Partitioning Orthogonal Polygons into Fat Rectangles

Joseph O'Rourke* Irena Pashchenko* Geetika Tewari*

Abstract

We provide an algorithm to partition an orthogonal polygon of n vertices into isothetic rectangles so that the smallest width over all rectangles is maximized. Thus no rectangle is "thin." We explore several restrictions on the cuts used in the partition, focusing our attention on cuts which include a polygon vertex. For such *vertex cuts*, we detail an $O(n^5)$-time, $O(n^4)$-space dynamic programming algorithm that finds an optimal partition. For the increasingly special classes of orthogonal polygons known as monotone mountains, pyramids, and staircases, we have $O(n^3)$, $O(n^2)$, and $O(n)$-time algorithms respectively. We have also proven that our general algorithm extends to the more general *anchored cuts*, leaving unrestricted cuts an open problem. Finally, we report on a preliminary implementation of the general vertex cut algorithm.

1 Introduction

VSLI masks are etched by electron beams 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. Thus it is of some interest to develop an algorithm that can find an optimal partition of a polygon into rectangles, in the sense of maximizing the minimum width of any rectangle in the partition.[1] This paper presents a polynomial-time algorithm for one version of this problem. We now proceed to define the problem more precisely.

We only consider *orthogonal polygons*, simple polygons whose edges meet at right angles; we will use the abbreviation *o-polygon* henceforth. We also only consider *isothetic rectangles*, those whose sides are parallel to the polygon edges. A *partition* of an o-polygon P into rectangles is a set of isothetic rectangles whose interiors are pairwise disjoint, and whose union covers P. The *thickness* of a partition is the minimum width of any rectangle in the partition. We seek a partition that maximizes the thickness over all partitions; we call such a partition *optimal*. Loosely, it employs only "fat" rectangles. Despite considerable earlier study of rectangle partitions (and covers) under a variety of optimality criteria, we could find no literature on our precise version.

Call maximal segments of the partition whose relative interior is strictly interior to the polygon *cuts*; one would need to cut these segments to separate the rectangles. It is natural to wonder if the cuts may be restricted in any way without altering the optimum value of partitions. For example: Is there always an optimal partition such that at least one cut connects two boundary points, i.e., cuts all the way through? Fig. 1 shows that the answer is NO.

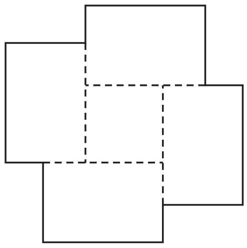

Figure 1: No cut in this optimal partition connects boundary-to-boundary.

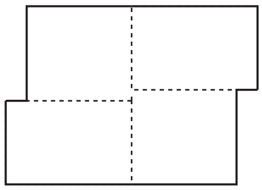

Figure 2: Not all cuts in the optimal partition are vertex cuts.

Is there always an optimal partition with every cut a *vertex cut*—including at least one vertex? Fig. 1 uses only vertex cuts, but Fig. 2 employs (necessarily) one cut that includes no vertex. Is there always an optimal partition with every cut lying on the *vertex grid* formed by all horizontal and vertical lines through all polygon vertices? Fig. 2 again provides a counterexample. Could we say that no reflex vertex ever need be incident to two cuts? Fig. 3 shows this is too strong a claim. Is there always an optimal partition such that every cut is *anchored* in the boundary—including at least one point of the polygon boundary? Figs. 1-3 use only anchored cuts, but Fig. 4 employs (necessarily) one *floating* cut that does not touch the boundary.

Other optimality criteria permit some such restrictions. For example, the minimum number of rectangles can be achieved via vertex cuts [LTL89]; and the minimum total cut length can be achieved by anchored cuts lying on the vertex grid [LPRS82]. We have proven that for our optimality measure, all cuts may be restricted to lie on a certain *midline grid*, a refinement

*Dept. Comput. Sci., Smith College, Northampton, MA 01063, USA. {orourke,ipashche,gtewari}@cs.smith.edu. Supported by NSF grant CCR-9731804.

[1] We thank Ulrich Finkler for suggesting this problem.

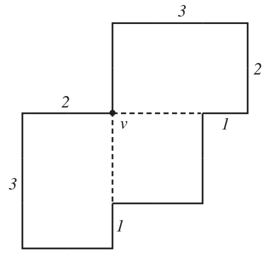

Figure 3: Two cuts incident to a reflex vertex in the optimal partition. Edges are labeled with their lengths.

