VC, RS, PHC-NRJNV, RJ, PHC-RJ, PHC-NRJV, or HC.

We shall identify 54 different *types* of rectangles (δ, a-z, Δ, A-Z) depending on their components. The components in each type of rectangle are given in Table 2.1. Figures 2.3 and 2.4 depict these rectangles. Table 2.1 should be interpreted as follows: a "*" in entry i, j indicates that rectangle type i contains component j and a "(*)" in entry $(i), j$ indicates that rectangle type i contains component j. We say that a rectangle is in *canonical form* if it is a rectangle of type δ, a-z, Δ, or A-Z. In Lemma 2.1 we show that each rectangle in the rectangular partition $E_{\text{opt}}(I)$ after adding the line segments in $C(I)$ is in canonical form.

Let $W_i = \{e \mid e \in C(I)$ and e is a line segment introduced by the scanning algorithm before the $(i+1)$st iteration$\}$. Clearly $W_n = C(I)$. Let $R_1, R_2, \ldots, R_q$ be the set of rectangles in $E_{\text{opt}}(I)$. For all i, j, let $R_{i,j} = R_j \cup \{$segments from W_i inside $R_j\}$. In what follows we prove by induction on i that rectangle $R_{i,j}$ is in canonical form. In the remaining part of this paper we say that $p = (x, y)$ is located to the *right* (*left*) of the vertical line segment $l = [(x', y'), (x', y'')]$ if

$x > x'$ ($x < x'$). Similarly, we say that $p = (x, y)$ is located *above* (*below*) the horizontal line segment $l = [(x', y'), (x'', y')]$ if $y > y'$ ($y < y'$).

LEMMA 2.1. *For all i and j, rectangle $R_{i,j}$ is in canonical form.*

PROOF. The proof is by induction on the number of iterations (i) performed by the scanning algorithm.

Basis. For all j, $R_{0,j}$ is in canonical form.

Since $W_0 = \varnothing$, it must be that $R_{0,j} = R_j$. Since each rectangle R_j is in canonical form (type δ), we know that $R_{0,j}$ is in canonical form.

Induction Hypothesis. Assume that $R_{i,j}$ is in canonical form.

Induction Step. $R_{i+1,j}$ is in canonical form.

By the induction hypothesis we know that $R_{i,j}$ is in canonical form. If the algorithm introduces no line segment inside of $R_{i,j}$ during the $(i+1)$st iteration, then $R_{i+1,j} = R_{i,j}$ is in canonical form. So, let us assume that the algorithm introduces some line segments inside $R_{i,j}$ during the $(i+1)$st iteration. It is simple to verify that p_{i+1} must be located to the right of the left side of $R_{i,j}$ and since $E_{\text{opt}}(I)$ is a rectangular partition we know that p_{i+1} cannot be located inside rectangle $R_{i,j}$. There are three cases depending on the type of rectangle $R_{i,j}$.

Case 1. Rectangle $R_{i,j}$ has no interior line segments (type δ).

There are three subcases depending on the type of segment introduced inside $R_{i,j}$ during the $(i+1)$st iteration.

Subcase 1.1. The scanning algorithm introduces either a horizontal or a vertical line segment.

Clearly, after introducing either of these line segments we obtain a rectangle of type a or Δ. Hence, $R_{i+1,j}$ is in canonical form.

Subcase 1.2. The scanning algorithm introduces a right corner segment.

Clearly, introducing a right corner inside $R_{i,j}$ transforms it into a rectangle of type Δ, a, or b, depending on which section of the right corner segment is introduced inside $R_{i,j}$. Hence, $R_{i+1,j}$ is in canonical form.

Subcase 1.3. The scanning algorithm introduces a left corner segment.

Clearly, introducing a left corner inside $R_{i,j}$ transforms it into a rectangle of type Δ, a, or c, depending on which section of the left corner segment is introduced inside $R_{i,j}$. Hence, $R_{i+1,j}$ is in canonical form.

This completes the proof of Case 1.

Case 2. Rectangle $R_{i,j}$ contains only interior line segments that do not intersect the right side of $R_{i,j}$ (type a, c, f, g, and l).

It is simple to prove that if point p_{i+1} is not located below the bottom side of $R_{i,j}$, then it must be located to the right of the rightmost point of any line segment

inside $R_{i,j}$; and if p_{i+1} is located below the bottom side of $R_{i,j}$, then it must be located not to the left of the rightmost point of any line segment inside $R_{i,j}$. There are four cases depending on the type of segment introduced by the algorithm.

Subcase 2.1. The algorithm introduces a vertical line segment.

Introduction of a vertical line segment inside $R_{i,j}$ results in the following set of rectangles $R_{i+1,j}$. Let us explain our notation by example. By "c→f|g" we mean that introducing a segment (vertical line segment in this case) in a rectangle type c results in either a rectangle of type f or type g.

$$\begin{aligned} &a \to a,\\ &c \to f|g,\\ &f \to l,\\ &g \to g,\\ &l \to l. \end{aligned}$$

Hence, $R_{i+1,j}$ is in canonical form.

Subcase 2.2. The algorithm introduces a horizontal line segment.

For this case it is simple to verify that point p_{i+1} must be located above the bottom side of $R_{i,j}$ and below the top side of $R_{i,j}$. Therefore the rectangles are transformed as follows:

$$\begin{aligned} &a \to A,\\ &c \to h|C,\\ &f \to F,\\ &g \to G,\\ &l \to L. \end{aligned}$$

Hence, in all cases we obtain a rectangle $R_{i+1,j}$ in canonical form.

Subcase 2.3. The algorithm introduces a left corner segment.

Clearly, in order to introduce some segment inside $R_{i,j}$, point p_{i+1} must be located above the bottom side of $R_{i,j}$. If point p_{i+1} is not to the left of the right side of $R_{i,j}$, then only a horizontal line segment is introduced inside $R_{i,j}$ and the proof follows the same lines as the one for subcase 2.2. On the other hand if point p_{i+1} is to the left of the right side of $R_{i,j}$, then we claim that the algorithm will not introduce the corner point of the left corner inside $R_{i,j}$ during the $(i+1)$st iteration, i.e., only the vertical portion of a left corner segment can be introduced inside $R_{i,j}$. We prove this by contradiction. Suppose the algorithm introduces the corner point inside $R_{i,j}$ during the ith iteration. We know from the above

restrictions on the location of point p_{i+1} and the scanning algorithm, that point p_{i+1} must be located above the top side of $R_{i,j}$, to the right of any line segment inside $R_{i,j}$, and to the left of the right side of $R_{i,j}$. Therefore, the left corner segment enters the rectangle on the top side of $R_{i,j}$ and ends on some existing internal line segment or on the left side of $R_{i,j}$. But this is not a minimum length path to a previously introduced line segment (L_1 metric). This contradicts our scanning algorithm. Hence, the corner point of a left corner segment cannot be introduced inside $R_{i,j}$. So the only remaining case is when a portion of the vertical section of the left corner is the only segment introduced inside $R_{i,j}$. The proof now follows the same arguments as the ones for subcase 2.1.

