# Rectangular Partitions of a Rectilinear Polygon*

Hwi Kim† Jaegun Lee‡ Hee-Kap Ahn§

**Abstract**

We investigate the problem of partitioning a rectilinear polygon $P$ with $n$ vertices and no holes into rectangles using disjoint line segments drawn inside $P$ under two optimality criteria. In the minimum ink partition, the total length of the line segments drawn inside $P$ is minimized. We present an $O(n^3)$-time algorithm using $O(n^2)$ space that returns a minimum ink partition of $P$. In the thick partition, the minimum side length over all resulting rectangles is maximized. We present an $O(n^3 \log^2 n)$-time algorithm using $O(n^3)$ space that returns a thick partition using line segments incident to vertices of $P$, and an $O(n^6 \log^2 n)$-time algorithm using $O(n^6)$ space that returns a thick partition using line segments incident to the boundary of $P$. We also show that if the input rectilinear polygon has holes, the corresponding decision problem for the thick partition problem using line segments incident to vertices of the polygon is NP-complete. We also present an $O(m^3)$-time 3-approximation algorithm for the minimum ink partition for a rectangle containing $m$ point holes.

## 1 Introduction

Partitioning geometric objects into disjoint parts of certain simple shapes (such as triangles, quadrilaterals, convex polygons, or their higher-dimensional analogues) is a fundamental problem, frequently arising in the analyses of geometric and combinatorial properties of geometric objects. A classic and typical example is the triangulation of a simple polygon in the plane [1, 2]. Geometric structures such as the Voronoi diagram [29, 30] and the Delaunay triangulation [5] partition the underlying space into regions based on proximity. There are various applications in chip manufacturing [19], geoinformatics [23], and pattern recognition [1, 27].

The problem of partitioning a *rectilinear* polygon into *rectangles* has attained attention in computational geometry in the last decades, due to its real-world applications, including VLSI layout design [20, 25, 26] and image processing [8, 13, 22].

In this paper, we study the problem of partitioning an axis-aligned rectilinear polygon $P$ into axis-aligned rectangles using disjoint open line segments under two optimality criteria. In the *minimum ink partition* (i-partition, in short), we obtain a partition of $P$ into rectangles such that the total length of the line segments used in the partition is the minimum among all partitions of $P$ into rectangles. Since the total length of the line segments is exactly half the sum of the

*This research were supported by the Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. 2017-0-00905, Software Star Lab (Optimal Data Structure and Algorithmic Applications in Dynamic Geometric Environment)) and (No. 2019-0-01906, Artificial Intelligence Graduate School Program(POSTECH)).

†Department of Computer Science and Engineering, Pohang University of Science and Technology, Pohang, Korea. hwikim@postech.ac.kr

‡Department of Convergence IT Engineering, Pohang University of Science and Technology, Pohang, Korea. jagunlee@postech.ac.kr

§Department of Computer Science and Engineering, Graduate School of Artificial Intelligence, Pohang University of Science and Technology, Pohang, Korea. heekap@postech.ac.kr

perimeters of the resulting rectangles minus half the perimeter of $P$, this partition minimizes the total perimeter of the resulting rectangles. In the *thick partition* (t-partition, in short), we obtain a partition of $P$ into rectangles such that the minimum side length of the rectangles in the partition is the maximum among all partitions of $P$ into rectangles. If there are two or more such partitions, we break ties among them by favoring one with the fewest number of rectangles.

Both problems have applications in VLSI layout design. A typical problem is to group VLSI circuits into several rectangle-shaped channels such that the channel-to-channel interaction is minimized. The amount of channel-to-channel interaction is known to be proportional to the total length of the sides incident to two different channels [18]. Another problem arises in etching VLSI masks by electron beams with a fixed minimum width. Then the mask is required to be partitioned into rectangles with side lengths at least the minimum width to avoid unnecessary overexposure [24].

### 1.1 Previous works

Lingas et al. [18] are perhaps the first who proposed the i-partition problem. They gave a sketch of an $O(n^4)$-time algorithm using dynamic programming for rectilinear polygons with $n$ vertices and no holes in the plane. Their algorithm has some flaws and does not work correctly for certain polygons, which can be fixed by handling missing cases without increasing the time complexity. For a rectilinear polygon containing holes, they showed that the corresponding decision problem for the i-partition problem is strongly NP-complete. From then on, approximation algorithms have been presented [9, 15, 17].

For the special case of partitioning a *rectangle* containing $m$ point holes into rectangles that contain no holes in their interiors, there are approximation algorithms using divide-and-conquer [10, 15], transformation [11], and dynamic programming [7]. There is a polynomial-time approximation scheme (PTAS) for this problem [21].

The t-partition problem was studied by O'Rourke and Tewari [24]. They conjectured that the problem is NP-hard if holes are allowed, and claimed an $O(n^{42})$-time algorithm for rectilinear polygons with $n$ vertices without holes. When the line segments of a partition are restricted to be incident to polygon vertices (*vertex incidence*), they gave an $O(n^5)$-time and $O(n^4)$-space algorithm, by using observations similar to the ones by Lingas et al. [18]. When each line segment of a partition is restricted to be incident to the boundary of the input polygon (*boundary incidence*), their algorithm under the vertex incidence can be used, with some modifications, for the problem with $O(n^{10})$ running time.

### 1.2 Our results

For a rectilinear polygon $P$ with $n$ vertices and no holes in the plane, we present dynamic programming algorithms improving upon the $O(n^4)$-time algorithm by Lingas et al. for the i-partition problem and the $O(n^5)$-time algorithm by O'Rourke and Tewari for the t-partition problem. Our i-partition algorithm takes $O(n^3)$ time and uses $O(n^2)$ space. This algorithm can be extended to a 3-approximation algorithm with $O(m^3)$ time for partitioning a rectangle containing $m$ point holes into rectangles containing no holes in their interiors [10]. Our t-partition algorithm takes $O(n^3 \log^2 n)$ time and $O(n^3)$ space under the vertex incidence, and $O(n^6 \log^2 n)$ time and $O(n^6)$ space under the boundary incidence. Finally, we show that if the input rectilinear polygon has holes, the corresponding decision problem of the t-partition problem under the vertex incidence is NP-complete.