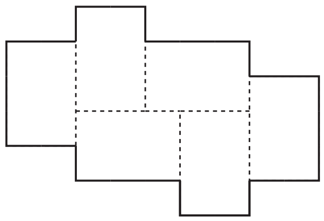

Figure 4: Not all cuts in the optimal partition are anchored: the central cut is "floating."

of the vertex grid, but we were unable to find a more tractable cut restriction. We have only been able to find polynomial-time algorithms for partitions restricted to anchored cuts. We will concentrate for the majority of the paper on vertex cuts, for the extension to anchored cuts is not difficult.

2 Special Polygon Classes

Our abstract mentions several special classes of polygons for which we have faster algorithms. Here we only mention one, *staircase polygons*, those with a single alternating series of reflex/convex vertices. Let the reflex vertices be indexed $1, \ldots, n-1$ as in Fig. 5, with x_i the horizontal and y_i the vertical edge length incident to vertex i.

Theorem 2.1 *The optimal width δ for a rectangle partition of a staircase polygon is*

$$\delta = \min\{ x_0, y_n, \min_i\{\max\{x_i, y_i\}, i = 1, \ldots, n-1\}\} \quad (1)$$

Proof: *(Sketch).* Each reflex vertex must be incident to a cut. Eq. (1) claims that the cut may be chosen to avoid creating a rectangle with the shorter edge, leaving the larger of the two, $\max\{x_i, y_i\}$. The key to the proof is establishing that the order in which the cuts are chosen does not affect the minimum width: a locally greedy algorithm leads to the global optimum. □

Theorem 2.1 leads to an $O(n)$-time algorithm. For orthogonal *pyramids*, two staircases glued back-to-back,

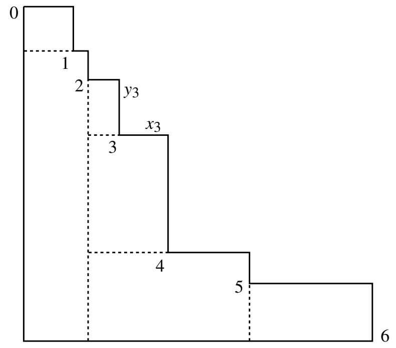

Figure 5: Staircase polygon; $n = 6$. Here $\delta = x_2$.

we can achieve $O(n^2)$ time for vertex cuts. And for *monotone mountains*, polygons with one horizontal edge whose length is the sum of all other horizontal edge lengths,[2] we have an $O(n^3)$ algorithm, again for vertex cuts. But for the natural next step up in shape complexity, *orthogonally convex* polygons, which can be viewed as formed from four staircases angling in the four diagonal directions (e.g., Figs. 1, 4), we were unable to find a simple algorithm. We turn then to the general algorithm.

3 Vertex Cuts for General Orthogonal Polygons

Although it is natural to hope that any partition problem would have a polynomial-time solution—because cuts insulate recursive subproblems from one another—naive attempts in this instance quickly lead to exponential behavior. A central insight from [LPRS82] may be imported to our situation, despite the differences between minimizing total cut length and maximizing thickness: that it may be possible to organize the recursion so that all subproblems fall into a polynomially-sized class. This then supports a dynamic programming approach.

The key property maintained as an induction hypothesis is that every subpolygon in the recursion is circumscribed by a contiguous portion of the original polygon's boundary, and at most two connecting cuts; call this the *2-cut IH*. It is crucial to avoid an arbitrary number of boundary cuts, for that way lies exponential explosion.

Given a subpolygon P' of the original P satisfying the 2-cut IH, the challenge is to advance in the partition while maintaining the IH. This is accomplished by finding one rectangle R touching the cuts of P' included in an optimal partition, such that the components of $P' \setminus R$ satisfy the IH. We achieve this goal via a series of four lemmas.

[2] Sometimes called *Manhattan skyline polygons*.

Lemma 3.1 *If two vertex cuts cross in an optimal partition, then there is another optimal partition in which one of these cuts terminates on the other.*

Following [LPRS82], let the corner c at the join of the two cuts of P' be called the *candidate* point. We seek to find suitable *match* points $m \in P'$ such that the rectangle R determined by c and m has the above mentioned properties. The reasoning differs depending on whether c is convex or reflex in P'.

Lemma 3.2 *If c is convex, then all the match points m that must be explored to guarantee an optimal partition lead to satisfaction of the 2-cut IH.*

Two cases are illustrated in Fig. 6. A considerable portion of our effort was directed toward explicating exactly which match points "must be explored," here and in the next two lemmas.