Subcase 2.4. The algorithm introduces a right corner segment.

In order to introduce some segment inside $R_{i,j}$ point p_{i+1} must be located below the top side of $R_{i,j}$. If point p_{i+1} is not located above the bottom side of $R_{i,j}$, only a vertical line segment is introduced inside $R_{i,j}$. Such a vertical line segment cannot be to the left of any line segment inside $R_{i,j}$. The remaining part of the proof for this case follows the same lines as the one for subcase 2.1. On the other hand, if point p_{i+1} is located above the bottom side of $R_{i,j}$ then either only a horizontal line segment or the corner point of the right corner segment is introduced inside $R_{i,j}$. The former case is identical to subcase 2.2. In the latter case we know from the scanning algorithm that the right corner segment must intersect the top side of R at a point located to the right of the rightmost segment inside $R_{i,j}$, or it must intersect the corner point of the left joint. The possible transformations are given below:

$$\begin{aligned} &a \to d, \\ &c \to i \mid j, \\ &f \to m, \\ &g \to n, \\ &l \to s. \end{aligned}$$

Hence, in all cases we obtain a rectangle $R_{i+1,j}$ in canonical form.

Case 3. One of the line segments inside $R_{i,j}$ intersects the right side of R_{ij} (type b, d, e, h–k, m–z, Δ, A–Z).

For this case we know that point p_{i+1} must not be located to the left of the right side of $R_{i,j}$. There are four cases depending on the type of segment introduced by the algorithm.

Subcase 3.1. The algorithm introduces a vertical line segment.

Since p_{i+1} is not located to the left side of $R_{i,j}$, we know that $R_{i+1,j} = R_{i,j}$ and by the induction hypothesis it is in canonical form.

Subcase 3.2. The algorithm introduces a horizontal line segment.

Introduction of a horizontal line segment inside $R_{i,j}$ results in the following set of rectangles:

$$
\begin{array}{ll}
\Delta \to \Delta, & \\
A \to A, & \\
B \to E|B, & b \to e|B, \\
C \to H|C, & \\
D \to K|D, & d \to k|D, \\
E \to E, & e \to e|E, \\
F \to F, & \\
G \to G, & \\
H \to H, & h \to h|H, \\
I \to O|I, & i \to o|I, \\
J \to Q|P|J, & j \to q|p|J, \\
K \to K, & k \to k|K, \\
L \to L, & \\
M \to T|M, & m \to t|M, \\
N \to U|N, & n \to u|N, \\
O \to O, & o \to o|O, \\
P \to V|P, & p \to v|p|P, \\
Q \to Q|V, & q \to q|v|Q, \\
R \to X|W|R, & r \to x|w|R, \\
S \to Y|S, & s \to y|S, \\
T \to T, & t \to t|T, \\
U \to U, & u \to u|U, \\
V \to V, & v \to v|V, \\
W \to Z|W, & w \to z|w|W, \\
X \to X|Z, & x \to x|z|X, \\
X \to Y, & y \to y|Y, \\
Z \to Z, & z \to z|Z.
\end{array}
$$

Hence, we obtain a rectangle $R_{i+1,j}$ in canonical form.

Subcase 3.3. The algorithm introduces a right corner segment.

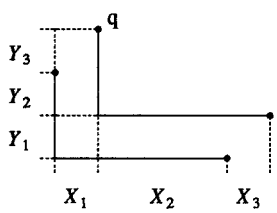
If the corner point of the right corner segment is not introduced inside $R_{i,j}$, but a horizontal line segment is introduced in $R_{i,j}$, the proof is similar to the one for Subcase 3.1. On the other hand, if only the vertical portion is introduced we can prove that rectangle $R_{i,j}$ is type h, j, p, q, or v. In each of these cases the transformations are given below:

$$\begin{aligned} &\text{h} \to \text{F}, \\ &\text{j} \to \text{m}, \\ &\text{p} \to \text{M}, \\ &\text{q} \to \text{t}, \\ &\text{v} \to \text{T}. \end{aligned}$$

We prove the following claim to eliminate the set of rectangles on which the scanning algorithm cannot introduce inside $R_{i,j}$ the corner point of a right corner segment.

CLAIM. *If there is a right joint inside $R_{i,j}$, the scanning algorithm cannot introduce the corner point of a right corner segment inside $R_{i,j}$ unless the right corner segment intersects the corner point of the left joint in a rectangle without a right semijoint, a partial vertical cut, or a vertical cut.*

PROOF. We prove this claim by contradiction. Suppose that there is a right joint inside $R_{i,j}$ and the algorithm introduces inside $R_{i,j}$ the corner point of a right corner segment that does not intersect the corner point of the left joint. From the scanning algorithm we know that it is not possible for the right corner segment to include all of the right joint nor can it intersect the right joint. So it must be that the right corner segment is introduced inside the right joint.



We know from the scanning algorithm that when the new right corner segment is introduced, it must be that $Y_2 + Y_3 \leq X_1$ and it must be that q is either the corner point of a left corner segment or one of the points in $p_1, \ldots, p_i$. Therefore, point $q \in C(I)$ when the right joint was introduced. Because of this, we know that when the scanning algorithm introduced the existing right joint, it must have been that $X_1 + Y_1 + Y_2 \leq Y_1 + Y_2 + Y_3$. Combining both inequalities we know that $Y_2 + Y_3 \leq X_1 \leq Y_3$. A contradiction since Y_2 is greater than zero. This completes the proof of the claim. □

From the above claim it is simple to verify that the corner point of a right corner segment can only be introduced in rectangles of type Δ, A, C, F-J, L, O-Q, V, h-j, o-q, and v. Introducing the corner point of the right corner segment in $R_{i,j}$ results in the following set of rectangles:

$$\Delta \to \mathrm{B},$$
$$\mathrm{A} \to \mathrm{D},$$
$$\mathrm{C} \to \mathrm{J} \mid \mathrm{I},$$
$$\mathrm{F} \to \mathrm{M},$$
$$\mathrm{G} \to \mathrm{N},$$
$$\mathrm{H} \to \mathrm{P} \mid \mathrm{O}, \qquad \mathrm{h} \to \mathrm{p} \mid \mathrm{o},$$
$$\mathrm{I} \to \mathrm{R}, \qquad \mathrm{i} \to \mathrm{r},$$
$$\mathrm{J} \to \mathrm{R}, \qquad \mathrm{j} \to \mathrm{r},$$
$$\mathrm{L} \to \mathrm{S},$$
$$\mathrm{O} \to \mathrm{W}, \qquad \mathrm{o} \to \mathrm{w},$$
$$\mathrm{P} \to \mathrm{W}, \qquad \mathrm{p} \to \mathrm{w},$$
$$\mathrm{Q} \to \mathrm{X}, \qquad \mathrm{q} \to \mathrm{x},$$
$$\mathrm{V} \to \mathrm{Z}, \qquad \mathrm{v} \to \mathrm{z}.$$

Hence, rectangle $R_{i+1,j}$ is in canonical form.

Subcase 3.4. The algorithm introduces a left corner segment.