**Sketches of our algorithms.** Our algorithms are based on the work by Lingas et al. [18]. Their algorithm uses *cutsets* (to be defined later), each consisting of disjoint open axis-aligned line

segments, called *cuts*, that induce a partition of $P$ into rectilinear subpolygons. For a rectangle $R$ contained in a rectilinear polygon $Q$, their algorithm defines a *uni-rectangle partition* of $Q$ for $R$ to be a partition of $Q$ by a cutset $\mathsf{L}$ into $R$ and other rectilinear subpolygons such that each cut $\ell \in \mathsf{L}$ overlaps a side of $R$. The algorithm separates rectangles one by one recursively from a rectilinear subpolygon $Q$ of $P$ by a uni-rectangle partition of $Q$. It maintains an invariant, *the 2-cut property,* that each subpolygon, except the rectangle, obtained from a uni-rectangle partition of every subpolygon $Q$ of $P$ has exactly one boundary chain consisting of at most two line segments contained in the interior of $P$.

Our i-partition algorithm also maintains the 2-cut property, but handles those subpolygons efficiently by classifying them into four types and enumerating uni-rectangle partitions without duplicates while guaranteeing an i-partition. By the 2-cut property, there are $O(n^2)$ subpolygons to consider for a rectilinear polygon with $n$ vertices in our algorithm. To obtain an i-partition, our algorithm considers a number of uni-rectangle partitions and chooses the one with the minimum total length of the cuts among the partitions. From the type classification and tight analyses, we show that there are $O(n^3)$ uni-rectangle partitions to consider in total. The length of the cuts in each uni-rectangle partition can be computed in $O(1)$ time, by maintaining some relevant information. Our algorithm uses $O(1)$ space for partitioning a subpolygon, which corresponds to a subproblem in our algorithm. It also maintains $O(n)$ arrays of length $O(n)$ to partition subpolygons of a certain type efficiently. Thus, our algorithm returns an i-partition of $P$ in $O(n^3)$ time using $O(n^2)$ space.

For a rectangle containing $m$ point holes, Gonzalez and Zheng [11] gave an $O(m^4)$-time 3-approximation algorithm that transforms the rectangle into a *weakly-simple polygon*[1] with $O(m)$ vertices and no point holes in $O(m^2)$ time. Then, it computes an i-partition of the polygon in $O(m^4)$ time. By replacing the i-partition algorithm with our $O(m^3)$-time algorithm, we can get a 3-approximation algorithm with $O(m^3)$ time for the problem.

We give a t-partition algorithm for $P$ under the vertex incidence, by modifying the t-partition algorithm by O'Rourke and Tewari [24]. Our algorithm handles subpolygons efficiently by classifying them and enumerating uni-rectangle partitions without duplicates, while guaranteeing a t-partition under the vertex incidence. It uses certain coherence among uni-rectangle partitions and compares them in $O(n^3 \log^2 n)$ time and $O(n^3)$ space in total. This guarantees the desired time and space complexities of our t-partition algorithm under the vertex incidence.

Our t-partition algorithm under the vertex incidence can be used to compute a t-partition under the boundary incidence with some modifications. Under the boundary incidence, there is an optimal cutset consisting of cuts, each lying on one of $O(n^2)$ horizontal and vertical lines, defined by pairs of vertices of $P$. Thus, our algorithm takes $O(n^6 \log^2 n)$ time and $O(n^6)$ space to compute a t-partition under the boundary incidence.

Finally, for a rectilinear polygon $P$ with holes, we consider the decision problem $\mathsf{TH}(P, \delta, k)$ for a positive real value $\delta$ and a positive integer $k$, determining whether there exists a rectangular partition $\mathsf{P}$ consisting of at most $k$ rectangles with side lengths at least $\delta$ under the vertex incidence. We show that $\mathsf{TH}$ is NP-hard, using a polynomial-time reduction from the planar 3-satisfiability (P3SAT) problem [16], which is known to be NP-complete.

This paper is organized as follows. Section 2 provides terms, definitions, notations and a base lemma that are used throughout the paper. Section 3 gives a review on the previous algorithms for the i-partition problem and the t-partition problem. We present our i-partition algorithm in Section 4, and our t-partition algorithm in Section 5. We conclude this paper with a few open problems in Section 6.

[1] A polygon is weakly-simple if for every $\epsilon > 0$, its vertices can be perturbed by at most $\epsilon$ to obtain a simple polygon.

## 2 Preliminaries

We denote by $P$ the input rectilinear polygon with $n$ vertices and no holes in the plane. We assume that $P$ is axis-aligned and it is given as a sequence of vertices in counterclockwise order along its boundary. For a compact set $X$, we use $\mathsf{int}(X)$ and $\partial X$ to denote the interior and the boundary of $X$, respectively. For any two points $p$ and $q$ in the plane, we denote by $pq$ the line segment connecting them, and by $R_{pq}$ the axis-aligned rectangle with opposite corners $p$ and $q$. We use $x(p)$ and $y(p)$ to denote the $x$-coordinate and the $y$-coordinate of a point $p$, respectively. The grid induced by the lines, each extended from an edge of $P$, is called the *canonical grid* of $P$ and it is denoted by $\mathsf{G}$.

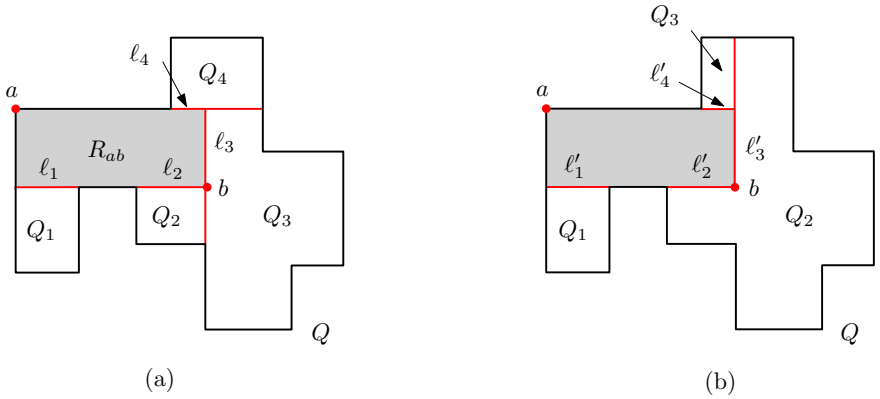



Figure 1: (a) Partition $\mathsf{Q} = \{R_{ab}, Q_1, Q_2, Q_3, Q_4\}$ of $Q$ and its cutset $\mathsf{L} = \{\ell_1, \ell_2, \ell_3, \ell_4\}$. $\mathsf{L}$ determines $R_{ab}$ from $Q$. (b) Uni-rectangle partition $\mathsf{U} = (Q, R_{ab}, \mathsf{L}')$ with $\mathsf{L}' = \{\ell'_1, \ell'_2, \ell'_3, \ell'_4\}$. $\mathsf{L}'$ (and thus $\mathsf{U}$) determines $R_{ab}$ from $Q$.

