Orthogonal Strip Partitioning of Polygons: Lattice-Theoretic Algorithms and Lower Bounds

Jaehoon Chung
Korea Institute for Advanced Study (KIAS), Seoul, Republic of Korea

Abstract

We study a variant of a polygon partition problem, introduced by Chung, Iwama, Liao, and Ahn [ISAAC'25]. Given orthogonal unit vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^2$ and a polygon P with n vertices, we partition P into connected pieces by cuts parallel to $\mathbf{v}$ such that each resulting subpolygon has width at most one in direction $\mathbf{u}$. We consider the value version, which asks for the minimum number of strips, and the reporting version, which outputs a compact encoding of the cuts in an optimal strip partition.

We give efficient algorithms and lower bounds for both versions on three classes of polygons of increasing generality: convex, simple, and self-overlapping. For convex polygons, we solve the value version in $O(\log n)$ time and the reporting version in $O\big(h \log\big(1 + \frac{n}{h}\big)\big)$ time, where h is the width of P in direction $\mathbf{u}$. We prove matching lower bounds in the decision-tree model, showing that the reporting algorithm is input-sensitive optimal with respect to h. For simple polygons, we present $O(n \log n)$-time, $O(n)$-space algorithms for both versions and prove an $\Omega(n)$ lower bound. For self-overlapping polygons, we extend the approach for simple polygons to obtain $O(n \log n)$-time, $O(n)$-space algorithms for both versions, and we prove a matching $\Omega(n \log n)$ lower bound in the algebraic computation-tree model via a reduction from the δ-closeness problem.

Our approach relies on a lattice-theoretic formulation of the problem. We represent strip partitions as antichains of intervals in the Clarke–Cormack–Burkowski lattice, originally developed for minimal-interval semantics in information retrieval. Within this lattice framework, we design a dynamic programming algorithm that uses the lattice operations of meet and join. To the best of our knowledge, this is the first geometric application of the Clarke–Cormack–Burkowski lattice.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Polygon partitioning, Strip partition, Lattice, Self-overlapping curves

Digital Object Identifier 10.4230/LIPIcs.SWAT.2026.14

Related Version *Full Version*: `https://arxiv.org/abs/2604.15247`

Funding This work was supported in part by a KIAS Individual Grant AP106101 via the Center for Artificial Intelligence and Natural Sciences at Korea Institute for Advanced Study, and by the Center for Advanced Computation at Korea Institute for Advanced Study.

1 Introduction

Partitioning a polygon into the minimum number of pieces has been studied extensively under shape-class constraints, for example into convex, star-shaped, quadrilateral, or monotone pieces [1, 10, 15]. Much less is known for metric constraints that bound geometric measures, such as width, diameter, or perimeter. In contrast to the classical shape-based setting, the problem becomes significantly harder under such metric bounds. Imposing a unit upper bound on diameter or perimeter already makes the minimum partition problem NP-hard for simple polygons without holes [3]. Moreover, no polynomial-time algorithm is known for computing an optimal solution even when the input is a square or an equilateral triangle [2].

In ISAAC'25, Chung et al. [11] introduced a minimum partition problem with an upper-bounded width constraint, which limits the width of each partitioned piece in prescribed directions. Such a constraint is motivated by manufacturing and recycling, where materials must be cut or processed within width limits. For instance, Lan et al. [16] recycle wind

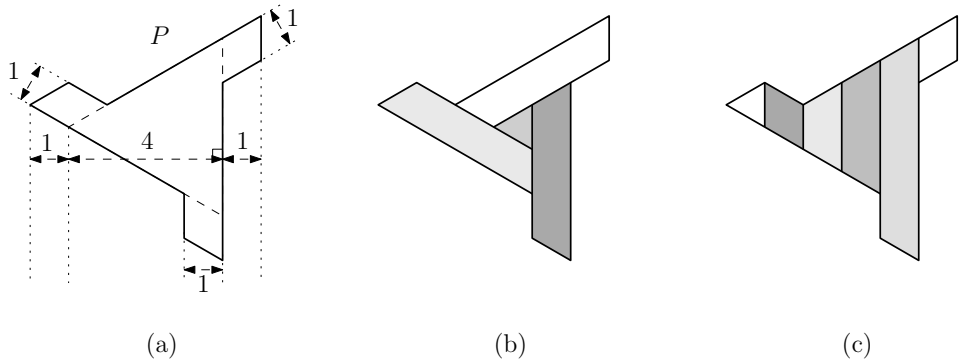


Figure 1 (a) The polygon P has a windmill shape with three arms extending from an equilateral triangle of height 4. (b) A minimum partition of P under width and cut constraints, $W = U = \mathbb{S}^+$. (c) A minimum orthogonal strip partition of P with $\mathbf{u} = (1, 0), \mathbf{v} = (0, 1)$.

turbine blades by first cutting the blades into panels small enough to fit the intake of crushing machines, and then crushing them into recyclates. Moreover, wind turbine blades are typically made of glass-fiber composites with directionally aligned fibers, so the cutting effort can depend on the cut direction [13].

From a theoretical perspective, the problem is related to Bang's conjecture, a long-standing open problem in convex geometry [5]. Chung et al. resolved a partition analogue of this conjecture, showing that for any convex body, there exists a direction in which an optimal partition can be achieved using only parallel cuts.

These applications and theoretical results motivate the study of minimum partitioning under an upper-bounded width constraint. Moreover, they suggest that it is natural to restrict attention to partitions in which all cuts are parallel to a single direction, which we call the *strip partition model*.

We consider three classes of input polygons: convex, simple, and self-overlapping, where a self-overlapping polygon generalizes a simple polygon by allowing the boundary to self-intersect. A *self-overlapping curve* was introduced by Shor and van Wyk [21] as the boundary of a topological disk immersed in $\mathbb{R}^2$. When this curve is polygonal, it serves as the boundary of a self-overlapping polygon. Unlike in the simple case, the boundary alone does not determine a unique interior or visibility relation. Following Shor and van Wyk, we define visibility with respect to a triangulation of the curve, so that partitions of the polygon are well defined.

In this work, we study orthogonal strip partitioning of polygons and present efficient algorithms with lower bounds for each class of input polygons.

Orthogonal strip partition problem. We first recall the partitioning problem under width and cut constraints introduced by Chung et al. [11]. Let P be a simple polygon, and let $\mathbb{S}^+ = \{(\cos\theta, \sin\theta) \mid 0 \le \theta < \pi\}$ denote the set of all unit vectors in the plane. For $\mathbf{v} \in \mathbb{S}^+$ and a set $X \subset \mathbb{R}^2$, let $\omega_{\mathbf{v}}(X)$ denote the width of X in direction $\mathbf{v}$, that is, the length of the projection of X onto a line parallel to $\mathbf{v}$. Given a subset $W \subseteq \mathbb{S}^+$, we say that X satisfies the *unit-width constraint* W if $\omega_{\mathbf{v}}(X) \le 1$ for some $\mathbf{v} \in W$. We also impose a *cut constraint* $U \subseteq \mathbb{S}^+$, meaning that every cut used to partition P must be aligned with some direction in U. Here, a cut is a line segment $c \subseteq P$ whose relative interior lies in the interior of P. Given a simple polygon P and sets $W, U \subseteq \mathbb{S}^+$, the goal is to find a partition of P into the minimum number of pieces such that every cut is aligned with a direction in U and each piece satisfies the unit-width constraint W.

The *orthogonal strip partition problem* is the special case with $W = \{\mathbf{u}\}$ and $U = \{\mathbf{v}\}$ for two orthogonal unit vectors $\mathbf{u}, \mathbf{v} \in \mathbb{S}^+$. A *strip* is a subpolygon $Q \subseteq P$ with $\omega_{\mathbf{u}}(Q) \leq 1$, and a strip partition divides P into strips using only cuts parallel to $\mathbf{v}$. Without loss of generality, we assume $\mathbf{u} = (1, 0)$ and $\mathbf{v} = (0, 1)$. Under this assumption, each cut is a vertical segment and every strip Q has width at most 1 in the horizontal direction. In this orthogonal setting, we regard a cut as a maximal vertical segment contained in P whose relative interior lies in the interior of P. See Figure 1 for an illustration.

Cut descriptors and lossless encoding. We define the *descriptor* of a cut as the pair of boundary edges of P on which its top and bottom endpoints lie, together with its x-coordinate. If a top or bottom endpoint lies on two incident edges, we choose the left edge, so the descriptor uniquely determines the position of the cut. A partition of P can be represented by the descriptors of its cuts. Our primary objective is to minimize the number of strips. In the reporting variant, we also compute the descriptors of all cuts in an optimal partition.

To store these descriptors succinctly, we fix a *lossless encoding* scheme for strip partitions. For each polygon P, the scheme assigns a codeword to each strip partition of P so that the partition can be reconstructed from the codeword without loss. Thus each strip partition of P is represented by a unique codeword.

The purpose of lossless encoding is to separate the combinatorial difficulty of finding an optimal partition from the cost of listing all cuts. For an integer $k \geq 1$, the optimal strip partition of $k \times 1$ rectangle consists of $k - 1$ cuts spaced one unit apart. If we store one descriptor per cut, any reporting algorithm requires $\Omega(k)$ time to output the explicit list. We therefore allow the algorithm to output a succinct codeword from which the full list of cut descriptors can be reconstructed without loss, for example by encoding the x-coordinate of the leftmost cut together with the number of cuts. This is called a *run-length* encoding [6]. Under this model, the cost is no longer dominated by explicit output size, but by distinguishing among combinatorially different optimal partitions.