Since point p_{i+1} is not located to the left of the right side of $R_{i,j}$, it must be that the corner point of the left corner segment is not introduced inside $R_{i,j}$. Hence, only a horizontal line segment can be introduced inside $R_{i,j}$ at this iteration. The proof for this case is similar to the one in Subcase 3.2.

This completes the proof for this case and the lemma. □

In what follows when we refer to rectangle R we mean any rectangle in set $\{R_{n,j}\}$. To obtain a rectangular partition from $E_{\mathrm{opt}}(I) \cup C(I)$, we must eliminate the "effects" (nonrectangular partition) of the joints and the semijoint by introducing a set of line segments. All of the line segments that we introduce form the set $D(I)$. Each of the sides of rectangle R is either a part of the boundary or a line segment from E_{opt}. Note that some line segments in $C(I)$ may overlap with the sides of R. The line segments in $C(I) - E_{\mathrm{opt}}(I)$ are *labeled A*. To prove that $L(E(I)) \leq 3L(E_{\mathrm{opt}}(I))$ we show that $L(D(I) \cup (C(I) - E_{\mathrm{opt}}(I))) \leq 2L(E_{\mathrm{opt}}(I))$. We prove this bound by showing that for each rectangle R, $\mathrm{LEN}(R) \leq \mathrm{OPT}(R)$, where $\mathrm{LEN}(R)$ is the sum of the length of the edges from $D(I)$ inside rectangle R plus the length of the line segments labeled A inside rectangle R; and $\mathrm{OPT}(R)$ is the sum of the length of the edges from $E_{\mathrm{opt}}(I)$ that are sides of R.

We say that component $C \in R$ if component C is inside R. When we refer to rectangle R, we use X as its width and Y as its height. We also use the symbols defined below to refer to the length of the different parts of the rectangle.

$J_{vl} = L_l =$ length of the vertical portion of the left joint, if LJ $\in R$; and 0 otherwise.
$J_{hl} = T_l = B_l =$ length of the horizontal portion of the left joint, if LJ $\in R$; and 0 otherwise.
$J_{vr} = R_t =$ length of the vertical portion of the right joint, if RJ $\in R$; and 0 otherwise.
$J_{hr} = T_r = B_r =$ length of the horizontal portion of the right joint, if RJ $\in R$; and 0 otherwise.
$S_h =$ length of the horizontal portion of the right semijoint, if RS $\in R$; and 0 otherwise.
$S_v =$ distance from the horizontal portion of the right semijoint to the top side of R, if RS $\in R$; and 0 otherwise.
$T_m = B_m = X - T_l$, if LJ $\in R$; and 0 otherwise.

$$R_m = \begin{cases} S_v, & \text{if RS} \in R; \\ \text{distance from the lowest partial horizontal cut to the top side of } R, & \text{if LJ, PHC-NRJNV} \in R \text{ and RS, PVC, VC} \notin R; \\ R_t, & \text{otherwise.} \end{cases}$$

$$T_v = B_v = \begin{cases} \text{distance from the left side of } R \text{ to the rightmost vertical cut,} & \text{if VC} \in R; \\ \text{distance from the left side of } R \text{ to the vertical portion of the left joint,} & \text{if PVC} \in R \text{ and VC} \notin R; \\ 0, & \text{otherwise.} \end{cases}$$

$$B_h = \begin{cases} \text{distance from the rightmost vertical cut to the right side of } R, & \text{if VC} \in R; \\ \text{distance from the vertical portion of the left joint to the right side of } R, & \text{if PVC} \in R \text{ and VC} \notin R; \\ X, & \text{if HC} \in R \text{ and VC, PVC} \notin R; \\ 0, & \text{otherwise.} \end{cases}$$

$$R_h = L_h = \begin{cases} \text{distance from the topmost horizontal cut to the bottom side of } R, & \text{if HC} \in R; \\ \text{distance from the topmost partial horizontal cut (PHC-NRJV) to the bottom side of } R, & \text{if PHC-NRJV} \in R; \\ 0, & \text{otherwise.} \end{cases}$$

$L_b = Y - L_t - L_h$.

$R_b = Y - R_m - R_h$.

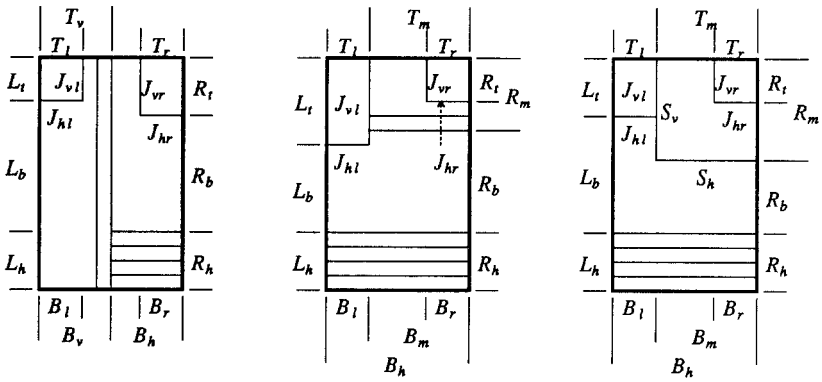


Fig. 2.5

Before proving our approximation bound, we prove some useful properties that are satisfied by every rectangle $R \in \{R_{n,j}\}$. At each iteration of the scanning algorithm a new set of line segments is introduced and one point in these line segments intersects either a line segment introduced during a previous iteration or the boundary (which cannot be the right-hand side boundary). Hence,

(1) there cannot be a path, all of it labeled A, that joins two boundaries; and
(2) if the right side of R is part of the boundary then no line segment labeled A can intersect it.

It is simple to prove that

(3) each of the joints is introduced by one iteration of the scanning algorithm and the segments that form the right semijoint are introduced by one iteration of the scanning algorithm.

It is simple to prove that when a point is connected directly to the boundary, it is connected by a straight line segment. Therefore,

(4) if there is a left joint, then neither the left side nor the top side of rectangle R is a part of the boundary;
(5) if there is a right joint, then neither the right side nor the top side of rectangle R is a part of the boundary; and
(6) if there is a right semijoint, then neither the left, top, nor right side of R is a part of the boundary.

Let us now establish a bound on the length of the partial horizontal cuts located inside the right joint (PHC-RJ). It is simple to prove that every pair of these cut lines must be at least T_r units apart and all of such cut segments must be at least T_r units from the horizontal portion of the right joint. Hence,

(7) if there is a right joint and it includes partial horizontal cuts (PHC-RJ), the total length of these cut segments is at most R_t.

Let us now consider rectangles with the left joint and without the right semijoint, a partial vertical cut, and vertical cuts (LJ $\in R$ and RS, PVC, VC $\notin R$). It is simple to prove that the horizontal cut lines that intersect the left joint (PHC-NRJNV) must be at least T_m units apart. Hence, if there is no right joint, the sum of the length of all the PHC-NRJNV segments is at most $T_m + R_m$. On the other hand, if there is a right joint, we know from the scanning algorithm that the distance between the horizontal portion of the right joint and the topmost PHC-NRJNV segment is at least $J_{vr} + J_{hr}$. From the above observations and (7) it is simple to show that

(8) if there are no partial vertical cuts, verticals cuts, or a right semijoint, and there is a left joint (LJ $\in R$ and PVC, VC, RS $\notin R$), then all the line segments to the right of the left joint (PHC-NRJNV, RJ, and PHC-RJ) have length $\leq T_m + R_m$.