**Cutsets.** A *partition* of a rectilinear polygon is a set of interior-disjoint rectilinear subpolygons induced by a *cutset* consisting of open axis-aligned line segments, called cuts, contained in the polygon such that the union of the subpolygons equals the polygon. For any partition $\mathsf{P}$ of a rectilinear polygon, we denote by $\mathsf{card}(\mathsf{P})$ the number of subpolygons in $\mathsf{P}$. We require the cuts in a cutset to be *disjoint* and *maximal* with respect to the corresponding partition. For each partition of a rectilinear polygon $Q$ in Figure 1, the cuts are disjoint and maximal with respect to the partition.

For a subpolygon $Q'$ in a partition $\mathsf{Q}$ of a rectilinear polygon $Q$, we say the cutset of $\mathsf{Q}$ *determines* $Q'$ from $Q$ if for every cut $\ell$ in the cutset of $\mathsf{Q}$, $\ell \cap \partial Q'$ is a (nondegenerate) line segment. In other words, every cut of the cutset contributes to the boundary of $Q'$. We also say $\mathsf{Q}$ determines $Q'$ from $Q$ if the cutset of $\mathsf{Q}$ determines $Q'$ from $Q$. Figure 1 shows two cutsets that determine rectangle $R_{ab}$ from $Q$.

**Uni-rectangle partitions.** A *rectangular partition* of a rectilinear polygon is a partition of the polygon into rectangles. A *uni-rectangle partition* of a rectilinear polygon $Q$ is a partition of $Q$ whose cutset determines a rectangle from $Q$. Thus, for two points $a, b \in Q$ such that $R_{ab}$ is contained in $Q$, and a uni-rectangle partition with cutset $\mathsf{L}$ determining $R_{ab}$ from $Q$, we use $(Q, R_{ab}, \mathsf{L})$ to denote the uni-rectangle partition of $Q$ for $R_{ab}$ induced by $\mathsf{L}$. Observe that there can be more than one uni-rectangle partition whose cutset determines $R_{ab}$ from $Q$ as there can be more than one such cutset. See Figure 1 for two different uni-rectangle partitions determining $R_{ab}$ from $Q$.

We use the following notations for the amounts of ink used in a rectangular partition, in an i-partition, and in a uni-rectangle partition of a polygon. We denote by $\mathsf{len}(\ell)$ the length of a line segment $\ell$.

- $\mathsf{Ink}(\mathsf{R}) := \sum_{\ell \in \mathsf{L}} \mathsf{len}(\ell)$, for a rectangular partition $\mathsf{R}$ and its cutset $\mathsf{L}$.
- $\mathsf{Ink}(Q)$ is the minimum of $\mathsf{Ink}(\mathsf{R})$ over all rectangular partitions $\mathsf{R}$ of a rectilinear polygon $Q$.
- $\mathsf{Ink}(Q, R_{ab}, \mathsf{L}) := \sum_{\ell \in \mathsf{L}} \mathsf{len}(\ell) + \sum_{Q' \in \mathsf{U}} \mathsf{Ink}(Q')$, for a uni-rectangle partition $\mathsf{U} = (Q, R_{ab}, \mathsf{L})$ of a rectilinear polygon $Q$.

For a rectilinear polygon $Q$, a rectangular partition $\mathsf{R}$ realizing $\mathsf{Ink}(Q) = \mathsf{Ink}(\mathsf{R})$ is called an i-partition of $Q$. A uni-rectangle partition $\mathsf{U} = (Q, R_{ab}, \mathsf{L})$ of $Q$ is *optimal* if there is an i-partition $\mathsf{R}$ of $Q$ consisting of $R_{ab}$ and the rectangles from some rectangular partitions of the subpolygons of $\mathsf{U}$. In this case, $\mathsf{R}$ is an *optimal refinement* of $\mathsf{U}$. Lemma 1 comes from the fact that each vertex of a rectilinear polygon is used as a corner of a rectangle in every rectangular partition of the polygon.

**Lemma 1.** *For a rectilinear polygon $Q$, $\mathsf{Ink}(Q) = \min_b \min_{\mathsf{L}} \mathsf{Ink}(Q, R_{ab}, \mathsf{L})$ for any fixed vertex $a$ of $Q$ and any point $b \in Q$.*

*Proof.* Since any optimal refinement of a uni-rectangle partition is a rectangular partition of $P$, we have $\mathsf{Ink}(Q) \leq \min_b \min_{\mathsf{L}} \mathsf{Ink}(Q, R_{ab}, \mathsf{L})$.

Assume to the contrary that $\mathsf{Ink}(Q) < \min_b \min_{\mathsf{L}} \mathsf{Ink}(Q, R_{ab}, \mathsf{L})$. Then there is no i-partition $\mathsf{R}$ of $Q$ such that $a$ is a corner of a rectangle in $\mathsf{R}$. If $a$ is a convex vertex of $Q$, there is a rectangle with a corner on $a$ in every i-partition, a contradiction. So $a$ must be a reflex vertex of $Q$. By the assumption, $a$ lies in the interior of a side of a rectangle $R$ in some i-partition $\mathsf{R}'$ of $Q$. Then there is a subpolygon in $Q \setminus R$ such that $a$ is a convex vertex of the subpolygon. This implies that there is a rectangle with a corner at $a$ in every rectangular partition of the subpolygon, and there is a rectangle in $\mathsf{R}'$ with a corner at $a$, a contradiction. ◻

**Vertex cuts and anchored cuts.** Cuts drawn inside a rectilinear polygon $Q$ can be classified into two types. A cut is a vertex cut (v-cut, in short) if it is incident to a reflex vertex of $Q$. A cut is an anchored cut (a-cut, in short) if it is incident to the boundary of $Q$. Note that every v-cut is an a-cut. A partition is said to be under the *vertex incidence* if every cut of the partition is a v-cut, and under the *boundary incidence* if every cut of the partition is an a-cut. Figure 1(a) shows a cutset consisting of v-cuts, and thus the partition is under the vertex incidence. In Figure 1(b), $\ell'_3$ is an a-cut, but not a v-cut. Thus, the partition is under the boundary incidence, but not under the vertex incidence.