▶ **Definition 1** (Orthogonal Strip Partition Problems). *An orthogonal strip partition of a polygon $P \subset \mathbb{R}^2$ is a partition of P into subpolygons (strips) using vertical cuts, such that each subpolygon has horizontal width at most 1. We consider three input domains $\mathcal{D} \in \{$convex, simple, self-overlapping$\}$. For $\mathcal{D} \in \{$convex, simple$\}$, an input instance consists of the vertices of P listed in counterclockwise order along its boundary. For $\mathcal{D} =$ self-overlapping, an instance consists of a triangulation of P. For each domain $\mathcal{D}$, we consider two versions:*

- Value version: *return the minimum number of strips.*
- Reporting version: *with respect to the fixed lossless encoding scheme, output any codeword representing an optimal strip partition of P.*

Let $\mathcal{D} \in \{\text{conv}, \text{simp}, \text{self}\}$ denote the type of P, corresponding to convex, simple, and self-overlapping, respectively. For a polygon P in domain $\mathcal{D}$, let $\text{OSP}_V^{\mathcal{D}}(P)$ and $\text{OSP}_R^{\mathcal{D}}(P)$ be the value and reporting versions of the orthogonal strip partition problem, respectively. We denote by $\mathsf{opt}(P)$ the minimum number of strips in an optimal strip partition of P.

Related work. The general version of the strip partition problem with arbitrary width and cut constraints $W, U \subseteq \mathbb{S}^+$, was first introduced by Chung et al. [11]. They studied structural properties in simple polygons, in particular the monotonicity of the minimum partition number under polygon containment. They proved a partition analogue of Bang's conjecture. If U contains every direction orthogonal to each $\mathbf{w} \in W$, then any convex body admits an optimal partition obtained by equally spaced cuts parallel to some direction $\mathbf{v} \in U$. Hence an optimal solution exists in the strip partition model.

Most algorithmic work for the strip partition model focuses on monotone variants, including the convex case. Liu [19] gave an $O(n \log n)$-time algorithm for partitioning a rectilinear polygon with rectilinear holes into x-monotone pieces using horizontal cuts, where n is the total number of vertices. Lee et al. [17] improved this to linear time for hole-free polygons. Lingas and Soltan [18] studied minimum convex partitions of polygons with holes by cuts restricted to a set U of directions; in particular, they give an $O(n \log n)$-time algorithm when $|U| = 1$ (parallel cuts).

Our results. We study orthogonal strip partitioning of convex, simple, and self-overlapping polygons, and give efficient algorithms with matching lower bounds in appropriate computational models, in particular the decision-tree and algebraic computation-tree models.

First, for a convex n-gon, the value version can be solved in $O(\log n)$ time and the reporting version in $O\big(h \log(1 + n/h)\big)$ time, where $h \geq 1$ is the horizontal width of the input polygon. As a function of h, this bound is $O(\log n)$ when $h = 1$ and approaches $O(n)$ as h increases. We prove matching lower bounds in the decision-tree model, showing that any algorithm needs $\Omega(\log n)$ time for the value version and $\Omega\big(h \log(1 + n/h)\big)$ time for the reporting version. Thus the reporting algorithm is input-sensitive optimal with respect to h.

Second, for simple polygons with n vertices, we use a lattice-theoretic formulation of strip partitions. We represent strip partitions as antichains of intervals in the Clarke–Cormack–Burkowski lattice, originally developed for minimal-interval semantics in information retrieval, and design a dynamic programming algorithm on a trapezoidal decomposition whose recurrence is expressed using the lattice operations of meet and join on these antichains. This yields an $O(n \log n)$-time, $O(n)$-space algorithm that solves both the value and reporting versions. We also show that, unlike for convex polygons, any algorithm requires $\Omega(n)$ accesses to the coordinates of the input vertices in the worst case. To the best of our knowledge, this is the first geometric application of the Clarke–Cormack–Burkowski lattice.

Finally, we extend this lattice-based framework to self-overlapping polygons given by a triangulation. We obtain an $O(n \log n)$-time, $O(n)$-space algorithm that solves both versions, and prove an $\Omega(n \log n)$ lower bound in the algebraic computation-tree model by a reduction from the δ-closeness problem. This shows that our algorithm is optimal in that model.

Because of page limits, the main body of the paper focuses on the lattice formulation (Section 3) and dynamic program (Section 4) for simple polygons. Omitted proofs and full details for the convex and self-overlapping cases are deferred to the full version.

2 Preliminaries

Let P be a polygon with n vertices in the plane, given as a list of vertices in counterclockwise order along its boundary. A *partition* of P is a set of connected pieces with pairwise disjoint interiors whose union equals P. The *cardinality* of a partition is the number of its pieces.

For a set $X \subseteq \mathbb{R}^2$, we denote by ∂X its boundary, by $\mathsf{int}(X)$ its interior, and by $\mathsf{cl}(X)$ its closure. We regard a polygon as the union of its interior and boundary; in particular, $\mathsf{cl}(P) = P$, ∂P is the boundary of P, and $\mathsf{int}(P)$ is its interior.

For a point $p \in \mathbb{R}^2$, let $x(p)$ and $y(p)$ denote its x- and y-coordinates, respectively. For any two points $p, q \in \mathbb{R}^2$, we use pq to denote the line segment connecting p and q, and write $|pq|$ for its length. We call pq a *cut* in P if $p, q \in \partial P$ and the interior points of pq lie in $\mathsf{int}(P)$. If ℓ is a vertical cut, we denote by $x(\ell)$ the x-coordinate of ℓ.

Let O be a totally ordered set. A subset $I \subseteq O$ is an *interval* of O if for all $x, y, z \in O$ with $x \leq z \leq y$, the condition $x, y \in I$ implies $z \in I$. A *closed interval* of O is an interval of the form $[a, b] = \{x \in O \mid a \leq x \leq b\}$. We denote by $\mathbf{I}_O$ the set of all closed intervals of O.

If $O \subseteq \mathbb{R}$ with the induced order, then for a closed interval $I = [a, b] \in \mathbf{I}_O$, we define its length by $|I| \coloneqq b - a$. The set O is *locally finite* if every closed interval $[a, b] \subseteq O$ contains only finitely many elements of O; for example, $\mathbb{Z}$ is locally finite, whereas $\mathbb{Q}$ and $\mathbb{R}$ are not.

We use the notation $[m] \coloneqq \{1, 2, \ldots, m\}$ for a positive integer m. For a finite set A, we use $|A|$ to denote its cardinality.

Computational model. We work in the real-RAM model, which supports unit-cost arithmetic $(+, -, \times, \div)$ and comparisons on real numbers. The n vertices of the input polygon P are given by real coordinates $(x_i, y_i)_{i=1}^n$. We allow a restricted use of floor function: for each input x_i, the integer part $\lfloor x_i \rfloor$ is also given and can be accessed in constant time, which increases the input size only by a constant factor. We do not allow the floor function on arbitrary real values, since this would make the model unrealistically powerful [20]. The integer labels do not increase the computational power for the problems we consider.

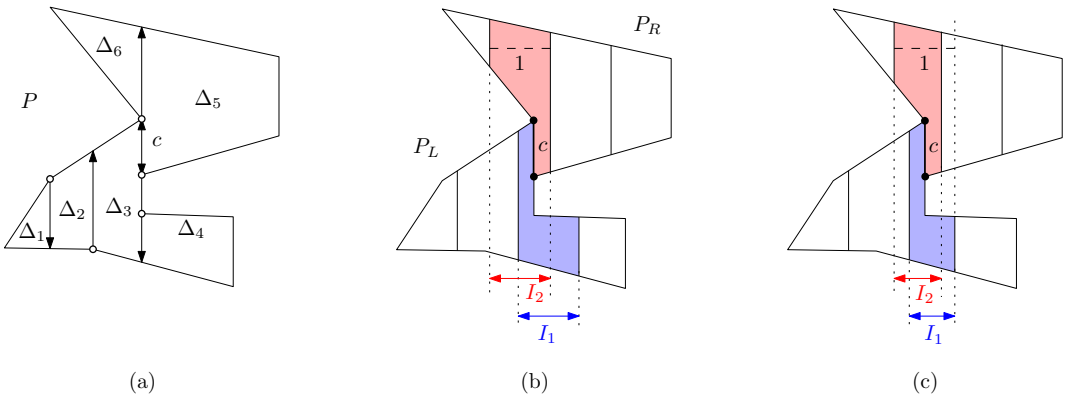
Our algorithms take floor/ceiling of differences between x-coordinates of input vertices. In our model, these are supported in constant time because the integer parts of all input x-coordinates are given. Let $(x_1, x_2, \ldots, x_n)$ be the x-coordinates of the vertices of P. For each $i \in [n]$, let $f_i = \lfloor x_i \rfloor$ (the integer part) and $r_i = x_i - f_i$ (the fractional part). Then, $\lfloor x_i - x_j \rfloor = f_i - f_j - \mathbf{1}[r_i < r_j]$, where $\mathbf{1}[\cdot]$ is the indicator function and its value is computed by a single comparison. For computing ceilings, we use the identity $\lceil x_i - x_j \rceil = -\lfloor x_j - x_i \rfloor$.

3 Lattice formulation for orthogonal strip partitions of simple polygons

In this section, P is a simple polygon with n vertices, stored in an array in counterclockwise order along its boundary. We give an $O(n \log n)$-time, $O(n)$-space algorithm for both $\mathrm{OSP}_V^{\mathrm{simp}}(P)$ and $\mathrm{OSP}_R^{\mathrm{simp}}(P)$.

We first compute a vertical trapezoidal decomposition of P and its dual graph in $O(n)$ time using the algorithm of Chazelle [9]; see Figure 2(a). Let $V = \{\Delta_1, \Delta_2, \ldots, \Delta_m\}$ be the set of trapezoids and let $G = (V, E)$ be its dual graph. Then $m = O(n)$ and G is a tree.