Using similar observations, it is simple to show that

(9) if there is a left joint and a right semijoint (LJ, RS $\in R$), then all the line segments inside the right semijoint (PHC-NRJNV, RJ, and PHC-RJ) have length $\leq R_m$.

Since all the PHC-NRJV cuts and all the horizontal cuts must be at least B_h units apart and since any of these line segments must be at least B_h units from the bottom side of the rectangle when such a side is part of the boundary, we know that

(10) if the bottom side of R is not part of the boundary, the length of all the PHC-NRJV cuts or the horizontal cuts is at most $B_h + R_h$. If the bottom side of R is part of the boundary, the length of all the PHC-NRJV cuts or the horizontal cuts is at most R_h.

From the scanning algorithm it is simple to verify that every pair of vertical cuts and the partial vertical cut must be at least Y units apart and any of these lines must be at least Y units from the left side of the rectangle when such a side is part of the boundary. Hence,

(11) if the left side of R is not part of the boundary then the length of the line segments composing the left joint (LJ) plus the length of the partial vertical cut (PVC) plus the length of all vertical cuts (VC) is at most $Y + B_v$. If the left side of R is part of the boundary then the length of all the vertical cuts (VC) is at most B_v.

Our notation for the figures in this section is defined as follows: thin line segments denote line segments labeled A; dotted line segments denote line segments from $D(I)$ in R; thick line segments with slashes (/) indicate the segment is part of the boundary; thick line segments with ×'s indicate that the line segment is either in $E_{\text{opt}}(I)$ or is part of the boundary; and thick line segments (without any symbol) indicate line segments from $E_{\text{opt}}(I)$. In the proof of Lemma 2.2 we use $D(R)$ to denote the line segments from $D(I)$ in R and $L(D(R))$ to denote the total length of the edges in $D(R)$.

LEMMA 2.2. *For all* R, $\mathrm{LEN}(R) \leq \mathrm{OPT}(R)$.

PROOF. There are two cases.

Case 1. VC, PVC $\notin R$ (rectangle is of type δ, Δ, b, c, B, C, e, E, h-j, H-J, o-r, O-R, v-x, V-X, z, and Z).

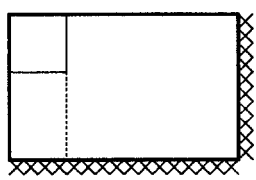
There are two cases depending on whether or not there are horizontal cuts.

Subcase 1.1. HC, PHC-NRJV $\notin R$ (rectangle is type δ, b, c, e, h-j, o-r, v-x, and z).

We only consider the case when there are line segments inside R, as otherwise we know that $\mathrm{LEN}(R) = 0$ and $\mathrm{LEN}(R) \leq \mathrm{OPT}(R)$. There are five cases depending on which components are located inside R.

Subcase 1.1.1. LJ $\in R$ and RJ, RS, PHC-RJ, PHC-NRJNV $\notin R$ (type c).

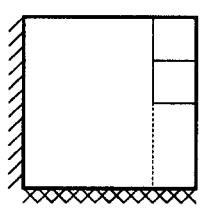
Since there is a left joint, the left and the top sides cannot be boundaries (4). Clearly, $J_{hl} = T_l$, $J_{vl} = L_t$, and $L(D(R)) = L_b$. Hence, $\mathrm{LEN}(R) \leq \mathrm{OPT}(R)$. This completes the proof for this subcase.



Subcase 1.1.2. RJ $\in R$ and LJ, RS, PHC-NRJNV $\notin R$ (type b and e).

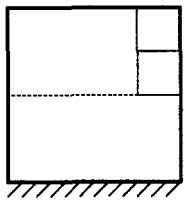
Since there is a right joint, we know from (5) that the right and top sides of R are in $E_{\mathrm{opt}}(I)$. We know from (7) that the sum of the length of all the partial horizontal cuts inside the right joint is less than or equal to R_t. There are three cases depending on whether or not the other sides of R are part of the boundary.

Subcase 1.1.2.1. The left side of R is part of the boundary.



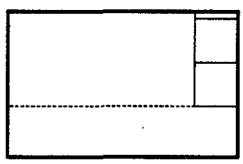
Since the left side of R is part of the boundary and since the right joint is introduced by one iteration of the scanning algorithm (3), we know that $J_{hr} + J_{vr} \leq X$. Clearly, $L(D(R)) = R_b$. Hence $\mathrm{LEN}(R) \leq \mathrm{OPT}(R)$. This completes the proof of this subcase.

Subcase 1.1.2.2. The left side of R is in $E_{\text{opt}}(I)$ and the bottom side of R is part of the boundary.



Since the bottom side of R is part of the boundary and since the right joint is introduced by one step of the scanning algorithm (3), we know that $J_{hr} + J_{vr} \leq R_b$. Clearly, $L(D(R)) = X - T_r$. Hence $\text{LEN}(R) \leq \text{OPT}(R)$. This completes the proof for this subcase.

Subcase 1.1.2.3. The left side and the bottom side of R are in $E_{\text{opt}}(I)$.



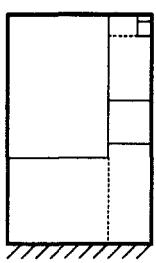
Clearly, the four sides of R belong to $E_{\text{opt}}(I)$, $J_{hr} = T_r$, $J_{vr} = R_t$, and $L(D(R)) = X - T_r$. Hence $\text{LEN}(R) \leq \text{OPT}(R)$. This completes the proof for this subcase and Subcase 1.1.2.

Subcase 1.1.3. LJ, PHC-NRJNV$\in R$ and RS$\notin R$ (type h, p, and v).

We know from (2) and (4) that the left, top, and right sides of R are in E_{opt} and we know from (8) that the length of all the line segments introduced by the scanning algorithm to the right of the left joint is $\leq T_m + R_m$. Clearly, $J_{hl} = T_l$ and $J_{vl} = L_t$. There are two cases depending on whether or not the bottom side of R is part of the boundary.

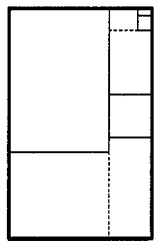
Subcase 1.1.3.1. The bottom side of R is part of the boundary.

Since there are partial horizontal cuts that intersect the left joint, we know from the scanning algorithm that $T_m \leq R_b$. Therefore, $L(D(R)) < L_b + R_b$. Hence, $\text{LEN}(R) \leq \text{OPT}(R)$. This completes the proof for Subcase 1.1.3.1.



Subcase 1.1.3.2. The bottom side of R is in $E_{\text{opt}}(I)$.