A subpolygon $Q$ of $P$ is a *$k$-cut subpolygon* of $P$ if the boundary of $Q$ consists of a contiguous boundary portion of $P$ and a chain consisting of $k$ line segments contained in $\mathsf{int}(P)$, alternating horizontal and vertical. The $k$ line segments constitute the cutset that partitions $P$ into $Q$ and $P \setminus Q$. We call each such segment a *boundary cut* of $Q$. Then $P$ itself is the 0-cut subpolygon of $P$. A subpolygon is a $\leq k$-cut subpolygon if it is a $k'$-cut subpolygon for some $k' \leq k$.

## 3 Previous algorithms for rectangular partitions

We give a sketch of the algorithm by Lingas et al. [18] for the i-partition problem and a sketch of the algorithm by O'Rourke and Tewari [24] for the t-partition problem. Both algorithms employ a dynamic programming approach, and the latter one is based on the former one.

### 3.1 i-partition algorithm by Lingas et al.

Lingas et al. observed that there is an i-partition of $P$ by a cutset $\mathsf{L}$ consisting of a-cuts lying on grid lines of the canonical grid $\mathsf{G}$, and gave an algorithm returning $\mathsf{L}$. The rectangles of the partition induced by $\mathsf{L}$ have corners all at grid points of $\mathsf{G}$.

Their algorithm works recursively as follows. Let $Q$ denote the input subpolygon for the recursive algorithm. Initially, $Q = P$. The algorithm finds a uni-rectangle partition for a rectangle $R \subset Q$ such that $R$ has its corners at grid points and it is incident to a boundary cut of $Q$. In doing so, it fixes a vertex of $Q$ as the *origin point* $o$, and searches for every grid point $p$ (called a *partner point* of $o$) of $\mathsf{G}$ such that $R_{op} \subset Q$ and each subpolygon in a uni-rectangle partition $(Q, R_{op}, \mathsf{L})$, except $R_{op}$, is either a 1-cut or 2-cut subpolygon of $P$. This is called the *2-cut property*. It finds a uni-rectangle partition $\mathsf{U}^* = (Q, R_{op^*}, \mathsf{L}^*)$ satisfying $\mathsf{Ink}(Q, R_{op^*}, \mathsf{L}^*) = \min_p \min_{\mathsf{L}} \mathsf{Ink}(Q, R_{op}, \mathsf{L})$. Thus, $\mathsf{Ink}(Q) = \mathsf{Ink}(Q, R_{op^*}, \mathsf{L}^*)$. Finally, it computes an optimal refinement $\mathsf{R}_{\mathsf{U}^*}$ by computing an i-partition of each subpolygon in $\mathsf{U}^* = (Q, R_{op^*}, \mathsf{L}^*)$ recursively. By Lemma 1, $\mathsf{R}_{\mathsf{U}^*}$ is an i-partition of $Q$. Their dynamic programming algorithm uses $\mathsf{Ink}(Q')$ value stored in a table for each subpolygon $Q' \in (Q, R_{op}, \mathsf{L})$. Figure 2 illustrates how the algorithm works.

The origin point $o$ of $Q$ is any convex vertex of $P$ for $Q = P$ (Figure 2(a)), the endpoint shared by the two boundary cuts for a 2-cut subpolygon $Q$ (Figure 2(b)), and an endpoint of the edge containing the boundary cut for a 1-cut subpolygon $Q$ (Figure 2(c)).

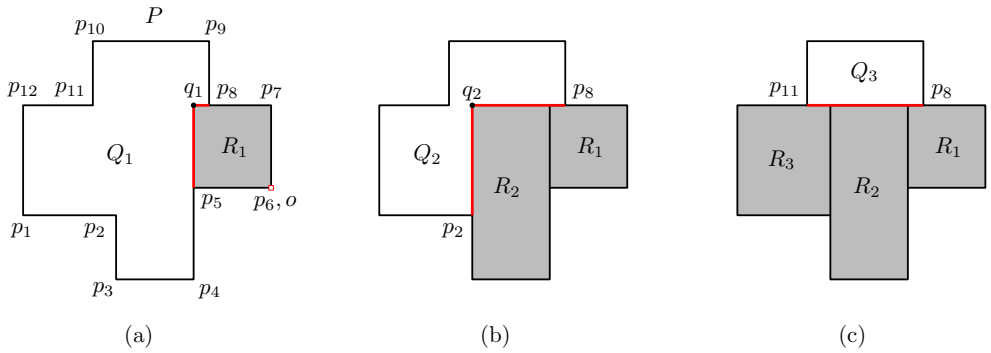



Figure 2: The i-partition algorithm by Lingas et al. in a series of recursive steps. During the algorithm, we encounter 2-cut subpolygons $Q_1$ (by cuts $q_1p_8, q_1p_5$) in (a) and $Q_2$ (by cuts $q_2p_8, q_2p_2$) in (b), and a 1-cut subpolygon (rectangle) $Q_3$ (by cut $p_{11}p_8$) in (c). The resulting i-partition is $\{R_1, R_2, R_3, Q_3\}$ with cutset $\{p_{11}p_8, q_1p_5, q_2p_2\}$.

Lingas et al. claimed that for a fixed origin point $o$, it suffices to check only certain partner points with respect to $R_{o\kappa}$ to compute an i-partition while maintaining the 2-cut property, where $\kappa$ is the *kitty corner* of $o$. The kitty corner $\kappa$ of the origin point $o$ for a 2-cut subpolygon $Q$ is a grid point of $\mathsf{G}$ such that $ot$ and $ot'$ are edges of $Q$ for corners $o, t, \kappa, t'$ of $R_{o\kappa}$. Note that $\kappa$ is not necessarily a partner point of $o$, and it may lie outside of $Q$. See Figure 3.

Every 1-cut subpolygon is of type 1. A 2-cut subpolygon $Q$ is classified into two subtypes: the endpoint shared by the two cuts is either convex (type 2C) or reflex (type 2R) with respect to $Q$. See Figure 4. The following two rules summarize their claim for a 2-cut subpolygon $Q$ with origin point $o$, kitty corner $\kappa$, and a partner point $p$ of $o$.