Let c be a vertical cut in the trapezoidal decomposition. Cutting P along c yields two subpolygons $P_L, P_R \subset P$, where P_L is the part of P lying to the left of c and P_R is the part lying to the right. Consider optimal strip partitions of P_L and P_R in which every strip has horizontal width at most 1. Taking the union of these two partitions, with c as a



(a) (b) (c)

Figure 2 (a) Vertical trapezoidal decomposition of P into $V = \{\Delta_1, \ldots, \Delta_6\}$, and decomposition of P into P_L and P_R by a vertical cut c of V. (b-c) Merging optimal strip partitions of P_L and P_R incident to c may or may not require an additional cut, depending on whether $|I_1 \cup I_2| \leq 1$ or > 1.

common boundary, yields a feasible strip partition of P, implying $\mathsf{opt}(P) \leq \mathsf{opt}(P_L)+\mathsf{opt}(P_R)$. Conversely, given an optimal partition of P, restricting it to P_L and P_R yields feasible strip partitions of these subpolygons. Thus, $\mathsf{opt}(P_L)+\mathsf{opt}(P_R)-1 \leq \mathsf{opt}(P) \leq \mathsf{opt}(P_L)+\mathsf{opt}(P_R)$, and we need to determine whether the lower or upper bound is achieved in a given instance.

To analyze this, observe that in any strip partition of P_L, there is exactly one strip incident to c. For P_L, let $\mathcal{I}_c(P_L)$ denote the set of x-intervals obtained as follows: for each optimal strip partition of P_L, take the strip incident to c and project it onto the x-axis, and include the resulting interval in $\mathcal{I}_c(P_L)$. We define $\mathcal{I}_c(P_R)$ for P_R analogously. Every interval in $\mathcal{I}_c(P_L)$ and in $\mathcal{I}_c(P_R)$ contains $x(c)$, since its corresponding strip is incident to c.

When gluing optimal partitions of P_L and P_R along c, the total number of strips decreases by one only when the two strips incident to c can be merged into a single strip without violating the width constraint. In terms of x-intervals, this is equivalent to checking whether there exist intervals $I_1 \in \mathcal{I}_c(P_L)$ and $I_2 \in \mathcal{I}_c(P_R)$ such that $|I_1 \cup I_2| \leq 1$. See Figure 2.

▶ **Observation 2.** *We have $\mathsf{opt}(P) = \mathsf{opt}(P_L)+\mathsf{opt}(P_R)-1$ if and only if there exist intervals $I_1 \in \mathcal{I}_c(P_L)$ and $I_2 \in \mathcal{I}_c(P_R)$ such that $|I_1 \cup I_2| \leq 1$.*

Moreover, it suffices to consider only inclusion-minimal intervals in each family $\mathcal{I}_c(P_L)$ and $\mathcal{I}_c(P_R)$. If $(I_1, I_2) \in \mathcal{I}_c(P_L) \times \mathcal{I}_c(P_R)$ satisfies $|I_1 \cup I_2| \leq 1$ and there exists $I_1' \in \mathcal{I}_c(P_L)$ with $I_1' \subsetneq I_1$, then $|I_1' \cup I_2| \leq 1$ also holds; by symmetry, the same argument applies to $\mathcal{I}_c(P_R)$. Thus, we may restrict attention to inclusion-minimal intervals in $\mathcal{I}_c(P_L)$ and $\mathcal{I}_c(P_R)$.

▶ **Observation 3.** *To decide whether there exist $I_1 \in \mathcal{I}_c(P_L)$ and $I_2 \in \mathcal{I}_c(P_R)$ with $|I_1 \cup I_2| \leq 1$, it suffices to consider only the inclusion-minimal intervals in $\mathcal{I}_c(P_L)$ and $\mathcal{I}_c(P_R)$.*

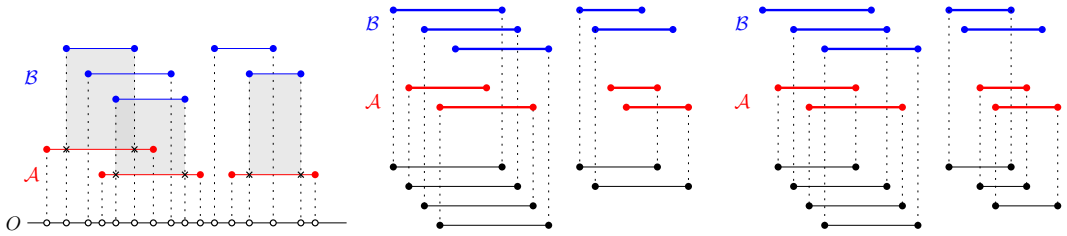
Algorithm overview and lattice formulation. Our algorithm performs a bottom-up dynamic program over the vertical trapezoidal decomposition of P and its dual tree G, rooted at an arbitrary node. For a node u of G, let Q be the subpolygon formed by the union of trapezoids in the subtree rooted at u. If u is not the root, we designate a vertical edge c of Q that serves as the interface to its parent. The associated subproblem for (Q, c) is to determine the minimum number $\mathsf{opt}(Q)$, together with the possible x-intervals of strips incident to c in optimal partitions of Q.

Observation 2 shows that, when two subpolygons are glued across c, the optimum of the combined region can be determined from the optimum of each subpolygon and the x-intervals of the strips incident to c in optimal partitions of the two subpolygons. Moreover, by Observation 3, it suffices to retain only the inclusion-minimal such intervals. Thus, for each subproblem (Q, c), we maintain the optimal value $\mathsf{opt}(Q)$ together with the inclusion-minimal intervals in $\mathcal{I}_c(Q)$.

We initialize these DP states at leaf trapezoids and combine them bottom-up at internal nodes, eventually reaching the root and obtaining $\mathsf{opt}(P)$. The crucial observation is that the update from child subproblems to a parent subproblem can be carried out entirely on these interval families. These interval families form an order-theoretic structure, namely the Clarke–Cormack–Burkowski (CCB) lattice. This lattice comes with two basic operations, meet($\wedge$) and join($\vee$). As we show next, the two basic operations naturally describe the two update situations of the dynamic program: gluing two incident strips into one, or introducing a new cut when such a gluing would violate the unit-width constraint.

3.1 Antichain completion and the Clarke–Cormack–Burkowski lattice

Let $(X, \leq)$ be a partially ordered set (poset). A subset $A \subseteq X$ is an *antichain* if no two distinct elements of A are comparable, that is, for all $a, b \in A$ with $a \neq b$, we have neither $a \leq b$ nor $b \leq a$. We denote by $\mathsf{AC}(X)$ the set of all antichains of X.



(a) $\mathcal{A} \preceq \mathcal{B}$ (b) $\mathcal{A} \wedge \mathcal{B}$ (c) $\mathcal{A} \vee \mathcal{B}$

Figure 3 Let $(\mathcal{E}_O, \preceq)$ be the antichain completion of $(\mathbf{I}_O, \supseteq)$. The figure illustrates the partial order, meet, and join operations for two antichains $\mathcal{A}, \mathcal{B} \in \mathcal{E}_O$: (a) $\mathcal{A} \preceq \mathcal{B}$; (b) $\mathcal{A} \wedge \mathcal{B}$; (c) $\mathcal{A} \vee \mathcal{B}$.

Boldi and Vigna [8] order antichains by their lower sets. For $\mathcal{A} \in \mathsf{AC}(X)$, its *lower set* is $\downarrow(\mathcal{A}) := \{x \in X \mid \exists y \in \mathcal{A} \text{ with } x \le y\}$. They define a partial order $\preceq$ on $\mathsf{AC}(X)$ by $\mathcal{A} \preceq \mathcal{B} \iff \downarrow(\mathcal{A}) \subseteq \downarrow(\mathcal{B})$. The poset $(\mathsf{AC}(X), \preceq)$ is called the *antichain completion* of $(X, \le)$.

Now let O be a totally ordered set, and let $\mathbf{I}_O$ be the set of all closed intervals of O. We order $\mathbf{I}_O$ by reverse inclusion, so $(\mathbf{I}_O, \supseteq)$ is a poset. We write $\mathcal{E}_O := \mathsf{AC}(\mathbf{I}_O)$ and view $(\mathcal{E}_O, \preceq)$ as the antichain completion of $(\mathbf{I}_O, \supseteq)$. For our purposes, it suffices to keep in mind the following description of $\mathcal{E}_O$ and $\preceq$. Each element $\mathcal{A} \in \mathcal{E}_O$ is a family of intervals in which no interval properly contains another. For $\mathcal{A}, \mathcal{B} \in \mathcal{E}_O$, the relation $\mathcal{A} \preceq \mathcal{B}$ means that every interval in $\mathcal{A}$ contains at least one interval in $\mathcal{B}$.

Clarke–Cormack–Burkowski (CCB) lattice. A poset is a *lattice* if every pair of elements has a unique least upper bound (join) and a unique greatest lower bound (meet). For a locally finite, totally ordered set O, Boldi and Vigna [8] show that the antichain completion $\mathcal{E}_O$ forms a complete lattice. This structure, known as the *Clarke–Cormack–Burkowski lattice*, is named after the work of Clarke, Cormack, and Burkowski on minimal-interval semantics in information retrieval [12].

