Partitioning Orthogonal Polygons into Fat Rectangles in Polynomial Time

Joseph O'Rourke[*] Geetika Tewari[*]

Abstract

We provide a polynomial-time algorithm to partition an orthogonal polygon of n vertices into isothetic rectangles so that the shortest rectangle side is maximized over all rectangles. Thus no rectangle is "thin"; all rectangles are "fat."

1 Introduction

In earlier work [OPT01] we introduced the problem of partitioning a simple, orthogonal polygon (one whose edges meet at right angles) into the fattest possible rectangles. The rectangle sides are parallel to the polygon edges. The goal is to maximize the minimum side length over all rectangles, i.e., to avoid "thin" rectangles. As there are usually many partitions with the same minimum, we secondarily require an *optimal partition* to employ as few rectangles as possible. The problem arises in the photolithography step of VLSI design. The VLSI mask is etched by an electron beam of some fixed minimum width. Complex shapes can only be masked without unnecessary overexposure if they can be partitioned into rectangles all of which are wider than this minimum width.

Our analysis focusses on the *cuts* needed to separate the rectangles in a partition: maximal segments of the partition whose relative interior is strictly interior to the polygon. In [OPT01] we distinguished three types of cuts:

1. *Vertex cuts*: those incident to a polygon vertex.
2. *Anchored cuts*: those incident (anchored on) a point of the polygon's boundary. (Vertex cuts are special cases of anchored cuts.)
3. *Floating cuts*: those which are strictly interior to the polygon—they "float" in the interior.

We provided examples that showed that neither vertex nor anchored cuts suffice for an optimal partition: sometimes nonvertex cuts are needed (Fig. 1), and sometimes floating cuts are needed (Fig. 2). We

*Dept. Comput. Sci., Smith College, Northampton, MA 01063, USA. {orourke,gtewari}@cs.smith.edu. Supported by NSF Distinguished Teaching Scholars award DUE-0123154.

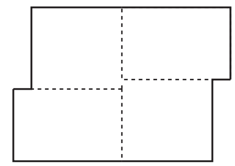

Figure 1: Not all cuts in the optimal partition are vertex cuts: the central cut is anchored but not incident to a vertex.

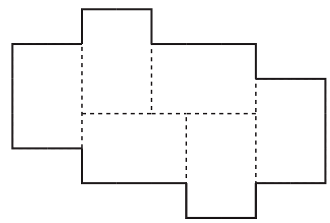

Figure 2: Not all cuts in the optimal partition are anchored: the central cut is floating.

then concentrated on vertex cuts, proving that an optimal vertex-cut partition can be found in $O(n^5)$ time. We implemented the algorithm, and discovered that, on random orthogonal polygons, it appears to run in $O(n^2)$ time. A sample of the code's output is shown in Fig. 3. Further, we claimed without proof that anchored cuts fall on a certain "midline" grid, which led to a polynomial-time algorithm for anchored cuts as well. We left unresolved whether there is a polynomial-time algorithm for unrestricted cuts, i.e., including floating cuts.

Here we continue the investigation, resolving most of the open problems, and in particular this last question. Time complexities for the three classes of cuts are as follows:

Cut type	Time complexity
vertex cuts	$O(n^5)$
anchored cuts	$O(n^{10})$
unrestricted cuts	$O(n^{42})$

All three algorithms follow a similar dynamic-programming structure, although we could only handle floating cuts with an intricate algorithm that leads to the large upper bound indicated.

Perhaps more interesting than these impractical algorithms are the results we could prove on the geomet-

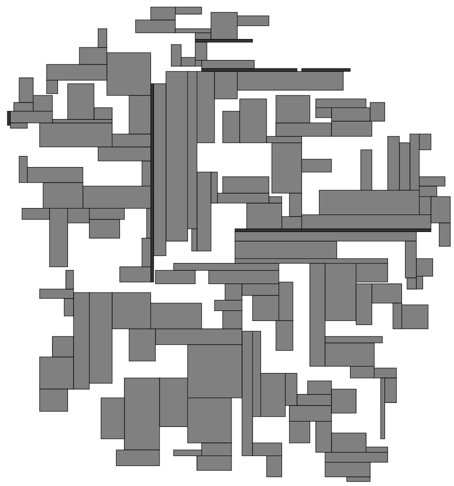

Figure 3: A polygon of $n = 348$ vertices optimally partitioned by vertex cuts into 155 rectangles. The tied-for-thinnest rectangles are shaded dark.

ric and combinatorial structure of optimal partitions. In particular, we discovered that our claim in [OPT01] that anchored cuts fall on a midline grid is only partially correct: they could also fall on thirds-lines (but not fourths-, etc.). Our most complex structural result is that an optimal partition never includes floating cuts that are too "deep" in the polygon interior (Thm. 6): cuts floating deep in a "sea" of floating cuts are never necessary. This leads to an $O(n)$ bound on the number of rectangles in an optimal partition (Thm. 7).