- For $Q$ of type 2C, $R_{op}$ appears in an i-partition of $Q$ only if $p \notin \mathsf{int}(R_{o\kappa})$. Every subpolygon in uni-rectangle partition $(Q, R_{op}, \mathsf{L})$ is a $\leq$2-cut subpolygon, where $\mathsf{L}$ is any cutset

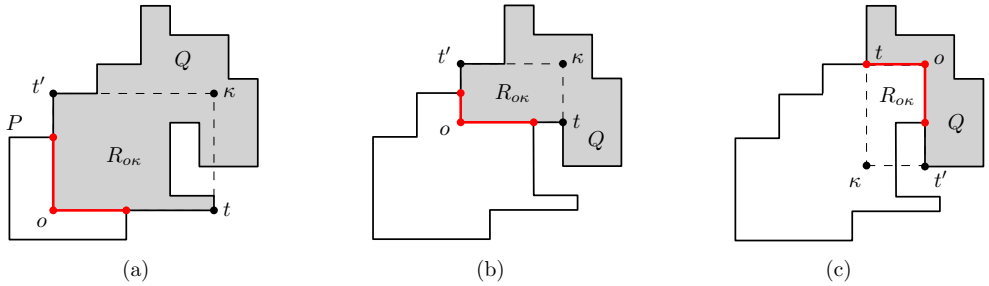



Figure 3: Polygon $P$ and 2-cut subpolygons $Q$ (gray). For a kitty corner $\kappa$ of $o$, (a) $\kappa \in Q$, but $R_{o\kappa} \not\subseteq Q$, (b) $\kappa \in Q$, and $R_{o\kappa} \subseteq Q$. (c) $\kappa \notin Q$, and thus $R_{o\kappa} \not\subseteq Q$.

consisting of a-cuts and determining $R_{op}$.

- For $Q$ of type 2R, $R_{op}$ appears in an i-partition of $Q$ only if either the vertical line or the horizontal line through $p$ intersects $R_{o\kappa}$. If so, every subpolygon in uni-rectangle partition $(Q, R_{op}, \mathsf{L})$ is a $\leq$2-cut subpolygon, where L is any cutset consisting of a-cuts and determining $R_{op}$.

Any partner point satisfying one of the two rules above is a *candidate partner point* in their algorithm. However, there are some flaws in the rules, which can be fixed easily. See Section 4 for details. They claimed that any 1-cut subpolygon can be handled in the same way for a 2-cut subpolygon by considering a subpolygon edge, incident and perpendicular to the boundary cut, as an additional boundary cut. However, as O'Rourke and Tewari [24] pointed out, this is not always the case. Again, this can be corrected by choosing the origin and partner points in a certain way. See Section 4 for details.

Their algorithm compares lnk(U) values of uni-rectangle partitions U with $R_{op}$ satisfying the rules above, and takes the uni-rectangle partition with the minimum lnk(U) value. It returns an i-partition of $P$ in $O(n^4)$ time using $O(n^2)$ space, because there are $O(n^2)$ $\leq$2-cut subpolygons, and $O(n^2)$ candidate partner points for each subpolygon.

### 3.2 t-partition algorithm by O'Rourke and Tewari

We use *vt-partition* to refer to a t-partition whose cutset consists of v-cuts. We give a sketch of the algorithm by O'Rourke and Tewari [24], denoted by vt-algo, that computes a vt-partition.

vt-algo makes use of observations similar to the ones in the i-partition algorithm by Lingas et al. [18] under the vertex incidence. In the following, we use terms in Section 3.1 such as the

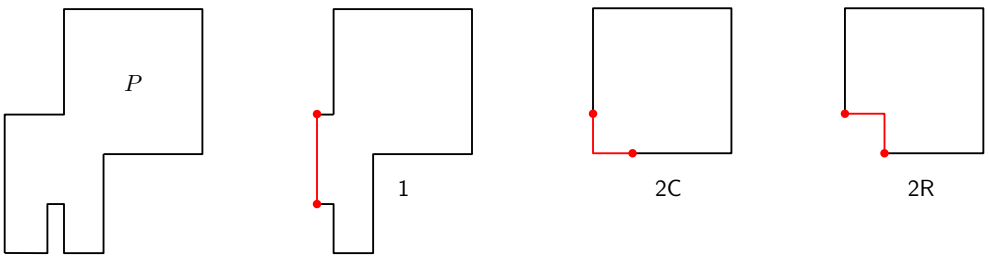



Figure 4: Three different types of $\leq$2-cut subpolygons of $P$, and their boundary cuts (red).

2-cut property, origin points, and partner points. The algorithm maintains the 2-cut property and handles subpolygons belonging to each of the three types (1, 2C, 2R) separately, as the i-partition algorithm by Lingas et al. does. For subpolygons of types 2C and 2R, it uses a few rules for determining origin points and their partner points in order to compute a vt-partition. A subpolygon of type 1 is considered as a subpolygon of type 2C or 2R.

However, there are degenerate 1-cut subpolygons $Q$ such that the two edges of $P$ connected by the boundary cut $\ell$ of $Q$ are collinear to $\ell$. To handle such a degenerate 1-cut subpolygon, vt-algo takes each of the $O(n)$ grid points of G lying on the grid line through its boundary cut $\ell$ and contained in $Q$ as an origin point. For each origin point $o$, it finds partner points $p$ such that $\ell$ intersects the boundary of $R_{op}$ in a (nondegenerate) line segment. Observe that every subpolygon in the uni-rectangle partition is again a $\leq$2-cut subpolygon. See Figure 5. Hence, vt-algo obtains a vt-partition of each 1-cut subpolygon by checking all such origin and partner point pairs.

Based on the type classification and analyses, together with dynamic programming, vt-algo computes a vt-partition of $P$ in $O(n^5)$ time and $O(n^4)$ space.

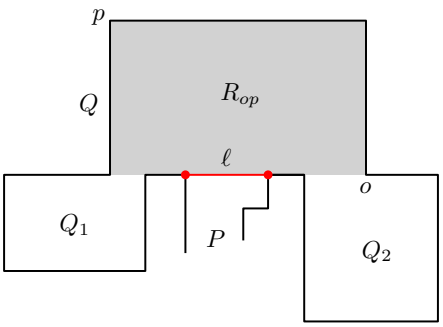



Figure 5: A degenerate 1-cut subpolygon $Q$ whose boundary cut $\ell$ connects two edges of $P$ collinear to $\ell$. Since $\ell$ intersects the boundary of $R_{op}$ in a line segment, $p$ is considered as a partner point of $o$ by vt-algo. The resulting partition $\{R_{op}, Q_1, Q_2\}$ is the vt-partition of $Q$.