Let O be a locally finite, totally ordered set, so $\mathcal{E}_O$ is a CCB lattice. Each $\mathcal{A} \in \mathcal{E}_O$ is a family of intervals of O in which no interval properly contains another. For a family $\mathcal{F}$ of intervals, let $\min_{\subseteq}(\mathcal{F}) := \{I \in \mathcal{F} \mid \nexists J \in \mathcal{F} \text{ with } J \subsetneq I\}$ be the set of inclusion-minimal intervals in $\mathcal{F}$. The join and meet of $\mathcal{A}, \mathcal{B} \in \mathcal{E}_O$ are defined, respectively, as

$$\mathcal{A} \vee \mathcal{B} := \min_{\subseteq}(\mathcal{A} \cup \mathcal{B}) \quad \text{and} \quad \mathcal{A} \wedge \mathcal{B} := \min_{\subseteq}\big\{[\min\{\ell, \ell'\}, \max\{r, r'\}] \mid [\ell, r] \in \mathcal{A},\ [\ell', r'] \in \mathcal{B}\big\}.$$

See Figure 3 for an illustration of the partial order, meet, and join of two antichains.

Boldi and Vigna also show that $\mathcal{E}_O$ is a *completely distributive* lattice: arbitrary meets distribute over arbitrary joins in this lattice. In particular, for any family $(\mathcal{A}_i)_{i \in I} \subseteq \mathcal{E}_O$ and any $\mathcal{B} \in \mathcal{E}_O$, we have $\left(\bigvee_{i \in I} \mathcal{A}_i\right) \wedge \mathcal{B} = \bigvee_{i \in I} (\mathcal{A}_i \wedge \mathcal{B})$.

Lattice-based representation of strip partitions. For a vertical cut c, the sets $\min_{\subseteq}(\mathcal{I}_c(P_L))$ and $\min_{\subseteq}(\mathcal{I}_c(P_R))$ are antichains of intervals, since no interval in each set properly contains another. By Observation 3, these antichains suffice to decide whether the strips incident to c can be merged. In fact, the equality $\mathsf{opt}(P) = \mathsf{opt}(P_L) + \mathsf{opt}(P_R) - 1$ holds if and only if the meet $\min_{\subseteq}(\mathcal{I}_c(P_L)) \wedge \min_{\subseteq}(\mathcal{I}_c(P_R))$ contains an interval of length at most 1. Thus, in our dynamic program, we only need to maintain, for each subpolygon Q with vertical edge c, the antichain $\mathsf{Ant}(Q, c) := \min_{\subseteq}\big(\mathcal{I}_c(Q)\big)$. In other words, $\mathsf{Ant}(Q, c)$ is the set of inclusion-minimal x-intervals of strips incident to c across all optimal strip partitions of Q.

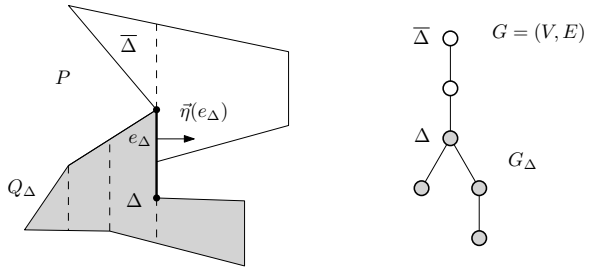


Figure 4 Subproblem structure $(Q_\Delta, e_\Delta, \vec{\eta}(e_\Delta))$ is defined for every non-root node Δ of $G = (V, E)$.

At first glance, these antichains appear to be families of intervals of $\mathbb{R}$, which is not locally finite. However, as we will show, once we fix some $\varepsilon \geq 0$, every interval that arises in our algorithm has endpoints of the form $x(p) + m$ or $x(p) + m \pm \varepsilon$, where p is a vertex of P and $m \in \mathbb{Z}$. Thus all endpoints lie in

$$\mathbb{Z}(P, \varepsilon) := \{\, x(p) + m - \varepsilon,\ x(p) + m,\ x(p) + m + \varepsilon \mid p \text{ is a vertex of } P,\ m \in \mathbb{Z} \,\},$$

a locally finite, totally ordered set. Consequently, every antichain maintained by our dynamic program lies in the CCB lattice $\mathcal{E}_{\mathbb{Z}(P,\varepsilon)}$. This lattice is completely distributive.

3.2 Subproblem structures in the refined dual tree

We now describe the subproblem structure underlying our dynamic program for simple polygons, formulated in terms of antichains of intervals.

We associate each subproblem with a pair (Q, e), where Q is a union of trapezoids from the vertical trapezoidal decomposition V and e is a vertical boundary edge of Q. (Recall that $\mathcal{I}_e(Q)$ denotes the family of x-intervals of all strips incident to edge e across all optimal strip partitions of Q.) We compress this family into an antichain by keeping only inclusion-minimal intervals: $\mathsf{Ant}(Q, e) := \min_{\subseteq}\big(\mathcal{I}_e(Q)\big)$. Every interval in $\mathsf{Ant}(Q, e)$ has length at most 1, and each such interval includes the common x-coordinate $x(e)$.

Let $G = (V, E)$ be the dual graph of the trapezoidal decomposition V, and fix an arbitrary leaf $\overline{\Delta} \in V$ as the root. For each node $\Delta \in V$, let G_Δ be the subtree of G rooted at Δ and let $Q_\Delta \subseteq P$ be the subpolygon obtained as the union of the trapezoids in G_Δ. A vertical cut of the trapezoidal decomposition that lies on ∂Q_Δ is called a *vertical interface* of Q_Δ.

Because G is a tree, all the vertical interfaces of Q_Δ lie on a single vertical edge of Q_Δ for $\Delta \neq \overline{\Delta}$. Let e_Δ denote this vertical edge. Note that e_Δ is contained in $\partial\Delta$. We define the *normal direction* of e_Δ, denoted $\vec{\eta}(e_\Delta)$, to be right if e_Δ lies on the right side of Δ, and left if it lies on the left side. We refer to the triple $(Q_\Delta, e_\Delta, \vec{\eta}(e_\Delta))$ as the *subproblem structure* associated with Δ. See Figure 4.

Refining G into a binary tree $\widetilde{G}$. We transform the dual graph G of V into a rooted binary tree $\widetilde{G}$ by locally refining each star centered at a trapezoid. Let $\Delta \in V$ be a trapezoid in V with subproblem structure $(Q_\Delta, e_\Delta, \vec{\eta}(e_\Delta))$, and assume without loss of generality that $\vec{\eta}(e_\Delta) = \mathsf{right}$. Suppose Δ has p children incident to its left side and q children on its right. Let $(\Delta_i^L, e_i^L, \vec{\eta}(e_i^L))_{i=1}^p$ (resp. $(\Delta_j^R, e_j^R, \vec{\eta}(e_j^R))_{j=1}^q$) be the subproblem structures associated with its left (resp. right) children. By construction, each $\vec{\eta}(e_i^L) = \mathsf{right}$ and each $\vec{\eta}(e_i^R) = \mathsf{left}$.

First, we construct a binary tree T_L whose leaves are the left children of Δ. If $p = 0$, we do not create T_L; if $p = 1$, then T_L is a single node Δ_1^L. For $p \geq 2$, we create $(p - 1)$ new internal nodes $u_2, \ldots, u_p$ and set each u_i as the parent of Δ_i^L and u_{i-1} for each $i = 2, \ldots, p$, with $u_1 := \Delta_1^L$. Symmetrically, we construct a binary tree T_R for the right children of Δ.

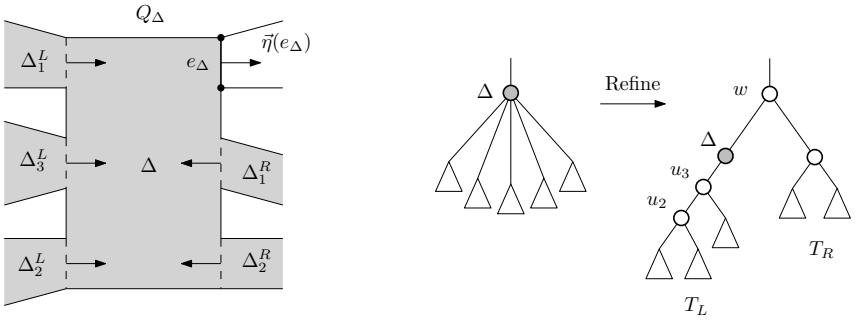


Figure 5 (a) The trapezoid Δ has three left children and two right children in the subtree G_Δ. (b) Refining the local star centered at Δ into a binary tree.

Next, we remove all edges from Δ to its children in G. If T_L exists (i.e., $p \geq 1$), we attach the root of T_L as the sole child of Δ. If T_R exists, let w be a new internal node whose children are the root of T_R and Δ; otherwise, let $w := \Delta$. If Δ has a parent z in G, we replace the edge between z and Δ by an edge between z and w. The case $\vec{\eta}(e_\Delta) = \mathsf{left}$ is handled symmetrically. See Figure 5 for an illustration.

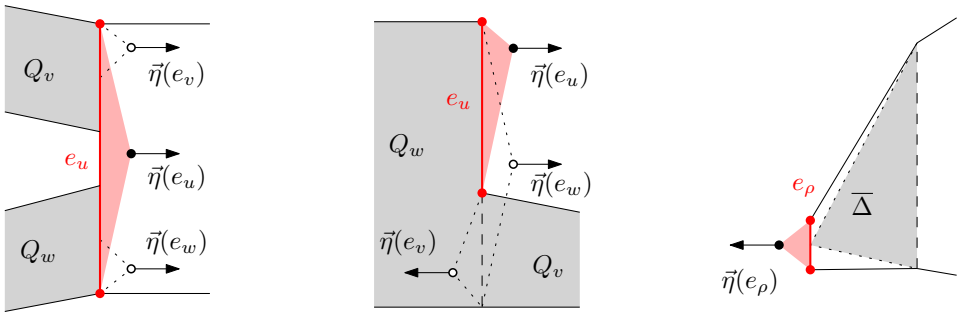
Processing all trapezoids in postorder replaces every star subgraph in G with a binary tree. At each step, only the adjacency between Δ and its children changes; the subtrees rooted at those children remain intact. This refinement adds only $O(n)$ new internal nodes in total, and $\widetilde{G}$ can be constructed in $O(n)$ time. We refer to the original nodes of G as *trapezoid nodes* and the new internal nodes as *bridge nodes*. By construction, every trapezoid node in $\widetilde{G}$ has at most one child, and every bridge node has exactly two children.