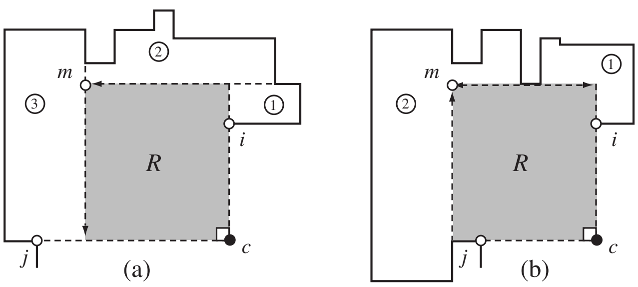

Figure 6: Candidate c is convex, and removal of $R = (c, m)$ leaves several subpolygons (three in (a) and two in (b)) satisfying the 2-cut IH.

Lemma 3.3 *If c is reflex, then the conclusions of the previous lemma hold.*

Lemma 3.4 *If P' has only one cut on its boundary, then again all the R's that need be explored result in 2-cut IH satisfaction for the components of $P' \setminus R$.*

Fig. 7 illustrates two cases.

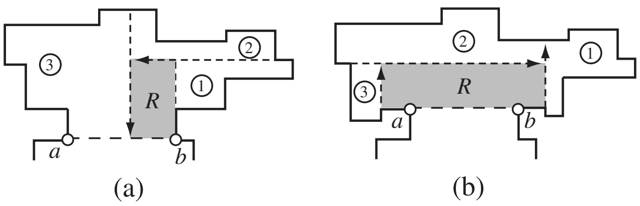

Figure 7: 1-cut cases. (a) Equivalent to a 2-cut convex candidate; (b) The "difficult" case.

These four lemmas are not straightforward. Lemmas 3.2 and 3.3 have their counterparts in [LPRS82], but their proofs were only sketched.[3] Lemma 3.1 is not needed in [LPRS82]; and Lemma 3.4, although our most difficult, is a "special case" not discussed by them.

These lemmas lead, with some analysis, to:

Theorem 3.5 *The optimal vertex-cut partition of an n-vertex o-polygon into rectangles can be found in $O(n^5)$ time and $O(n^4)$ space.*

Despite the high time-complexity of the algorithm,[4] and the intricacy of its induction step, the dependence upon shape leads to the possibility that it will run fast on "most" polygons. Indeed, we are not certain the claimed bounds can be achieved in the worst case. To explore empirical questions, we have completed a partial implementation, to which we now turn.

4 Implementation

Although the algorithm is conceptually simple, we have found it a challenge to implement. The overall structure for computing the optimal partition of a 2-cut subpolygon with candidate corner c is as follows:

Optimal Rectangle Partition from c
Compute all match points for the given candidate c.
for each potential match point m do
  if m is viable for c then
    for each way to cut out $R = (c, m)$ do
      Create subpolygons $P_1, P_2, \ldots$.
      Recursively find optimal partition of each P_i.
      w = min over all optimal values returned.
    Save best (max) w over all cuts.
return best (max) width over all match points.

The myriad ways to excise R (hinted at in Figs. 6-7) lead to cases that seem difficult to unify. Consequently, we have not implemented the most complex case, Fig. 7b, which requires a second loop to find all R that cover the cut ab. This case only arises when two edges are collinear and connectable by a cut.

Although the above pseudocode is written as if it were recursive, the recursion's first step is to check the dynamic programming table to see if the subpolygon has already been partitioned. Indexing into the table relies crucially on the 2-cut IH. Our current implementation stands at 3000 lines of C++.

We have developed a model for generating random orthogonal polygons, and tested the code on random polygons up to $n = 558$ vertices. A typical example is shown in Fig. 8. Because we break width-ties in favor of partitions with fewer rectangles, the partition shown is both thickest and most parsimonious.

[3]And are in any case not relevant because of reliance on the optimality criterion. This seminal conference paper was never followed by a journal version.

[4][LPRS82] achieves $O(n^4)$ time, but our extra factor of n derives from a degenerate case below the level of their scrutiny.