## 4 i-partition algorithm

We present an efficient algorithm that computes an i-partition of a rectilinear polygon $P$ with $n$ vertices and no holes in the plane. Our algorithm is based on the work by Lingas et al. [18], but with careful analyses on the subproblems and by exploiting certain geometric and combinatorial coherence among them, we could improve upon their algorithm.

We classify subpolygons into four types (1C, 1R, 2C, 2R) and enumerate uni-rectangle partitions of each type without duplicates, while guaranteeing an i-partition. We show that there are $O(n^3)$ uni-rectangle partitions of types 1C, 1R, and 2C, and $O(n^4)$ uni-rectangle partitions of type 2R, to consider for an i-partition. We use certain coherence among uni-rectangle partitions of type 2R so that it suffices to check only $O(n^3)$ of them. We can compute lnk$(\cdot)$ value for each uni-rectangle partition in $O(1)$ time, after $O(n^2)$ time preprocessing (details in Section 4.2), so our algorithm runs in $O(n^3)$ time.

We show that our algorithm uses $O(n^2)$ space. The preprocessing step for lnk$(\cdot)$ queries and the query data structure maintained in the algorithm requires $O(n^2)$ space. The algorithm also maintains $O(n)$ arrays of length $O(n)$ to efficiently compute lnk$(\cdot)$ values of uni-rectangle partitions of type 2R. Thus, our algorithm uses $O(n^2)$ space in total.

Our algorithm is based on the following lemma that guarantees an i-partition by a cutset consisting of v-cuts. This is stronger than the observation by Lingas et al. using a-cuts in Section 3.1.

**Lemma 2** (Lemma 5.1 of [6]). *There is an i-partition of $P$ by a cutset consisting of v-cuts.*

By Lemma 2, our algorithm considers cutsets consisting of v-cuts only. It maintains the 2-cut property and uses the type classification of 2-cut subpolygons into 2C and 2R in Section 3.1. We classify 1-cut subpolygons $Q$ further into 1R (degenerate 1-cut subpolygons mentioned in Section 3.2) and 1C (1-cut subpolygons not belonging to 1R).

Throughout this section, we denote by $Q$ a $\leq$2-cut subpolygon. For any 2-cut subpolygon $Q$, we use $\ell_h$ (horizontal) and $\ell_v$ (vertical) to denote its two boundary cuts.

**Choosing origin points.** If $Q$ is of type 2C or 2R, the endpoint shared by two cuts is the origin point $o$. We use $\kappa$ to denote the kitty corner of $o$. If $Q$ is of type 1C, an endpoint of its boundary cut lying on a convex vertex of $Q$ is the origin point $o$. If $Q$ is of type 1R, the grid points contained in $Q$ and lying on the grid line through the boundary cut of $Q$ are the origin points. Thus, there are $O(n)$ origin points to consider for $Q$ of type 1R.

**Optimal uni-rectangle partitions.** In the following lemma, we characterize the uni-rectangle partitions that appear in an i-partition with respect to the subpolygon type (2C, 2R, and 1R), and give rules for choosing partner points based on the characterization. Each subpolygon $Q$ of type 1C can be considered as a subpolygon of type 2C, by using an edge of $Q$ incident and perpendicular to its boundary cut, as an additional boundary cut. Then, rule 1 in the following lemma can be applied to the induced subpolygon of type 2C.

**Lemma 3.** *Given a $\leq$2-cut subpolygon $Q$ of $P$ and its origin point $o$, we can compute an optimal uni-rectangle partition of $Q$ while maintaining the 2-cut property, by checking partner points $p$ satisfying one of the following three rules.*

1. *For $Q$ of type 2C or 1C, (1) a side of $R_{op}$ incident to $p$ intersects $\partial Q$ in a (nondegenerate) line segment or (2) $p \notin \mathsf{int}(R_{o\kappa})$.*
2. *For $Q$ of type 2R, (1) a side of $R_{op}$ incident to $p$ intersects $\partial Q$ in a (nondegenerate) line segment or (2) $p$ lies in a quadrant (defined by the horizontal line and the vertical line through $o$) not containing $\kappa$ and not opposite to the quadrant containing $\kappa$.*
3. *For $Q$ of type 1R, the boundary cut of $Q$ intersects the boundary of $R_{op}$ in a (nondegenerate) line segment.*

*Proof.* Lingas et al. [18] claim (in *Lemma 3* of the paper) that for a subpolygon $Q$ of type 2C and its origin point $o$, $p \notin \mathsf{int}(R_{o\kappa})$ for every partner point $p$. The claim holds only when no side of $R_{op}$ incident to $p$ intersects $\partial Q$ in a nondegenerate line segment. Thus, we have rule 1. They also claim (in *Lemma 4* of the paper) that for each subpolygon $Q$ of type 2R and its origin point $o$, either the vertical line or the horizontal line through $p$ intersects $R_{o\kappa}$, for the kitty corner $\kappa$ of $o$. Thus we can rule out the partner points $p$ in the quadrant (defined by the horizontal line and the vertical line through $o$) opposite to the quadrant containing $\kappa$ such that both sides of $R_{op}$ incident to $p$ do not intersect $\partial Q$ in a nondegenerate line segment. Finally, we can show that the iteration over the partner points satisfying rule 3 can be done by checking all uni-rectangle partitions of each subpolygon of type 1R without duplicates, using an argument similar to the proof of Lemma 1. □

### 4.1 Counting distinct $(Q, o, p)$ triplets

Our algorithm computes $\mathsf{Ink}(\cdot)$ values for the uni-rectangle partitions satisfying the 2-cut property and Lemma 3. It takes time linear to the number of such uni-rectangle partitions because we can compute $\mathsf{Ink}(\cdot)$ value for each uni-rectangle partition in $O(1)$ time, after preprocessing.