Subproblem structures in $\widetilde{G}$. We lift the subproblem structures defined for nodes of G to the nodes of the refined tree $\widetilde{G}$. For each non-root node u, we associate a triple $(Q_u, e_u, \vec{\eta}(e_u))$, where Q_u is a weakly simple subpolygon of P, e_u is a vertical edge on ∂Q_u, and $\vec{\eta}(e_u) \in \{\mathsf{left}, \mathsf{right}\}$ is the outward normal direction of e_u with respect to Q_u. If u is a trapezoid node corresponding to a trapezoid $\Delta \in V$, we inherit its subproblem structure by setting $(Q_u, e_u, \vec{\eta}(e_u)) := (Q_\Delta, e_\Delta, \vec{\eta}(e_\Delta))$.

Suppose that u is a bridge node with children v and w. Assume that $(Q_v, e_v, \vec{\eta}(e_v))$ and $(Q_w, e_w, \vec{\eta}(e_w))$ are already defined. Let s be the minimal vertical segment spanning both e_v and e_w, and define $Q_u := Q_v \cup s \cup Q_w$. If $\vec{\eta}(e_v) = \vec{\eta}(e_w)$, then u is a *same-side bridge*. We set $e_u := s$ and $\vec{\eta}(e_u) := \vec{\eta}(e_v)$. In this case, Q_u has boundary edges that touch each other without crossing, so Q_u is only weakly simple.

If $\vec{\eta}(e_v) \neq \vec{\eta}(e_w)$, then u is a *cross-side bridge*. In this case, Q_u is always simple and one of e_v, e_w properly contains the other. If $e_v \subsetneq e_w$, we set $e_u := e_w \setminus e_v$ and $\vec{\eta}(e_u) := \vec{\eta}(e_w)$; otherwise, $e_u := e_v \setminus e_w$ and $\vec{\eta}(e_u) := \vec{\eta}(e_v)$. Figure 6(a-b) illustrates the subproblem structures at bridge nodes.

For the root node ρ of $\widetilde{G}$, which corresponds to the leaf $\overline{\Delta}$ of G, the region $Q_\rho = P$ has no vertical interface on its boundary. Then, we cannot directly define a subproblem structure for ρ. Let e be the vertical side of $\overline{\Delta}$ that is not incident to any other trapezoid of V. Note that e may be a single vertex if $\overline{\Delta}$ is a triangle. In this degenerate case, we conceptually replace e with a very short vertical segment, obtained by extending it slightly in both vertical directions. This extension preserves the x-interval of $\overline{\Delta}$, keeps P simple, and does not affect



(a) (b) (c)

Figure 6 The subproblem structures $(Q_u, e_u, \vec{\eta}(e_u))$ in $\widetilde{G}$ when u is (a) a same-side bridge, (b) a cross-side bridge, and (c) the root; attaching the cap $T_\rho(\varepsilon)$ along e_ρ yields the simple polygon Q_ρ^ε.

any strip partition of P. We set $e_\rho := e$ and define $\vec{\eta}(e_\rho)$ as left if e lies on the left side of $\overline{\Delta}$, or right if e lies on the right side. This yields the subproblem structure $(Q_\rho, e_\rho, \vec{\eta}(e_\rho))$ for the root node ρ. See Figure 6(c).

▶ **Lemma 4.** *For every node u of $\widetilde{G}$, the subproblem structure $(Q_u, e_u, \vec{\eta}(e_u))$ is well-defined: Q_u is a weakly simple subpolygon of P, e_u is a vertical edge of Q_u with positive length, and $\vec{\eta}(e_u)$ is the outward normal direction of e_u with respect to Q_u.*

Reducing weakly simple subpolygons to simple ones. By Lemma 4, every node u of $\widetilde{G}$ has an associated subproblem structure $(Q_u, e_u, \vec{\eta}(e_u))$. If u is a trapezoid node or a cross-side bridge node, then Q_u is simple. If u is a same-side bridge node, however, Q_u is only weakly simple. To handle all subproblems in the simple polygon setting, we convert Q_u into a simple polygon by attaching a thin triangular cap along e_u on the outside of Q_u.

Let q_u be any point on the edge e_u, excluding its endpoints. For sufficiently small $\varepsilon > 0$, let $q_u(\varepsilon)$ be the point obtained by shifting q_u by ε units in the direction of $\vec{\eta}(e_u)$. Let $T_u(\varepsilon)$ be the triangle with base e_u and apex $q_u(\varepsilon)$.

▶ **Lemma 5.** *For each node u of $\widetilde{G}$, there exists $\varepsilon_0 > 0$ such that $Q_u \cup T_u(\varepsilon)$ is a simple polygon for all $0 < \varepsilon \le \varepsilon_0$.*

Let $(x_i)_{i=1}^n$ be the x-coordinates of the vertices of P, and let $\mathbb{Z}(P, 0) := \{\, x_i + m \mid i \in [n],\, m \in \mathbb{Z} \,\}$. Let $\delta > 0$ be the minimum positive distance between two distinct points of $\mathbb{Z}(P, 0)$. From Lemma 5, for each node u, there exists $\varepsilon_0(u) > 0$ such that $Q_u \cup T_u(\varepsilon)$ is simple for all $0 < \varepsilon \le \varepsilon_0(u)$. Choose a particular $\varepsilon > 0$ satisfying $\varepsilon \le \min_u \varepsilon_0(u)$ and $\varepsilon < \frac{\delta}{2}$. Note that δ and ε are not computed explicitly; we only fix such values conceptually.

Now we define the capped subpolygon $Q_u^\varepsilon := Q_u \cup T_u(\varepsilon)$, which is a simple polygon. Figure 6 illustrates how the cap $T_u(\varepsilon)$ attaches to e_u for each node type. Let $\mathsf{opt}^\varepsilon(Q_u)$ denote the minimum number of strips in an optimal strip partition of Q_u^ε. Among all optimal strip partitions of Q_u^ε, let $\mathcal{S}^\varepsilon(Q_u, e_u)$ be the set of strips incident to the apex $q_u(\varepsilon)$, and define $\mathsf{Ant}^\varepsilon(Q_u, e_u) := \min_\subseteq \{ I_x(S) \mid S \in \mathcal{S}^\varepsilon(Q_u, e_u) \}$, where $I_x(S)$ is the interval obtained by projecting S onto the x-axis. Then $\mathsf{Ant}^\varepsilon(Q_u, e_u)$ is an antichain of intervals. By definition, every interval in $\mathsf{Ant}^\varepsilon(Q_u, e_u)$ is realized as the x-projection of a strip incident to $q_u(\varepsilon)$ in some optimal partition of Q_u^ε. Although the family $\mathcal{S}^\varepsilon(Q_u, e_u)$ may be infinite, Algorithm 1 and Lemma 9 together imply that $\mathsf{Ant}^\varepsilon(Q_u, e_u)$ is finite. We define the dynamic-programming

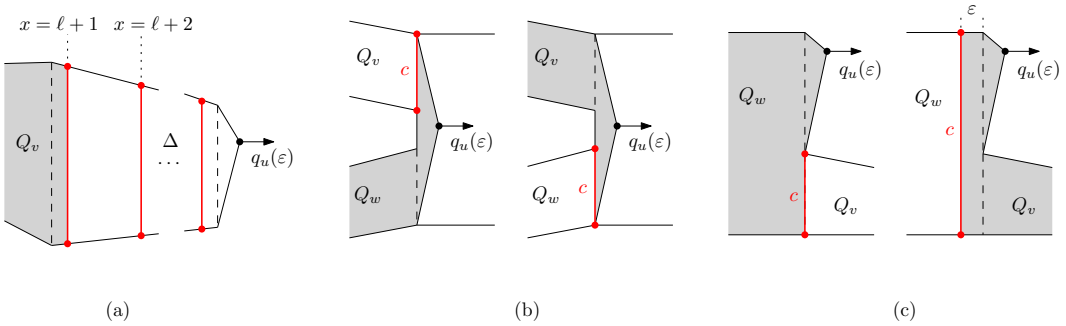


Figure 7 Additional vertical cuts are inserted when no strip can be merged or extended across the node: (a) trapezoid node, (b) same-side bridge node, and (c) cross-side bridge node.

subproblem $\mathrm{OSP}^{\varepsilon}(Q_u, e_u)$ to be the task of computing both $\mathsf{opt}^{\varepsilon}(Q_u)$ and $\mathsf{Ant}^{\varepsilon}(Q_u, e_u)$. The pair $\big(\mathsf{opt}^{\varepsilon}(Q_u), \mathsf{Ant}^{\varepsilon}(Q_u, e_u)\big)$ is called the *DP state* at node u; its first entry is the *value component*, and its second entry is the *antichain component*.

4 Dynamic programming algorithm and complexity

We process the nodes of the rooted binary tree $\widetilde{G}$ in bottom-up order. For each node u, after the subproblems for all of its descendants have been solved, we compute the DP state of $\mathrm{OSP}^{\varepsilon}(Q_u, e_u)$ using the update rule for either a trapezoid node or a bridge node.