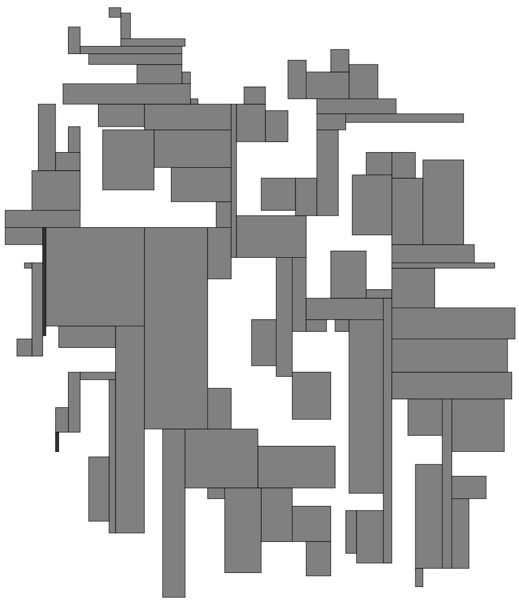

Figure 8: A polygon of $n = 212$ vertices optimally partitioned by vertex cuts into 92 rectangles. The two tied-for-thinnest rectangles are shaded dark.

Our current implementation is highly inefficient. For example, our dynamic programming table lookups are via a linear scan, adding an extra factor of n. Fig. 9 shows that the growth rate is apparently cubic for our code applied to these random polygons, suggesting quadratic behavior or better for a more efficient implementation. Interestingly, the space growth is nearly

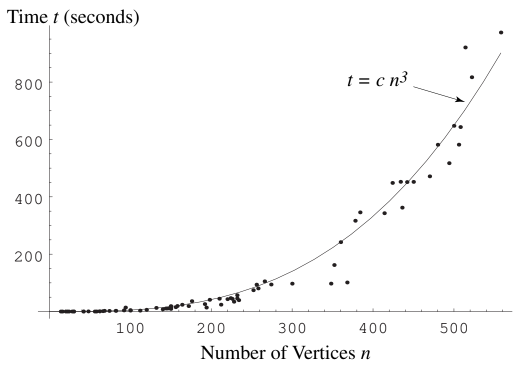

Figure 9: Time growth for random polygons. Measurements on a Pentium III 931 MHz processor.

linear—about $3n$ as Fig. 10 shows—in contrast to the $O(n^4)$ bound guaranteed by Theorem 3.5. This leads to some hope that a better upper bound could be derived.

5 Discussion

Aside from improving our bounds, there are at least three interesting open problems:

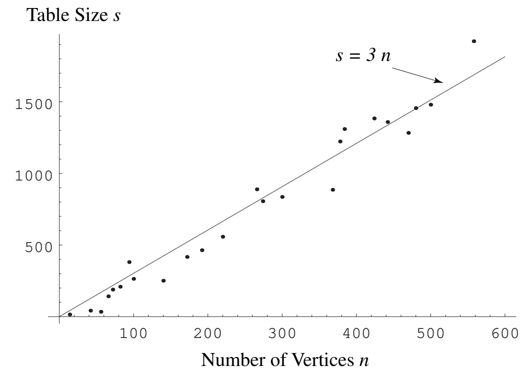

Figure 10: Dynamic programming table length.

1. Is the problem NP-complete for polygons with holes? We conjecture that it is.
2. Although not relevant for our motivating application, guillotine cuts [MS98, Mit99] (a.k.a. glass cuts [PT00]) are often of great practical importance. Is there a polynomial-time algorithm for guillotine cuts?
3. Finally, and most pressingly, is there a polynomial-time algorithm for unrestricted cuts, i.e., including floating cuts (Fig. 4).

Acknowledgment. Initial investigations (Sec. 1: Figs. 1, 2, 4), were carried out by the second author in collaboration with Ulrich Finkler at IBM Watson.

References

[LPRS82] A. Lingas, R. Pinter, R. Rivest, and A. Shamir. Minimum edge length partitioning of rectilinear polygons. In *Proc. 20th Allerton Conf. Commun. Control Comput.*, pages 53–63, 1982.

[LTL89] W. T. Liou, J. J. M. Tan, and R. C. T. Lee. Minimum partitioning simple rectilinear polygons in $O(n \log \log n)$ time. In *Proc. 5th Annu. ACM Sympos. Comput. Geom.*, pages 344–353, 1989.

[Mit99] J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM J. Comput.*, 28:1298–1309, 1999.

[MS98] H. Martini and V. Soltan. Minimum convex partition of polygonal domains by guillotine cuts. *Discrete Comput. Geom.*, 19:291–304, 1998.

[PT00] J. Pach and G. Tardos. Cutting glass. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 360–369, 2000.