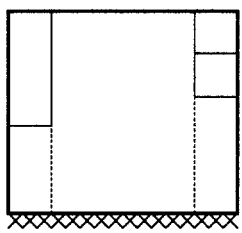
Clearly, $L(D(R)) < L_b + T_m$. Hence, $\text{LEN}(R) \leq \text{OPT}(R)$. This completes the proof for Subcase 1.1.3.



Subcase 1.1.4. LJ, RJ $\in R$ and RS, PHC-NRJNV $\notin R$ (type j and q).

Since the two joints are present we know from (4) and (5) that the left, top, and right sides of R are in $E_{\text{opt}}(I)$. We know from (7) that the sum of the length of all the partial horizontal cuts inside the right joint is less than or equal to R_t. Clearly $J_{hl} = T_l$ and $J_{vl} = L_t$. There are two cases depending on the relative values of J_{vl} and J_{vr}.

Subcase 1.1.4.1. $J_{vl} > J_{vr}$.

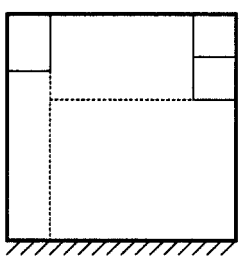


Clearly, $L(D(R)) = L_b + R_b$. Since both of the joints are introduced by a single step of the scanning algorithm (3) and the right joint appears to the right of the left joint, it must have been that the scanning algorithm introduced the right joint after the left joint and $J_{vr} + J_{hr} \leq (X - T_l)$. Hence, $\text{LEN}(R) \leq \text{OPT}(R)$. This completes the proof for this subcase.

Subcase 1.1.4.2. $J_{vl} \leq J_{vr}$.

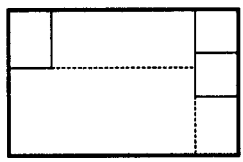
There are two caases depending on whether or not the bottom side of R is part of the boundary.

Subcase 1.1.4.2.1. The bottom side of R is part of the boundary.



Clearly, $L(D(R)) = L_b + (X - T_r - T_l)$. Since the bottom side of R is part of the boundary and since the right joint is introduced by one step of the scanning algorithm (3), we know that $J_{vr} + J_{hr} \leq R_b$. Hence, $\text{LEN}(R) \leq \text{OPT}(R)$. This completes the proof for this subcase.

Subcase 1.1.4.2.2. The bottom side of R belongs to $E_{\text{opt}}(I)$.



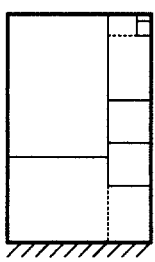
Clearly, the four sides of R belong to $E_{\text{opt}}(I)$ and $L(D(R)) = (X - T_l - T_r) + R_b$. Since both of the joints are introduced by a single step of the scanning algorithm (3) and the right joint appears to the right of the left joint, it must have been that the scanning algorithm introduced the right joint after the left joint and $J_{hr} + J_{vr} \leq (X - T_l) + (J_{vr} - J_{vl})$. Hence, $\text{LEN}(R) \leq \text{OPT}(R)$. This completes the proof of this subcase and Subcase 1.1.4.

Subcase 1.1.5. LJ, RS $\in R$ (type i, o, r, w, x, and z).

Since there is a right semijoint, we know from (6) that the left, top, and right sides of R are in $E_{\text{opt}}(I)$. Clearly, $J_{hl} = T_l$ and $J_{vl} = L_t$. We know from (9) that the length of all the line segments introduced by the scanning algorithm inside the right semijoint is $\leq R_m$. There are two cases depending on whether or not the bottom side of R is part of the boundary.

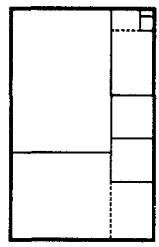
Subcase 1.1.5.1. The bottom side of R is part of the boundary.

Since RS $\in R$ and the bottom side of R is part of the boundary we know from the scanning algorithm that the two segments forming the right semijoint have length $\leq R_b$. Clearly, $L(D(R)) < L_b + T_m$. Hence, $\text{LEN}(R) \leq \text{OPT}(R)$. This completes the proof for Subcase 1.1.5.1.



Subcase 1.1.5.2. The bottom side of R is in $E_{\text{opt}}(I)$.

The length of the segments that form the right semijoint is $T_m + (R_m - L_t)$. Clearly, $L(D(R)) < (L_b - (R_m - L_t)) + B_m$. Hence, $\text{LEN}(R) \leq \text{OPT}(R)$. This completes the proof for this subcase and Subcase 1.1.

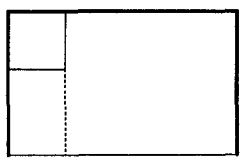


Subcase 1.2. HC $\in R$ (type Δ, B, C, E, H-J, O-R, V-X, and Z).

Since there are horizontal cuts, we know from (2) that the right side of R is not part of the boundary. We know from (10) that the length of all the horizontal cuts is at most $B_h + R_h$ if the bottom side of R is not part of the boundary and it is at most R_h if the bottom side of R is part of the boundary. Hence, if there are no line segments inside R other than the horizontal cuts $\text{LEN}(R) \leq \text{OPT}(R)$. For simplicity, all the figures in this subcase will be drawn including only the topmost horizontal cut. The remaining cases are treated separately.

Subcase 1.2.1. LJ $\in R$ and RJ, RS, PHC-RJ, PHC-NRJNV $\notin R$ (type C).

Since there is a left joint, we know from (4) that the left and top sides of R cannot be part of the boundary. Clearly, $L(D(R)) = L_b$, $J_{hl} = T_l$, and $J_{vl} = L_l$. Hence, $\text{LEN}(R) \leq \text{OPT}(R)$. This completes the proof for this subcase.

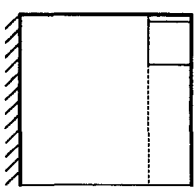


Subcase 1.2.2. RJ $\in R$ and LJ, RS, PHC-NRJNV $\notin R$ (type B and E).

Since there is a right joint, we know from (5) that the right and top sides of R cannot be part of the boundary. We know from (7) that the length of the partial horizontal cuts inside the right joint have length $\leq R_t$. There are two cases depending on whether or not the left side R is part of the boundary.

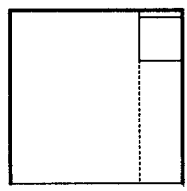
Subcase 1.2.2.1. The left side of R is part of the boundary.

Since the left side is part of the boundary and the right joint is introduced by a single step of the scanning algorithm (3), we know that $J_{vr} + J_{hr} \leq X$. Clearly, $L(D(R)) = R_b$. Hence, $\text{LEN}(R) \leq \text{OPT}(R)$. This completes the proof for this subcase.



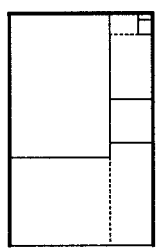
Subcase 1.2.2.2. The left side of R is not part of the boundary.

Clearly, $L(D(R)) = R_b$, $J_{vr} = R_t$, and $J_{hr} = T_r$. Hence, $\text{LEN}(R) \leq \text{OPT}(R)$. This completes the proof for this subcase and Subcase 1.2.2.