During this process, we maintain the following invariant for each node u of $\widetilde{G}$. The value $\mathsf{opt}^{\varepsilon}(Q_u)$ is the minimum number of strips in a strip partition of Q_u^{ε}. Let $\mathcal{S}^{\varepsilon}(Q_u, e_u)$ be the set of strips incident to the apex $q_u(\varepsilon)$ among all optimal partitions of Q_u^{ε}. Then $\mathsf{Ant}^{\varepsilon}(Q_u, e_u)$ consists of the inclusion-minimal x-intervals of the strips in $\mathcal{S}^{\varepsilon}(Q_u, e_u)$.

4.1 Bottom-up evaluation on the refined tree

The update procedure is described in detail in Algorithm 1. Throughout the algorithm, we assume that $\vec{\eta}(e_u) = \mathsf{right}$; the case $\vec{\eta}(e_u) = \mathsf{left}$ is symmetric. We use the low-pass operator Filter_1 on the family of intervals. For $\mathcal{F} \subseteq \mathbf{I}_{\mathbb{R}}$, let $\mathrm{Filter}_1(\mathcal{F}) := \{\, I \in \mathcal{F} \mid |I| \le 1 \,\}$, which discards intervals of length greater than 1.

Trapezoid nodes. Let u be a trapezoid node corresponding to a trapezoid Δ. If u has a child v in $\widetilde{G}$, the DP state $(\mathsf{opt}^{\varepsilon}(Q_v), \mathsf{Ant}^{\varepsilon}(Q_v, e_v))$ has already been computed. If u is a leaf, we introduce a virtual child v by taking e_v to be the vertical side of Δ opposite to e_u, letting Q_v be the degenerate trapezoid consisting of e_v, and setting $\vec{\eta}(e_v) = \vec{\eta}(e_u)$; then $\mathsf{opt}^{\varepsilon}(Q_v) = 1$ and $\mathsf{Ant}^{\varepsilon}(Q_v, e_v)$ consists of a single interval of length ε adjacent to $x(e_v)$.

The update at a trapezoid node proceeds as follows. Let $[a, b]$ be the x-interval of Δ, where $a = x(e_v)$ and $b = x(e_u)$. For each strip $S \in \mathcal{S}^{\varepsilon}(Q_v, e_v)$, we remove the old cap $T_v(\varepsilon)$, insert Δ into the polygon, and then attach the new cap $T_u(\varepsilon)$ along e_u. If $[\ell, r]$ is the x-interval of S, then the extended strip has x-interval $[\min\{\ell, a\}, \max\{r, b + \varepsilon\}]$, which is exactly the interval obtained by taking the meet of $\{[\ell, r]\}$ with $\{[a, b + \varepsilon]\}$. Thus, the inclusion-minimal x-intervals of all feasible extensions are exactly $\mathrm{Filter}_1\big(\mathsf{Ant}^{\varepsilon}(Q_v, e_v) \wedge \{[a, b + \varepsilon]\}\big)$.

If at least one extended strip survives, then we do not introduce any new cut at u. If none survive, then no strip can be extended through Δ without violating the unit-width constraint. Let $[\ell, r]$ be the x-interval of the rightmost strip in $\mathcal{S}^{\varepsilon}(Q_v, e_v)$. If no extended

Algorithm 1 DP update at a node u (case $\vec{\eta}(e_u) = \mathsf{right}$).

Input: node u of $\widetilde{G}$ and the DP states at its children
Output: DP state $(\mathsf{opt}^\varepsilon(Q_u), \mathsf{Ant}^\varepsilon(Q_u, e_u))$ at u

if *u is a trapezoid node with child v* **then**
  TrapezoidUpdate(u, v)
else if *u is a bridge node with children v, w* **then**
  BridgeUpdate(u, v, w)

procedure *TrapezoidUpdate*(u, v)
  Let Δ be the trapezoid of u with x-interval $[a, b]$
  $\mathsf{Ant}^\varepsilon(Q_u, e_u) \leftarrow \mathrm{Filter}_1\big(\mathsf{Ant}^\varepsilon(Q_v, e_v) \wedge \{[a, b + \varepsilon]\}\big), \quad \mathsf{opt}^\varepsilon(Q_u) \leftarrow \mathsf{opt}^\varepsilon(Q_v)$
  if $\mathsf{Ant}^\varepsilon(Q_u, e_u) = \varnothing$ **then**
    let $[\ell, r]$ be the interval in $\mathsf{Ant}^\varepsilon(Q_v, e_v)$ with maximum left endpoint ℓ
    $k \leftarrow \lfloor b + \varepsilon - \ell \rfloor$
    $\mathsf{Ant}^\varepsilon(Q_u, e_u) \leftarrow \{[\ell + k,\ b + \varepsilon]\}, \quad \mathsf{opt}^\varepsilon(Q_u) \leftarrow \mathsf{opt}^\varepsilon(Q_v) + k$

procedure *BridgeUpdate*(u, v, w)
  $\mathsf{Ant}^\varepsilon(Q_u, e_u) \leftarrow \mathrm{Filter}_1\big(\mathsf{Ant}^\varepsilon(Q_v, e_v) \wedge \mathsf{Ant}^\varepsilon(Q_w, e_w)\big)$
  $\mathsf{opt}^\varepsilon(Q_u) \leftarrow \mathsf{opt}^\varepsilon(Q_v) + \mathsf{opt}^\varepsilon(Q_w) - 1$
  if $\mathsf{Ant}^\varepsilon(Q_u, e_u) = \varnothing$ **then**
    $\mathsf{Ant}^\varepsilon(Q_u, e_u) \leftarrow \mathsf{Ant}^\varepsilon(Q_v, e_v) \vee \mathsf{Ant}^\varepsilon(Q_w, e_w)$
    $\mathsf{opt}^\varepsilon(Q_u) \leftarrow \mathsf{opt}^\varepsilon(Q_v) + \mathsf{opt}^\varepsilon(Q_w)$

strip survives, we insert a series of vertical cuts in $\Delta \cup T_u(\varepsilon)$ at x-coordinates $\ell + 1, \ell + 2, \ldots$, decomposing the extended region into unit-width strips (with the last piece possibly shorter than 1). We then increase $\mathsf{opt}^\varepsilon(Q_u)$ by the number of new strips, and define $\mathsf{Ant}^\varepsilon(Q_u, e_u)$ to consist of the single x-interval of the rightmost suffix strip incident to $q_u(\varepsilon)$.

Bridge nodes. Let u be a bridge node of $\widetilde{G}$ with children v and w. At a bridge node, we consider pairs of strips $S_v \in \mathcal{S}^\varepsilon(Q_v, e_v)$ and $S_w \in \mathcal{S}^\varepsilon(Q_w, e_w)$, and ask whether they can be combined into a single strip incident to $q_u(\varepsilon)$. By the choice of ε, the x-interval of each of S_v and S_w already contains $[x(e_u) - \varepsilon,\ x(e_u) + \varepsilon]$. Hence, replacing the child caps $T_v(\varepsilon), T_w(\varepsilon)$ with the cap $T_u(\varepsilon)$ does not change either x-interval.

For a same-side bridge, this combination is obtained by removing the caps $T_v(\varepsilon)$ and $T_w(\varepsilon)$, gluing the two strips along e_u, and attaching the cap $T_u(\varepsilon)$. For a cross-side bridge, we remove the two caps, take the union of the resulting strips, and attach $T_u(\varepsilon)$ along e_u. In either case, if $[\ell, r]$ and $[\ell', r']$ are the x-intervals of S_v and S_w, respectively, then the combined region has x-interval $[\min\{\ell, \ell'\}, \max\{r, r'\}]$. Thus, the inclusion-minimal x-intervals of all feasible combinations are exactly $\mathrm{Filter}_1\big(\mathsf{Ant}^\varepsilon(Q_v, e_v) \wedge \mathsf{Ant}^\varepsilon(Q_w, e_w)\big)$.

If at least one pair of strips yields a feasible strip, then no additional cut is introduced at u, and the feasible combined strips form $\mathcal{S}^\varepsilon(Q_u, e_u)$. Otherwise, we insert a vertical cut near e_u so that strips from the two child subproblems cannot be combined across u. The strips from Q_v and Q_w are then extended separately to $q_u(\varepsilon)$. Hence, $\mathsf{Ant}^\varepsilon(Q_u, e_u) = \mathsf{Ant}^\varepsilon(Q_v, e_v) \vee \mathsf{Ant}^\varepsilon(Q_w, e_w)$.

Cut placement at nodes. We now describe how new cuts are placed when strips cannot be merged or extended at a node u without violating the unit-width constraint.

If u is a trapezoid node, no strip extends through its trapezoid Δ, so we place vertical cuts in $\Delta \cup T_u(\varepsilon)$. They start at $x = \ell + 1$, where $[\ell, r]$ is the interval with the maximum left endpoint in $\mathsf{Ant}^\varepsilon(Q_v, e_v)$, and continue at unit distance. See Figure 7(a).

Now suppose that u is a bridge node with children v and w. If some strip in $\mathcal{S}^\varepsilon(Q_v, e_v)$ can merge with one in $\mathcal{S}^\varepsilon(Q_w, e_w)$ into a unit-width strip, no new cut is added at u. Otherwise, we place a single vertical cut near e_u to prevent any strip from Q_v from merging with a strip from Q_w, so that all strips are extended to $q_u(\varepsilon)$ separately. See Figure 7(b–c).