To bound the number of such uni-rectangle partitions, it suffices to count the distinct $(Q, o, p)$ triplets (called *candidate triplets*) in the uni-rectangle partitions, because for a triplet $(Q, o, p)$, there are $O(1)$ distinct cutsets $\mathsf{L}$ consisting of $\mathsf{v}$-cuts for uni-rectangle partitions $(Q, R_{op}, \mathsf{L})$. We will show this in Section 4.2.

We classify candidate triplets $(Q, o, p)$ into types by the types of $Q$, and count the distinct candidate triplets in the following order of types, $\mathsf{2C}$, $\mathsf{1C}$, $\mathsf{1R}$, $\mathsf{2R}$, for ease of description. For any type $\mathsf{A} \in \{\mathsf{2C}, \mathsf{1C}, \mathsf{1R}, \mathsf{2R}\}$, $\mathsf{T_A}$ denotes the number of distinct candidate triplets of type $\mathsf{A}$.

For a 2-cut subpolygon $Q$, the *cut direction* of $Q$ is *up-right* if one boundary cut lies above $o$ and the other boundary cut lies to the right of $o$. The *vertex type* of $R_{o\kappa}$ for $Q$ with cut direction up-right is *convex-reflex* if the bottom-right corner of $R_{o\kappa}$ is a convex vertex of $Q$, and the top-left corner of $R_{o\kappa}$ is a reflex vertex of $Q$. Because $o$ and $\kappa$ are fixed for $Q$, we say that the vertex type of $Q$ is convex-reflex if the vertex type of $R_{o\kappa}$ is convex-reflex.

#### 4.1.1 Candidate triplets of type 2C.

We count the candidate triplets for $Q$ of type $\mathsf{2C}$. We first show that there are $O(n^3)$ candidate triplets for $Q$ satisfying condition (1) of rule 1 in Lemma 3. Then, we consider the candidate triplets for $Q$ satisfying (2) but not satisfying (1) of rule 1 in Lemma 3, classify them into three subtypes, and then bound the number of candidate triplets of each subtype to $O(n^3)$.

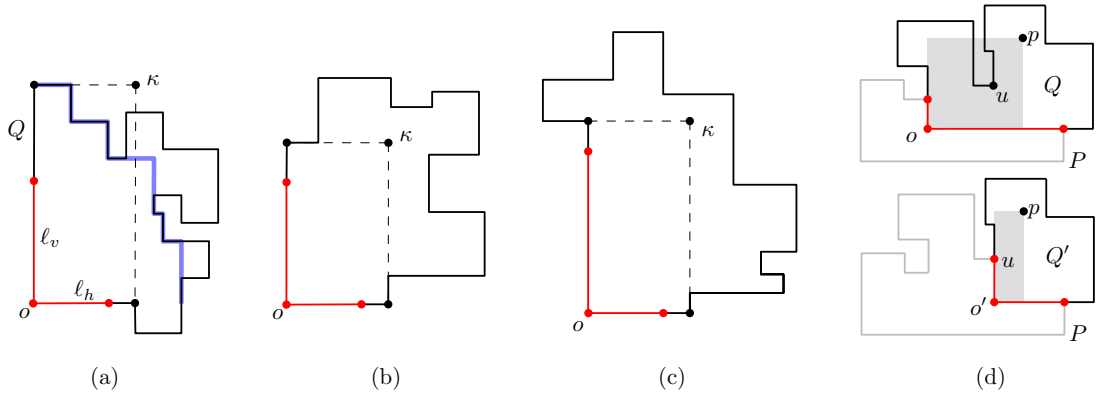
We first count the candidate triplets for $Q$ of type $\mathsf{2C}$ that satisfy condition (1). The grid points $p$ such that $(Q, o, p)$ satisfies condition (1) of rule 1 in Lemma 3 are on a staircase chain of line segments on grid lines of $\mathsf{G}$ which alternates between horizontal and vertical. See Figure 6(a). Since there are $O(n^2)$ subpolygons of type $\mathsf{2C}$ and $O(n)$ grid points of $\mathsf{G}$ lie on such a chain, there are $O(n^3)$ candidate triplets of type $\mathsf{2C}$ satisfying condition (1).

We then count the candidate triplets for $Q$ of type $\mathsf{2C}$ that do not satisfy condition (1) but do satisfy condition (2) of rule 1 in Lemma 3. To do this efficiently, we classify candidate triplets further into three subtypes by the vertex types of the corners $t, t'$ of $R_{o\kappa}$ other than $o$ and $\kappa$. If both $t$ and $t'$ are convex vertices of $Q$, $Q$ is *closed* (subtype $\mathsf{2Cc}$). If both $t$ and $t'$ are reflex vertices of $Q$, $Q$ is *open* (subtype $\mathsf{2Co}$). If one is a convex vertex and the other is a reflex vertex of $Q$, $Q$ is *half-open* (subtype $\mathsf{2Ch}$). We assume that the cut direction of $Q$ is up-right with $\ell_h$ and $\ell_v$.

**When $Q$ is closed ($\mathsf{2Cc}$).** Observe that $R_{op} \subseteq Q$ only if $p \in R_{o\kappa}$. See Figure 6(b). On the other hand, $p \notin \mathsf{int}(R_{o\kappa})$ by condition (2) of rule 1 in Lemma 3. Hence, if $(Q, o, p)$ is a candidate triplet, $p$ lies on the boundary of $R_{o\kappa}$. More precisely, $p$ lies on one of the two sides of $R_{o\kappa}$ incident to $\kappa$, since otherwise $R_{op}$ becomes a line segment. Because each such $p$ satisfies condition (1) of rule 1 in Lemma 3, there is no candidate triplet of type $\mathsf{2Cc}$.

**When $Q$ is half-open ($\mathsf{2Ch}$).** Observe that $R_{op} \subseteq Q$ only if $p$ lies in the intersection of $Q$ and the vertical slab bounded by the two lines, one through the left side of $R_{o\kappa}$ and one through the right side of $R_{o\kappa}$. See Figure 6(c). On the other hand, $p \notin \mathsf{int}(R_{o\kappa})$ by condition (2) of rule 1 in Lemma 3. Hence, if $(Q, o, p)$ is a candidate triplet, $p$ lies in the intersection of $Q \setminus \mathsf{int}(R_{o\kappa})$ and the vertical slab.

We need the following lemma to bound the number of candidate triplets of type $\mathsf{2Ch}$.