2 Anchored Cuts

The example in Fig. 1 suggests that anchored cuts can be chosen to lie on lines midway between two edges, for it would seem to be advantageous to slide such a cut to balance the dimensions of the rectangles supported to either side. We were therefore surprised to discover the example shown in Fig. 4, which has the property that the optimal partition needs to use two anchored cuts which are not midway between any pair of edges, but rather lie at one-third and two-thirds between two edges. Call a partition that uses only anchored cuts an *anchored partition*. Let an optimal partition have minimum rectangle side length δ, and call a rectangle a *δ-rectangle* if at least one of its dimensions is δ. Say that a cut c *supports* a rectangle R if one of R's sides shares a positive-length portion of c. The following lemma shows that the situation in Fig. 4 is in a sense the worst that can happen with anchored cuts:

Lemma 1 *No optimal anchored partition includes three parallel anchored cuts supporting δ-rectangles between them.*

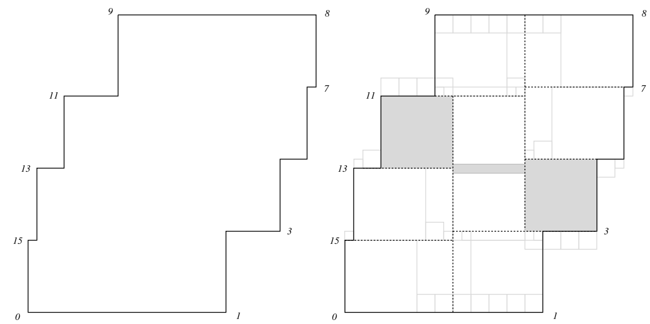

Figure 4: An $n = 16$ orthogonal polygon and its unique optimal partition. The two shaded rectangles are $\delta \times \delta$, and the tall central rectangle is $\delta = 8$ wide (indicated by the shaded bar). Light lines indicate grid dimensions.

With only two parallel anchored cuts, a $\delta + \delta + \delta$ configuration as illustrated is the most complex possible. This lemma yields $O(n^2)$ possible anchor points on the boundary: at fractions $\{\frac{1}{3}, \frac{1}{2}, \frac{2}{3}\}$ between each pair of edges. Treating each anchor point as a "pseudovertex," and applying the $O(n^5)$ vertex-cut algorithm, yields:

Theorem 2 *An optimal anchored partition for a polygon of n vertices can be found in $O(n^{10})$ time.*

3 Unrestricted Cuts

Lem. 1 led us to wonder if a similar result might not hold for floating cuts. Fig. 5 shows that no easy generalization is possible. Here the positioning of four cuts—

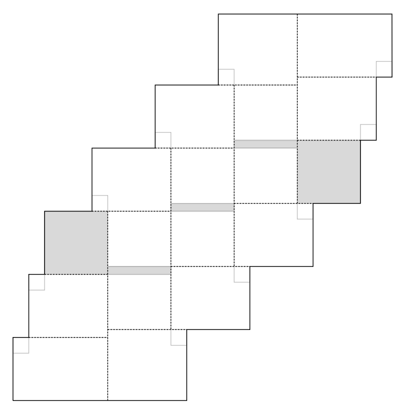

Figure 5: The unique optimal partition employs floating cuts at multiples of one-fifth between two polygon edges. Shading indicates δ dimensions.

two anchored and two floating–is determined by a chain of five δ-rectangles. Generalizing the example shows that $1/k$-th lines might be necessary for any $k = O(n)$.

Although in such an example, the position of some central cuts is determined by distant edges, none of these cuts is far from the boundary in another sense. To make this sense clear, we introduce the notion of the *depth* of a cut. First, say that a segment a is *incident* to an object b if an endpoint of a is on b. Note that this definition is not symmetric. For the symmetric notion, we use the term *touch*: two segments touch if they share a point. Cut depth is defined inductively:

0 Vertex cuts have depth 0. Polygon edges are also defined to have depth 0, being viewed as cuts incident to their vertex endpoints.