The position of this cut depends on the bridge type and on the child from which each x-interval in $\mathsf{Ant}^\varepsilon(Q_u, e_u)$ originates. For a same-side bridge, the cut is placed along e_w if the interval comes from $\mathsf{Ant}^\varepsilon(Q_v, e_v)$ and along e_v if it comes from $\mathsf{Ant}^\varepsilon(Q_w, e_w)$. For a cross-side bridge, either $e_v \subsetneq e_w$ or $e_w \subsetneq e_v$. Assume $e_v \subsetneq e_w$. If the interval comes from $\mathsf{Ant}^\varepsilon(Q_w, e_w)$, the cut is placed along e_v. Otherwise it is placed inside Q_w at x-distance ε from e_u, blocking any strip from Q_w^ε from reaching $q_u(\varepsilon)$; see Figure 7(c). The case $e_w \subsetneq e_v$ is symmetric. Note that inserting these cuts at bridge nodes does not increase the x-interval of any strip from Q_v^ε or Q_w^ε; we prove this in the proof of Lemma 6.

Every interval appearing in an antichain $\mathsf{Ant}^\varepsilon(Q_u, e_u)$ has endpoints of the form $x(p) + m$ or $x(p) + m \pm \varepsilon$, where p is a vertex of P and $m \in \mathbb{Z}$. Thus all endpoints lie in the locally finite, totally ordered set $\mathbb{Z}(P, \varepsilon)$, and each antichain maintained by our DP is an element of the Clarke–Cormack–Burkowski lattice $\mathcal{E}_{\mathbb{Z}(P,\varepsilon)}$. The update rules are written using meet ($\wedge$) and join ($\vee$) in this lattice, and the outputs of these operations remain in $\mathcal{E}_{\mathbb{Z}(P,\varepsilon)}$.

Since $\varepsilon > 0$ is chosen sufficiently small, the values $\mathsf{opt}(Q_u)$ and $\mathsf{opt}^\varepsilon(Q_u)$ differ by at most one. This difference occurs only when, in every optimal partition of Q_u^ε, the only strip incident to $q_u(\varepsilon)$ is the cap $T_u(\varepsilon)$; in this case we have $\mathsf{Ant}^\varepsilon(Q_u, e_u) = \{[x(e_u),\ x(e_u) + \varepsilon]\}$.

In the DP state, we store an endpoint of the form $x(p) + m$ or $x(p) + m \pm \varepsilon$ as the value $x(p) + m$ together with a single bit indicating whether the ε–offset is present. Since $\varepsilon > 0$ is chosen sufficiently small, all order and width tests used by the algorithm can be decided from the base values and the flags. For example, if $a, b \in \mathbb{Z}(P, 0)$ and $m \in \mathbb{Z}$, then

$$\begin{array}{ccccccc} b - a - 2\varepsilon > m & \iff & b - a > m, & \qquad & b - a + 2\varepsilon > m & \iff & b - a \geq m, \\ \lfloor (b - a) + 2\varepsilon \rfloor & = & \lfloor b - a \rfloor, & & \lfloor (b - a) - 2\varepsilon \rfloor & = & \lceil b - a \rceil - 1. \end{array}$$

▶ **Lemma 6.** *When Algorithm 1 is executed on the tree $\widetilde{G}$ in bottom-up order, it correctly computes, for every node u of $\widetilde{G}$, the pair $(\mathsf{opt}^\varepsilon(Q_u), \mathsf{Ant}^\varepsilon(Q_u, e_u))$, where $\mathsf{opt}^\varepsilon(Q_u)$ is the number of strips in an optimal partition of Q_u^ε, and $\mathsf{Ant}^\varepsilon(Q_u, e_u)$ is the antichain of x-intervals of the strips incident to $q_u(\varepsilon)$ across all optimal partitions of Q_u^ε.*

By Lemma 6, Algorithm 1 correctly computes $(\mathsf{opt}^\varepsilon(Q_\rho), \mathsf{Ant}^\varepsilon(Q_\rho, e_\rho))$ at the root ρ of $\widetilde{G}$, where $Q_\rho = P$. By the discussion above on the relation between $\mathsf{opt}^\varepsilon(Q_u)$ and $\mathsf{opt}(Q_u)$, we obtain $\mathsf{opt}(P)$ from this pair in constant time.

Encoding cut descriptors. We show that the additional cuts placed at trapezoid and bridge nodes admit a lossless encoding from which all cut descriptors in the optimal partition can be reconstructed in $O(n)$ time using $O(n)$ extra space. Importantly, we can generate this encoding during the DP without increasing the asymptotic running time.

▶ **Lemma 7.** *The cut descriptors produced by the dynamic program admit an $O(n)$-space encoding from which all cuts in an optimal partition can be reconstructed in $O(n)$ time.*

Failure of greedy approach. One might suspect that we can discard some intervals from an antichain component before passing information to its parent. This is impossible in general: if we discard even one interval before passing the DP state upward, the reduced state no longer determines the optimum.

Let Q be a simple polygon and let e be a vertical edge of Q. Let $\vec{\eta}(e) \in \{\mathsf{left}, \mathsf{right}\}$ denote the outward normal direction of e. Let $\mathcal{P}(Q, e)$ be the set of all simple polygons P for which there exists a simple polygon R such that $P = Q \cup R$ and $Q \cap R \subseteq e$. In other words, $\mathcal{P}(Q, e)$ consists of all simple polygons that contain $(Q, e, \vec{\eta}(e))$ as a subproblem structure in the orthogonal strip partition problem.

Let $\mathsf{Ant}(Q, e)$ be the antichain component computed by Algorithm 1 for the subproblem structure $(Q, e, \vec{\eta}(e))$. A *certificate* for $(Q, e, \vec{\eta}(e))$ is a subset $\mathcal{W} \subseteq \mathsf{Ant}(Q, e)$ with the following property: there exists an algorithm that, for every $P \in \mathcal{P}(Q, e)$, computes $\mathsf{opt}(P)$ from given $\mathcal{W}$, $\mathsf{opt}(Q)$, and the remaining region $P \setminus Q$. The *certificate complexity* of $(Q, e, \vec{\eta}(e))$ is $\min\{|\mathcal{W}| \mid \mathcal{W}$ is a certificate for $(Q, e, \vec{\eta}(e))\}$. This viewpoint is standard in lower bound arguments [4], and shows whether pruning $\mathsf{Ant}(Q, e)$ loses information necessary to determine $\mathsf{opt}(P)$.

▶ **Lemma 8.** *For every integer $n \geq 1$, there exists a subproblem instance $(Q, e, \vec{\eta}(e))$ with $O(n)$ vertices whose certificate complexity is at least n.*

Although keeping all intervals in the antichain component often appears redundant, Lemma 8 shows that one cannot safely prune intervals in general. There exist instances in which every interval is necessary; hence any greedy approach that discards intervals using only information available in the subproblem fails on some instance.

4.2 Algorithmic complexity and lower bounds

We analyze the running time and space of our dynamic program. The value component is updated only by testing whether the resulting antichain is empty, and the cut descriptor is computed during the meet or join operations without extra asymptotic cost. Thus, the overall running time is dominated by the cost of the meet and join operations on antichains.

In an antichain, intervals are strictly ordered: for two distinct intervals $[a, b]$ and $[a', b']$, we have $a < a'$ if and only if $b < b'$. Thus, we store each antichain as a list of intervals sorted by their left (or right) endpoints. Every antichain produced by our lattice operations is maintained in this sorted order.

▶ **Lemma 9.** *Let $\mathcal{A}$ and $\mathcal{B}$ be finite antichains of intervals in $\mathcal{E}_{\mathbb{Z}(P,\varepsilon)}$. Then, $|\mathcal{A} \wedge \mathcal{B}| \leq |\mathcal{A}| + |\mathcal{B}| - 1$ and $|\mathcal{A} \vee \mathcal{B}| \leq |\mathcal{A}| + |\mathcal{B}|$.*

If u is a leaf node of $\widetilde{G}$, $\mathsf{Ant}^{\varepsilon}(Q_u, e_u)$ consists of a single interval. At every trapezoid node or bridge node, Algorithm 1 performs exactly one meet or one join of two antichains. At every node u of $\widetilde{G}$, Lemma 9 implies that $|\mathsf{Ant}^{\varepsilon}(Q_u, e_u)|$ is bounded by the size of the subtree rooted at u. Boldi and Vigna [7] give one-pass, optimally lazy algorithms that compute $\mathcal{A} \wedge \mathcal{B}$ and $\mathcal{A} \vee \mathcal{B}$ in time linear in $|\mathcal{A}| + |\mathcal{B}|$. Moreover, the operation Filter_1 can clearly be implemented in time linear in the number of intervals in the antichain.

Let $T(t)$ be the total running time of Algorithm 1 on a subtree with t nodes, where $T(1) = O(1)$. If u is a trapezoid node, its unique child has a subtree of size $t-1$, the antichain at u has size $O(t)$, and the update takes $O(t)$ time; thus $T(t) = T(t-1)+O(t)$. If u is a bridge node with children whose subtrees have sizes a and $t-a-1$, then the two antichains together have size $O(t)$, so the update again takes $O(t)$ time and $T(t) = T(a) + T(t-a-1) + O(t)$. In the worst case, this recurrence yields $T(n) = O(n^2)$.

To improve this bound, we work in the real-RAM model, where random access and binary search allow these operations to be implemented more efficiently than the one-pass algorithms. Let $L(\mathcal{B})$ and $R(\mathcal{B})$ denote the sorted list of left and right endpoints of intervals in $\mathcal{B}$, respectively. For each interval $I = [\ell, r] \in \mathcal{A}$, we compute the insertion ranks of ℓ in $L(\mathcal{B})$ and of r in $R(\mathcal{B})$, that is, their positions in these sorted lists.

Given two sorted lists X and Y with $|X| = k$ and $|Y| = m$, we use a selection technique of Kaplan et al. [14] to compute all insertion ranks of X in Y in $O\left(k \log \frac{m}{k}\right)$ time for $m \geq 4k$. We partition Y into blocks of size $B := \lfloor m/(4k) \rfloor$; for each $x \in X$, we first identify the block containing x and then perform a binary search within that block. The block size B itself can be computed in $O\left(\log \frac{m}{k}\right)$ comparisons by testing powers of two, so this does not change the overall running time.