Subcase 1.2.3. LJ, PHC-NRJNV$\in R$ and RS$\notin R$ (type H, P, and V).

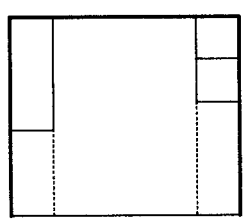
Since there is a left joint and there are partial horizontal cuts, we know from (2) and (4) that the left, top, and right sides of R belong to $E_{\text{opt}}(I)$. Clearly, $J_{hl} = T_l$ and $J_{vl} = L_t$. Since there are partial horizontal cuts that intersect the left joint, we know that $B_m \leq R_b$. Therefore, $L(D(R)) < L_b + R_b$. We know from (8) that the length of all the line segments to the right of the left joint and above the topmost horizontal cut is at most $T_m + R_m$. Hence, $\text{LEN}(R) \leq \text{OPT}(R)$. This completes the proof for this subcase.



Subcase 1.2.4. LJ, RJ$\in R$ and RS, PHC-NRJNV$\notin R$ (type J and Q).

Since both joints are present, we know from (4) and (5) that the left, top, and right sides of R are in $E_{\text{opt}}(I)$. Clearly, $J_{hl} = T_l$, $J_{vl} = L_t$, and we know from (7) that the sum of the length of all the partial horizontal cuts inside the right joint is less than or equal to R_t. There are two cases depending on the relative values of J_{vl} and J_{vr}.

Subcase 1.2.4.1. $J_{vl} > J_{vr}$.

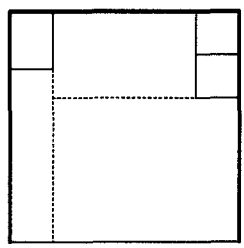


Clearly, $L(D(R)) = L_b + R_b$. Since both of the joints are introduced by a single step of the scanning algorithm (3) and the right joint appears to the right of the left joint, it must have been that the scanning algorithm introduced the right joint after the left joint and $J_{vr} + J_{hr} \leq (X - T_l)$. Hence $\text{LEN}(R) \leq \text{OPT}(R)$. This completes the proof for this subcase.

Subcase 1.2.4.2. $J_{vl} \leq J_{vr}$.

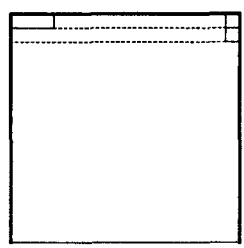
There are two cases depending on whether or not the topmost horizontal cut was introduced by the scanning algorithm before the right joint.

Subcase 1.2.4.2.1. The topmost horizontal cut in R was introduced by the scanning algorithm before the right joint.



Clearly, $L(D(R)) = L_b + (X - T_l - T_r)$. Since the topmost horizontal line segment was introduced before the right joint and since the right joint is introduced by a single step of the scanning algorithm, we know that $J_{vr} + J_{hr} \leq R_b$. Hence $\text{LEN}(R) \leq \text{OPT}(R)$. This completes the proof for this subcase.

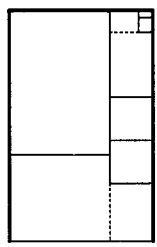
Subcase 1.2.4.2.2. The topmost horizontal cut in R was introduced by the scanning algorithm after the right joint.



Since the topmost horizontal cut in R was introduced by one iteration of the scanning algorithm after the right joint, we know that $X \leq R_b$. Hence, $L(D(R)) < 2R_b$. Since both of the joints are introduced by a single step of the scanning algorithm (3) and the right joint appears to the right of the left joint, it must have been that the scanning algorithm introduced the right joint after the left joint and $J_{hr} + J_{vr} \leq (X - T_l) + (R_t - L_t)$. Hence, $\text{LEN}(R) \leq \text{OPT}(R)$. This completes the proof of this subcase and Subcase 1.2.4.

Subcase 1.2.5. LJ, RS $\in R$ (type I, O, R, W, X, and Z).

Since RS $\in R$, we know from (6) that the left, top, and right side of R belong to $E_{\text{opt}}(I)$. Clearly, $L(D(R)) = R_b + (X - T_l - T_r)$, $J_{hl} = T_l$, $J_{vl} = L_t$, and the length of the vertical portion of the right semijoint is equal to $L_b - R_b$. Since RS $\in R$ and there are horizontal cuts, we know from the scanning algorithm that $S_h < R_b$. We know from (9) that the length of all the line segments introduced by the algorithm that are located inside the right semijoint is at most R_m. Hence, $\text{LEN}(R) \leq \text{OPT}(R)$. This completes the proof for this subcase.

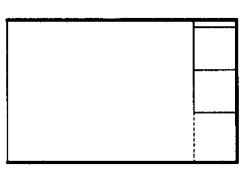


Case 2. VC $\in R$ and/or PVC $\in R$ (rectangle is of type a, A, d, D, f, g, F, G, k-n, K-N, s-u, S-U, y, and Y).

For simplicity, all the figures in this case will be drawn including only the topmost partial horizontal cut (PHC-NRJV) and either the rightmost vertical cut or the partial vertical cut. The latter case is when VC $\notin R$. We know from (11) that if the left side of R is not part of the boundary then the length of the line segments composing the left joint plus the length of the partial vertical cut plus the length of all the vertical cuts is at most $Y + B_v$; otherwise it is at most B_v. There are two cases depending on whether or not there are PHC-NRJV cuts.

Subcase 2.1. PHC-NRJV $\notin R$ (rectangles types a, d, f, g, k-n, s-u, and y).

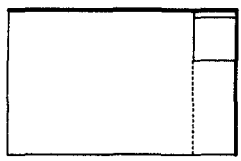
If there is no right joint inside R, it is simple to see that $\text{LEN}(R) \leq \text{OPT}(R)$. So let us assume that there is a right joint inside R. For this case, it is simple to prove that $J_{hr} + J_{vr} \leq (X - T_v)$. The sum of the length of all the partial horizontal cuts inside the right joint is less than or equal to R_t [see (3)]. Clearly, $L(D(R)) = R_b$. Hence, $\text{LEN}(R) \leq \text{OPT}(R)$. This completes the proof for this subcase.



Subcase 2.2. PHC-NRJV $\in R$ (rectangle types A, D, F, G, K-N, S-U, and Y).

We know from (10) that if the bottom side of R is not part of the boundary, the length of all the PHC-NRJV cuts is at most $B_h + R_h$; and if the bottom side

of R is part of the boundary, the length of all the PHC-NRJV cuts is at most B_h. If there is no right joint then it is simple to verify that $\text{LEN}(R) \leq \text{OPT}(R)$. So let us assume that $\text{RJ} \in R$. For this case it is simple to prove that $J_{vr} + J_{hr} \leq (X - T_v)$. We know from (7) that the length of all partial horizontal cuts inside the right joint is $\leq R_t$. By construction, $L(D(R)) = R_b$. Hence, $\text{LEN}(R) \leq \text{OPT}(R)$. This completes the proof for this subcase, Case 2, and the lemma.