1 Otherwise, a cut has depth 1 plus the minimum depth over all segments that touches it.

Thus, every point on a cut of depth k can reach a vertex by a "cut path" with k turns. Nonvertex anchored cuts have depth 1, because they touch a polygon edge. Floating cuts have depth ≥ 1. Notice that cut depth does not distinguish between anchored and floating, because a floating cut could have both *endcaps* (the cuts incident to its two endpoints) vertex cuts, i.e., depth-0 cuts. We call all cuts of depth ≥ 1 *movable cuts*, for they, unlike vertex cuts, might slide perpendicularly without changing the structure of the partition.

The polynomial-time algorithm for unrestricted cuts relies on a series of structural lemmas.

3.1 Cut-depth Bound

Lemma 3 *In an optimal partition, every rectangle that is bounded by two or more movable cuts, is bounded by the cuts in a "pin-wheel" pattern (in either orientation), as illustrated in Fig. 6.*

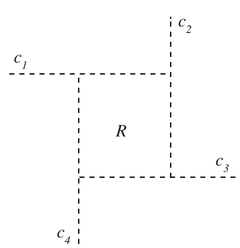

Figure 6: A pinwheel of cuts surrounding rectangle R.

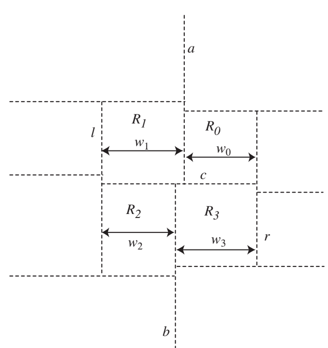

Figure 7: A movable cut with all incident cuts movable.

Lemma 4 *In any optimal partition, every movable cut c to which is incident only movable cuts, has the structure indicated in Fig. 7: there are just two incident cuts a and b, staggered on c by less than δ, and the endcaps l and r terminate on cuts as illustrated.*

We will rely on figures to hint at our most complex structural lemma:

Lemma 5 *An optimal partition never includes the structure shown in Fig. 8(a), for it can always be improved to the structure shown in (c).*

This lemma obviates the need for depth-(≥ 3) cuts:

Theorem 6 *An optimal partition never includes a floating cut of depth greater than 2.*

That this result is best possible is established by the example in Fig. 9, which (nonobviously) requires a cut c of depth 2.

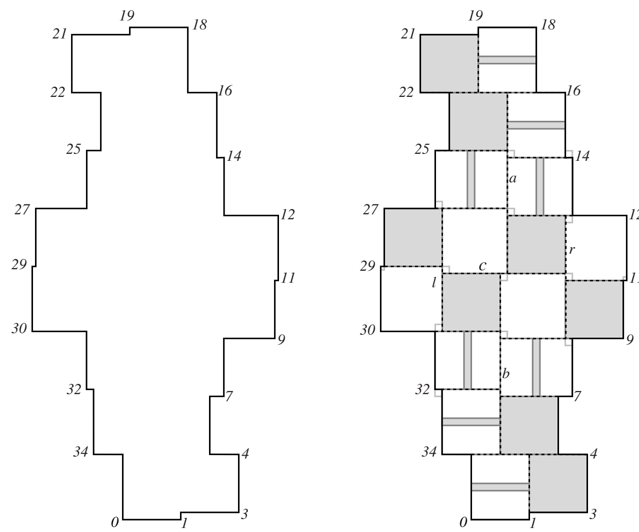

Figure 9: A polygon that requires a floating cut c of depth 2. Shading indicates δ dimensions.

3.2 Combinatorial Lemmas

We need to bound the number of rectangles employed in any optimal partition. Secondly, we need to bound the number of possible values of δ, and from that define a finite grid on which the partition may be drawn.

The first bound relies crucially on the depth bound of Thm. 6:

Theorem 7 *An optimal partition consists of $O(n)$ rectangles.*

The second bound is obtained from the "δ-graph." Let $\mathcal{P}$ be an optimal partition of polygon P with minimum width δ. Define G_δ, the *δ-graph*, to be a graph whose nodes are the segments of the partition (both cuts and polygon edges), with two nodes connected by an arc iff the corresponding segments mutually support a rectangle of $\mathcal{P}$ of width δ on opposite sides. A *monotonic path* in G_δ is a path whose node segments are sorted horizontally or vertically.