The following lemma shows that the join and filtered meet of two antichains can be computed efficiently, assuming precomputed insertion ranks.

▶ **Lemma 10.** *Let $\mathcal{A}, \mathcal{B}$ be two finite antichains of intervals in $\mathcal{E}_{\mathbb{Z}(P,\varepsilon)}$ such that every interval in $\mathcal{A} \cup \mathcal{B}$ has length at most 1. Assume that, for every interval $I \in \mathcal{A}$, the insertion ranks of its endpoints among those of $\mathcal{B}$ are available in constant time. Then, $\mathcal{A} \vee \mathcal{B}$ and $\mathrm{Filter}_1(\mathcal{A} \wedge \mathcal{B})$ can be computed in $O(|\mathcal{A}|)$ time.*

Let $|\mathcal{A}| = k$ and $|\mathcal{B}| = m$ with $k \leq m$. We first compute the insertion ranks of all endpoints of intervals in $\mathcal{A}$ among $L(\mathcal{B})$ and $R(\mathcal{B})$ in $O\left(k \log \frac{m}{k}\right)$ time. If $4k \leq m$, we then compute $\mathrm{Filter}_1(\mathcal{A} \wedge \mathcal{B})$ and $\mathcal{A} \vee \mathcal{B}$ in $O(k)$ time. If $k \leq m \leq 4k$, we instead use the linear-time algorithm of Boldi and Vigna, which runs in $O(m + k) = O(k)$ time.

Let $T(t)$ be the running time of Algorithm 1 on a subtree of size t. For a trapezoid node with a unique child, one antichain in the meet is a singleton, so the update takes $O(\log t)$ time and $T(t) = T(t-1) + O(\log t)$. For a node whose children have subtree sizes s and $t-s-1$ with $s \leq t-s-1$, the cost at that node is $O\left(s \log \frac{t-s}{s}\right)$, so $T(t) = T(s) + T(t-s-1) + O\left(s \log \frac{t-s}{s}\right)$. These recurrences give $T(n) = O(n \log n)$ in the worst case.

▶ **Theorem 11.** *For a simple polygon P with n vertices, the problems $OSP_V^{\mathrm{simp}}(P)$ and $OSP_R^{\mathrm{simp}}(P)$ can be solved in $O(n \log n)$ time using $O(n)$ space.*

Lower bounds. We now show that no sublinear-time algorithm exists for the value or reporting version, so our algorithm is optimal up to a logarithmic factor. If the input polygon is given as an array of vertices in boundary order and each query reveals one input vertex at unit cost, then any algorithm requires $\Omega(n)$ time in the worst case.

▶ **Theorem 12.** *Let P be a simple polygon with n vertices stored in an array in boundary order, and suppose that each query reveals one vertex of the input polygon and has unit cost. Then, any algorithm for $OSP_V^{\mathrm{simp}}(P)$ or $OSP_R^{\mathrm{simp}}(P)$ requires $\Omega(n)$ time in the worst case.*

5 Orthogonal strip partitioning of convex and self-overlapping polygons

We also consider the orthogonal strip partition problem for convex and self-overlapping polygons.

- **Convex polygons.** For a convex polygon P with n vertices given in boundary order, $\mathrm{OSP}_V^{\mathrm{conv}}(P)$ and $\mathrm{OSP}_R^{\mathrm{conv}}(P)$ can be solved in $O(\log n)$ and $O\left(h \log\left(1 + \frac{n}{h}\right)\right)$ time, respectively, where h is the horizontal width of P. Both bounds are optimal in the decision-tree model.

- **Self-overlapping polygons.** For a self-overlapping polygon P with n vertices given together with a triangulation, $\mathrm{OSP}_V^{\mathrm{self}}(P)$ and $\mathrm{OSP}_R^{\mathrm{self}}(P)$ can be solved in $O(n \log n)$ time using $O(n)$ space. This bound is optimal in the algebraic computation-tree model.

6 Concluding remarks

We studied orthogonal strip partitioning for convex, simple and self-overlapping polygons. For convex polygons, we presented optimal algorithms for both value and reporting versions. In particular, for the reporting version, our reporting algorithm is input-sensitive optimal with respect to the horizontal width of P.

For simple and self-overlapping polygons, we presented a dynamic programming approach that runs in $O(n \log n)$ time and uses $O(n)$ space for both the value and reporting versions. The DP recurrences are naturally formulated on the Clarke–Cormack–Burkowski lattice from order theory. For self-overlapping inputs, the running time is optimal in the algebraic computation-tree model; for simple polygons, it is optimal up to a logarithmic factor in the decision-tree model.

The main open problem is to close the logarithmic gap between the upper and lower bounds for simple polygons. Lemma 8 rules out any greedy pruning rule, so in the worst case the algorithm must compute a meet or join operation between antichains of size $\Theta(n)$. However, it seems difficult to construct a simple polygon in which such expensive lattice operations occur repeatedly, whereas this is easy for self-overlapping inputs. This implies that simplicity may limit the number of costly lattice operations. Even if this intuition is correct, a binary-search based implementation still incurs $O(n \log n)$ time. To obtain an $O(n)$-algorithm, one likely needs a linear-scan procedure, similar to that of Boldi and Vigna [7], together with an amortized analysis of the total cost of all lattice operations.

References

1. Mikkel Abrahamsen, Joakim Blikstad, André Nusser, and Hanwen Zhang. Minimum star partitions of simple polygons in polynomial time. In *Proc. 56th Annual ACM Symposium on Theory of Computing (STOC)*, pages 904–910. Association for Computing Machinery, 2024. doi:10.1145/3618260.3649756.
2. Mikkel Abrahamsen and Nichlas Langhoff Rasmussen. Partitioning a polygon into small pieces. In *Proc. 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3562–3589. Society for Industrial and Applied Mathematics, 2025. doi:10.1137/1.9781611978322.118.
3. Mikkel Abrahamsen and Jack Stade. Hardness of packing, covering and partitioning simple polygons with unit squares. In *Proc. 65th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1355–1371. IEEE, 2024. doi:10.1109/FOCS61266.2024.00087.
4. Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.
5. Thøger Bang. A solution of the "plank problem". *Proc. American Mathematical Society*, 2(6):990–993, 1951.
6. Timothy C. Bell, John G. Cleary, and Ian H. Witten. *Text Compression*. Prentice-Hall, 1990.
7. Paolo Boldi and Sebastiano Vigna. Efficient optimally lazy algorithms for minimal-interval semantics. *Theoretical Computer Science*, 648:8–25, 2016. doi:10.1016/J.TCS.2016.07.036.
8. Paolo Boldi and Sebastiano Vigna. On the lattice of antichains of finite intervals. *Order*, 35(1):57–81, 2018. doi:10.1007/S11083-016-9418-8.
9. Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6(3):485–524, 1991. doi:10.1007/BF02574703.

10 Bernard Chazelle and David Dobkin. Decomposing a polygon into its convex parts. In *Proc. 11th Annual ACM Symposium on Theory of Computing (STOC)*, pages 38–48. Association for Computing Machinery, 1979. doi:10.1145/800135.804396.

11 Jaehoon Chung, Kazuo Iwama, Chung-Shou Liao, and Hee-Kap Ahn. Minimum Partition of Polygons Under Width and Cut Constraints. In *36th International Symposium on Algorithms and Computation (ISAAC 2025)*, volume 359 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ISAAC.2025.22.

12 Charles L. A. Clarke, G. V. Cormack, and F. J. Burkowski. An algebra for structured text search and a framework for its implementation. *The Computer Journal*, 38(1):43–56, 1995. doi:10.1093/COMJNL/38.1.43.

13 J. W. S. Hearle. The structural mechanics of fibers. *Journal of Polymer Science Part C: Polymer Symposia*, 20(1):215–251, 1967.

14 Haim Kaplan, László Kozma, Or Zamir, and Uri Zwick. Selection from heaps, row-sorted matrices, and x+y using soft heaps. In *Proc. 2nd Symposium on Simplicity in Algorithms (SOSA)*, pages 5:1–5:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/OASIcs.SOSA.2019.5.

15 J. Mark Keil. Decomposing a polygon into simpler components. *SIAM Journal on Computing*, 14(4):799–817, 1985. doi:10.1137/0214056.

16 Tianhui Lan, Bingze Wang, Junchao Zhang, Hao Wei, and Xu Liu. Utilization of Waste Wind Turbine Blades in Performance Improvement of Asphalt Mixture. *Frontiers in Materials*, 10:1164693, 2023.

17 Jaegun Lee, Hyojeong An, Hwi Kim, and Hee-Kap Ahn. Monotone partitions of simple polygons. In *Proc. 36th International Workshop on Combinatorial Algorithms (IWOCA)*, pages 72–85, 2025. doi:10.1007/978-3-031-98740-3_6.

18 A. Lingas and V. Soltan. Minimum convex partition of a polygon with holes by cuts in given directions. *Theory of Computing Systems*, 31(5):507–538, 1998. doi:10.1007/S002240000101.

19 Robin Ru-Sheng Liu. *Partitioning Rectilinear Polygons into Simpler Components*. PhD thesis, The University of Texas at Dallas, 1985.

20 Arnold Schönhage. On the power of random access machines. In *Proc. International Colloquium on Automata, Languages and Programming (ICALP)*, pages 520–529, 1979. doi:10.1007/3-540-09510-1_42.

21 Peter W. Shor and Christopher J. Van Wyk. Detecting and decomposing self-overlapping curves. *Computational Geometry*, 2(1):31–50, 1992. doi:10.1016/0925-7721(92)90019-O.