□

THEOREM 2.2. *For any instance I for the* RG-P *problem, algorithm* TRANS *generates a solution* $E_{\text{apx}}(I)$ *such that* $L(E_{\text{apx}}(I)) \leq 3L(E_{\text{opt}}(I))$.

PROOF. Follows from Lemma 2.2 and the discussion preceding Lemma 2.1. □

The tightness of the approximation bound 3 for our algorithm TRANS is established by the following theorem.

THEOREM 2.3. *For any small* $\varepsilon > 0$, *there exists an* RG-P *problem instance for which* $L(E_{\text{apx}}(I)) > 3L(E_{\text{opt}}(I)) - \varepsilon$.

PROOF. Let $\varepsilon = 2^{-k}$ and let $\varepsilon' = \varepsilon/(k+2) = 2^{-k}/(k+2)$. The origin (lower-left corner of the rectangle) is $(0, 0)$, $X = 1$, and $Y = k(1+\varepsilon)$. The set P contains $k(k+2)-1$ points defined as follows:

$$p_{1,i} = (x_{1,i}, y_{1,i}) = (1-\varepsilon, i(1+\varepsilon)), \qquad 1 \leq i \leq k-1;$$

$$p_{2,i} = (x_{2,i}, y_{2,i}) = \begin{cases} (1-\varepsilon+\varepsilon', 1) & \text{for } i = 1; \\ (1-\varepsilon+\varepsilon', y_{2,i-1}+1+\varepsilon) & \text{for } 1 < i \leq k; \end{cases}$$

and, for $1 \leq i \leq k$,

$$p_{3,i,j} = (x_{3,i,j}, y_{3,i,j}) = \begin{cases} (x_{2,i}+\varepsilon', y_{2,i}-1/2) & \text{for } j = 1; \\ (x_{2,i}+j\varepsilon', y_{3,i,j-1}-2^{-j}) & \text{for } 1 < j \leq k. \end{cases}$$

It is easy to see that all these points are located in a belt area close to the right boundary of the rectangle and the distance between any two points is greater than ε. It is simple to show that for any small ε,

$$L(E_{\text{opt}}(I)) = k(1+\varepsilon) + k(k+1)\varepsilon = k + k(k+2)*2^{-k},$$

and

$$\begin{aligned} L(E_{\text{apx}}(I)) &= [3/2+k(\varepsilon-\varepsilon')+\varepsilon]+(k-2)[2+(k+1)\varepsilon] \\ &\quad +[3/2+\varepsilon+2\varepsilon+(k-1)(\varepsilon-2\varepsilon')]+(k-1)[1-\varepsilon] \\ &= 3k-2+(k^2+2)\varepsilon-(3k-2)\varepsilon' \\ &= 3k-2+(k^2+2)*2^{-k}-((3k-2)/(k+2))*2^{-k} \\ &= 3k-2+(k^2-(k-6)/(k+2))*2^{-k}. \end{aligned}$$

Figure 2.6 shows $C(I)$, $E_{\text{apx}}(I)$, and $E_{\text{opt}}(I)$ for the case when $k=3$. Combining these bounds we obtain

$$\begin{aligned} &\lim_{k\to\infty} (L(E_{\text{apx}}(I))/L(E_{\text{opt}}(I))) \\ &\quad = \lim_{k\to\infty} \{[3k-2+(k^2-(k-6)/(k+2))*2^{-k}]/[k+(k^2+2k)*2^{-k}]\} \\ &\quad = 3-\lim_{k\to\infty} \{[2^{k+1}+2k^2+6k+1-8/(k+2)]/[k2^k+k^2+2k]\} \\ &\quad = 3. \end{aligned}$$

This completes the proof of the theorem. □

We defined our algorithm in such a way that when a point (dot) is connected to another point (×) the lines introduced are defined in Figure 2.7(a). As a result of this operation joints are only introduced on the top side of R (Figure 2.7(a)). We could also define algorithms in which the connections are defined in Figure 2.7(b)-(d). In each of these cases the joints will appear only on the right, left, and bottom side of R. It is interesting to note that if the line segments introduced by the scanning algorithm are such that the joints appear only on the right side of R, then LEN(R) is sometimes greater than OPT(R). A rectangle for which this holds true is given in Figure 2.8(a). If the joints are only introduced on the left side of R, it seems that we can also prove the approximation bound of 3. However, by using analysis similar to the one in this section we cannot prove any approximation bound smaller than 3. A rectangle R for which LEN(R) is almost equal to OPT(R) is given in Figure 2.8(b). In this case it is much harder to characterize the different types of rectangles formed by $E_{\text{opt}}(I) \cup C(I)$. The main difficulty is that there might be a straight line segment inside the rectangle that was introduced by several steps of the scanning algorithm. When the joints only appear on the bottom side of R, we obtain a set of rectangles similar to the ones in Figures 2.3 and 2.4. The analysis in this case is similar to the one in this section.

3. Discussion. Our algorithm has an approximation bound that is not so small, however, we believe that the solutions generated by our algorithms are usually

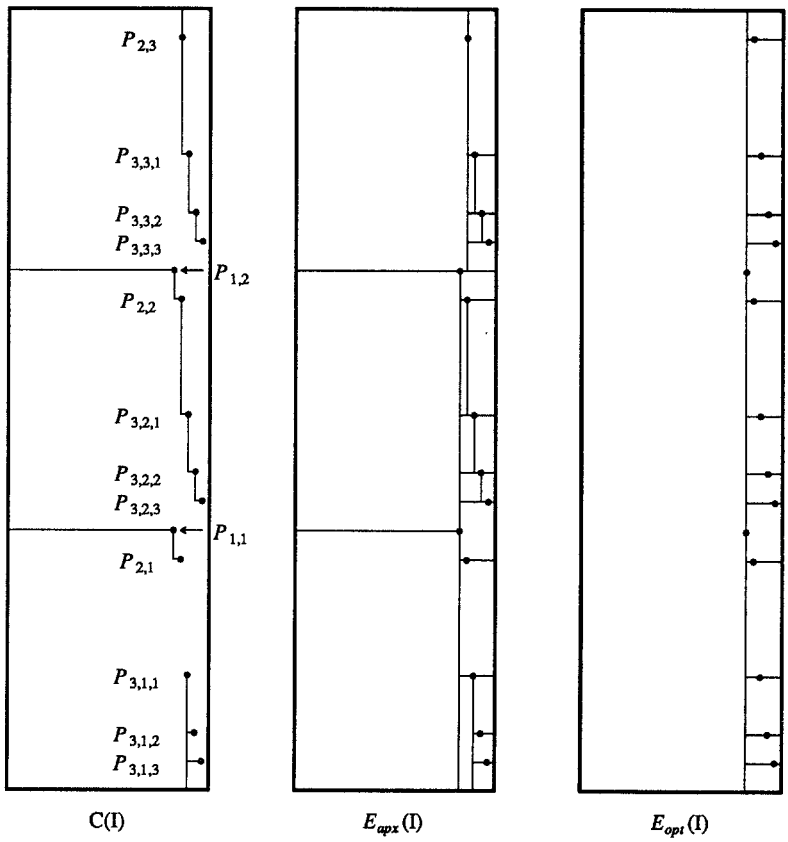

$P_{2,3}$
$P_{3,3,1}$
$P_{3,3,2}$
$P_{3,3,3}$
$P_{1,2}$
$P_{2,2}$
$P_{3,2,1}$
$P_{3,2,2}$
$P_{3,2,3}$
$P_{1,1}$
$P_{2,1}$
$P_{3,1,1}$
$P_{3,1,2}$
$P_{3,1,3}$
C(I)
E_{apx}(I)
E_{opt}(I)