Lemma 8 *The δ-graph G_δ of an optimal partition $\mathcal{P}$ with minimum width δ contains a monotonic path both of whose end nodes are either polygon edges or vertex cuts (i.e., both are cuts of depth 0).*

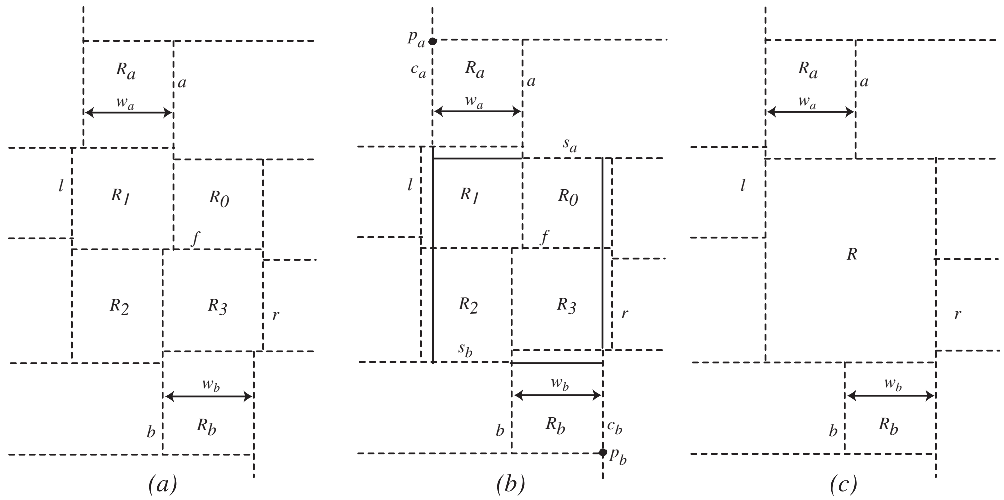

Figure 8: The structure in (a) can be repartitioned as indicated in (b) to the improved structure shown in (c).

Lem. 8 is manifest in Figs. 4, 5, and 9.

Theorem 9 *Let an optimal partition of an n-vertex polygon have minimum side length δ. Then $\delta \in \Delta$, where Δ is a set of rational numbers of cardinality $|\Delta| = O(n^3)$, and which can be computed in time $O(n^3)$.*

Corollary 10 *There is an optimal partition whose cuts fall on a subset of $O(n^4)$ gridlines.*

3.3 Dynamic Programming Chain

On the basis of the constraints established in previous sections, it is now possible to describe a polynomial-time and space algorithm to solve the problem using these types of cuts.

Let C be a chain of $k \leq K$ cuts connecting vertices v_i to v_j, separating P into two pieces: P_1, the unprocessed piece, and P_2, the already processed piece. (The constant upper bound K will be specified later.) Our goal is to advance C to C' inside P_1. If we can reduce the computation of the optimal partition of P_1 to a polynomial number of combinations of optimal partitions of smaller pieces of P_1, each bound by $\leq k$ cuts, then we will achieve a polynomial-time and space algorithm for computing the optimal partition via dynamic programming.

If we can find a short-enough path from a central-enough interior cut of C to a vertex of P_1, then we are finished. For then this path would partition P_1 into two pieces, each bounded by a chain of $\leq k$ cuts connecting vertices. Call any such path a *good boundary path*. If we can find a path from C, inside P_1 and back to C, such that C' is $\leq k$ cuts, then we are finished, for the constant-size portion sandwiched between C and C' can be optimally partitioned in polynomial time, and the remaining subset of P_1 bound by C' is handled by dynamic programming. Call such a path a *good shortcut*.

In both instances, we need to find a polynomial-number of good paths that cover all eventualities.

We rely on this technical lemma, which employs the depth bound of Thm. 6 crucially:

Lemma 11 *There always exists a good boundary path or a good shortcut of length at most 4 from a "central cut" of any chain C with $|C| \leq K = 10$.*

This leads to our main theorem:

Theorem 12 *An optimal partition of a polygon of n vertices may be found in $O(n^{42})$ time.*
Proof: We know from Cor. 10 that the cuts lie on a grid of $O(n^4)$ lines. Trying all possible paths of length 4 then costs no more than $O(n^{16})$: n^4 locations for the first cut, n^4 lines on which to stop the first cut and start the second, and so on. The size of the dynamic programming table would be bound by $O(n^2)$ for the choice of v_i and v_j, and then all the possible K-chains, which is (alas!) $O(n^{40})$. We then have a polynomial bound, $O(n^{42})$, on the space for the table. Constructing the table dominates all other computations, so the time complexity is $O(n^{42})$ as well. □

Acknowledgment. We thank Irena Pashchenko, our coauthor on [OPT01], for her initial work on this project.

References

[OPT01] J. O'Rourke, I. Pashchenko, and G. Tewari. Partitioning orthogonal polygons into fat rectangles. In *Proc. 13th Canad. Conf. Comput. Geom.*, pages 133–136, August 2001.