Fig. 2.6

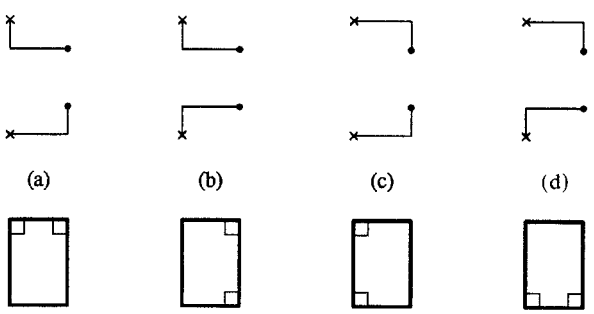

(a)
(b)
(c)
(d)

Fig. 2.7

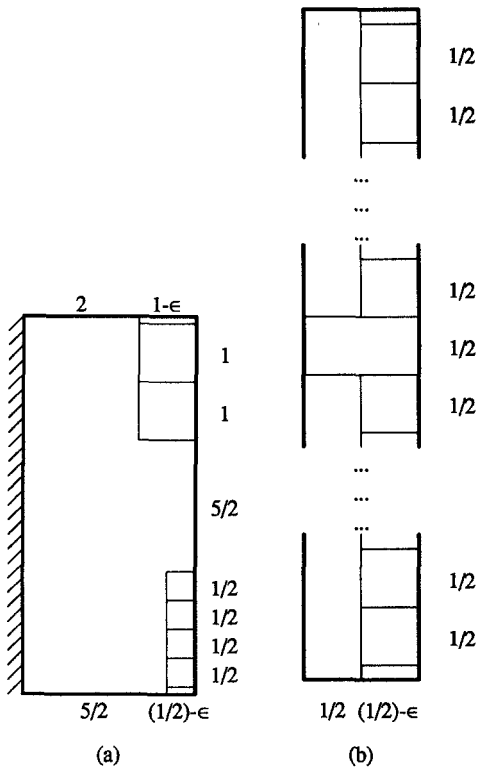


Fig. 2.8

very close to the optimal-solution value. Simple heuristics can be used to obtain a better solution most of the time. One of these heuristics consists of executing our algorithm after rotating the figure 90°, 180°, and 270°. The best of the partitions constructed is usually close to optimal. We believe that the techniques used in this paper can be modified to generate good approximate solutions to the RP-RPP problem.

Our algorithm has a large time complexity bound. The main bottleneck is the time required to solve the JRP-HF problem. To obtain a faster algorithm, we must first find a faster algorithm for the RP-HF problem. Several researchers have been working on this problem.

Gonzalez and Zheng [GZ1] show how to adapt any approximation algorithm for the RG-P problem to solve the RP-P problem. Their technique consists of using the algorithm given in [LPRS] to partition the rectilinear polygon into rectangles, then each (component) rectangle along with the points inside it becomes an RG-P problem instance. Then they use any algorithm that generates approximation solutions for the RG-P problem to solve each of these RG-P subproblems. This technique together with our algorithm TRANS can be used to generate solutions for the RP-P problem such that $L(E_{\text{apx}}(I)) \leq 4L(E_{\text{opt}}(I))$.

Our algorithm TRANS can also be generalized to solve the RP-RPP problem in which $L(B(I))+L(H(I)) \leq L(E_{opt}(I))$. Our modified algorithm generates a solution $E_{apx}(I)$ such that $L(E_{apx}(I)) \leq 4L(E_{opt}(I))$. The best previously known approximation algorithm for this restricted version of the RP-RPP problem has an approximation bound of 4.5 [L]. Interested readers may refer to [GZ2] for more details.

References

[AHU] Aho, A. V., J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[AT] Avis, D., and G. T. Toussaint, An Efficient Algorithm for Decomposing a Polygon into Star-shaped Polygons, *Pattern Recognition*, Vol. 13, 1981.

[CD] Chazelle, B., and D. Dobkin, Decomposing a Polygon into Its Convex Parts, *Proceedings of the 11th ACM Symposium on Theory of Computing*, 1979.

[GJPT] Garey, M. R., D. S. Johnson, F. P. Preparata, and R. E. Tarjan, Triangulating a Simple Polygon, *Information Processing Letters*, Vol. 7, No. 4, 1978.

[GZ1] Gonzalez, T., and S. Zheng, Bounds for Partitioning Rectilinear Polygons, *Proceedings of the 1st ACM Symposium on Computational Geometry*, June 1985, pp. 281–287.

[GZ2] Gonzalez, T., and S. Zheng, Approximation Algorithms for Partitioning Rectilinear Polygons, Technical Report 85-22, Computer Science Department, University of California, Santa Barbara, CA, 1985.

[Lev1] Levcopoulos, C., Minimum Length and Thickest-First Rectangular Partitions of Polygons, *Proceedings of the 23rd Allerton Conference on Communications, Control, and Computing*, University of Illinois, Oct. 1985.

[Lev2] Levcopoulos, C., Fast Heuristics for Minimum Length Rectangular Partitions of Polygons, *Proceedings of the 2nd Computational Geometry Conference*, June 1986.

[L] Lingas, A., Heuristics for Minimum Edge Length Rectangular Partitions of Rectilinear Figures, *Proceedings of the 6th GI Conference*, Dortmund, 1983, Lecture Notes in Computer Science, Vol. 195, Springer-Verlag, Berlin.

[LPRS] Lingas, A., R. Y. Pinter, R. L. Rivest, and A. Shamir, Minimum Edge Length Partitioning of Rectilinear Polygons, *Proceedings of the 20th Annual Allerton Conference on Communication, Control, and Computing*, University of Illinois, Oct. 1982.

[LLMP] Lodi, E., F. Luccio, C. Mugnai, and L. Pagli, On Two-Dimensional Data Organization, I, *Fundamenta Informaticae*, Vol. 2, No. 2, 1979.

[LLMPL] Lodi, E., F. Luccio, C. Mugnai, L. Pagli, and W. Lipski, Jr., On Two-Dimensional Data Organization, II, *Fundamenta Informaticae*, Vol. 2, No. 3, 1979.

[R] Rivest, R. L., The "PI" (Placement and Interconnect) System, *Proceedings of the 19th Design Automation Conference*, June 1982.

[S] Sack, J. R., An $O(n \log n)$. Algorithm for Decomposing Simple Rectilinear Polygons into Convex Quadrilaterals, *Proceedings of the 20th Annual Allerton Conference on Communication, Control and Computing*, Oct. 1982.