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

Figure 6: (a) For $Q$ of type 2C, the grid points $p$ such that $(Q, o, p)$ satisfies condition (1) of rule 1 in Lemma 3 are on a staircase chain of line segments on grid lines, alternating between horizontal and vertical. (b) A subpolygon of type 2Cc. (c) A subpolygon of type 2Ch. (d) Subpolygons $Q$ and $Q'$ of type 2Ch with origin points lying on the same horizontal grid line.

**Lemma 4.** *Let $Q$ and $Q'$ be subpolygons of type* 2Ch *with cut direction up-right and vertex type convex-reflex. If the origin points $o$ of $Q$ and $o'$ of $Q'$ lie on the same horizontal grid line, at most one of $(Q, o, p)$ and $(Q', o', p)$ is a candidate triplet for any fixed grid point $p$.*

*Proof.* Let $Q$ and $Q'$ be such subpolygons of type 2Ch with origin points $o$ and $o'$, respectively, lying on the same horizontal grid line with $x(o) < x(o')$. Since $x(o) < x(o')$, either the horizontal cut of $Q'$ is contained in the horizontal cut of $Q$ (see Figure 6(d)) or there is an edge of $P$ contained in the horizontal line segment with endpoints $o$ and $o'$.

Let $u$ be the upper endpoint of the vertical cut of $Q'$. If both $(Q, o, p)$ and $(Q', o', p)$ are candidate triplets, then $x(o) < x(o') = x(u) < x(p)$ and $y(o) = y(o') < y(u) < y(p)$, because $p$ satisfies condition (2) of rule 1 in Lemma 3. Thus, $u \in \mathsf{int}(R_{op})$ and $R_{op} \not\subset Q$ because $u$ is a reflex vertex of $Q$. This contradicts that $(Q, o, p)$ is a candidate triplet. ◻

There are $O(n^2)$ grid points of G. By Lemma 4, each grid point $p$ induces at most one candidate triplet $(Q, o, p)$ among subpolygons $Q$ of type 2Ch whose cut direction is up-right, whose vertex type is convex-reflex, and whose origin points lie on the same horizontal grid line of G. So there are $O(n^2)$ candidate triplets for such subpolygons of type 2Ch with the same cut direction, the same grid line of G on which origin points lie, and the same vertex type which is determined by the cut direction and the grid line. There are $O(1)$ cut directions and vertex types, and $O(n)$ grid lines of G. Hence, $\mathsf{T}_{\mathsf{2Ch}} = O(n^3)$.

**When $Q$ is open (2Co).** The four regions of the plane subdivided by the vertical line and the horizontal line through $\kappa$ are called the *$\kappa$-quadrants* labeled from $\kappa_{\mathsf{I}}$ (top-right region) to $\kappa_{\mathsf{IV}}$ (bottom-right region), in counterclockwise direction around $\kappa$. Since the cut direction of $Q$ is up-right, $o$ always lies in $\kappa_{\mathsf{III}}$. We consider the partner points of $o$ lying in $\kappa_{\mathsf{I}}, \kappa_{\mathsf{II}}$, and $\kappa_{\mathsf{IV}}$, but no one lying in the interior of $\kappa_{\mathsf{III}}$, by condition (2) of rule 1 in Lemma 3.

We need the following lemma to bound the number of candidate triplets for $Q$ of type 2Co.

**Lemma 5.** *Let $Q$ and $Q'$ be subpolygons of type* 2Co *with up-right cut direction and origin points $o$ and $o'$, respectively. If $o$ and $o'$ lie on the same horizontal grid line of* G*, at most one of $(Q, o, p)$ and $(Q', o', p)$ is a candidate triplet for any fixed grid point $p$ in $(\kappa_{\mathsf{I}} \cup \kappa_{\mathsf{II}}) \cap (\kappa'_{\mathsf{I}} \cup \kappa'_{\mathsf{II}})$, for kitty corners $\kappa$ of $o$ and $\kappa'$ of $o'$.*

*Proof.* Let $u$ be the upper endpoint of the vertical boundary cut of $Q'$. If $(Q', o', p)$ is a candidate triplet for a grid point $p \in \kappa'_{\mathsf{I}} \cup \kappa'_{\mathsf{II}}$ and $(Q, o, p)$ is a candidate triplet for the same grid point $p \in \kappa_{\mathsf{I}} \cup \kappa_{\mathsf{II}}$, then $x(o) < x(o') = x(u) < x(p)$ and $y(o) = y(o') < y(u) < y(p)$, as in the proof of Lemma 4. Thus, $u \in \mathsf{int}(R_{op})$, and $R_{op} \not\subset Q$ because $u$ is a reflex vertex of $Q$. This contradicts that $(Q, o, p)$ is a candidate triplet. ◻

Similarly, we can prove the following corollary.

**Corollary 1.** *Let $Q$ and $Q'$ be subpolygons of type* 2Co *with up-right cut direction. If their origin points $o$ of $Q$ and $o'$ of $Q'$ lie on the same vertical grid line of* G*, then at most one of $(Q, o, p)$ and $(Q', o', p)$ is a candidate triplet for any fixed grid point $p$ in $(\kappa_{\mathsf{I}} \cup \kappa_{\mathsf{IV}}) \cap (\kappa'_{\mathsf{I}} \cup \kappa'_{\mathsf{IV}})$, for kitty corners $\kappa$ of $o$ and $\kappa'$ of $o'$.*

By Lemma 5, each grid point $p$ induces at most one candidate triplet with $p \in \kappa_{\mathsf{I}} \cup \kappa_{\mathsf{II}}$ among the subpolygons $Q$ of type 2Co with origin points lying on the same horizontal grid line. By Corollary 1, each grid point $p$ induces at most one candidate triplet with $p \in \kappa_{\mathsf{I}} \cup \kappa_{\mathsf{IV}}$ among the subpolygons $Q$ of type 2Co with origin points lying on the same vertical grid line. Since there are $O(n^2)$ grid points and $O(n)$ grid lines of G, there are $O(n^3)$ candidate triplets $(Q, o, p)$ of type 2Co such that the cut direction of $Q$ is up-right. There are $O(1)$ cut directions, so $\mathsf{T}_{\mathsf{2Co}} = O(n^3)$.

Thus, we bound the number of candidate triplets of type 2C.

**Lemma 6.** *There are $O(n^3)$ candidate triplets of type* 2C*.*

#### 4.1.2 Candidate triplets of type 1C.

Recall that the boundary cut $\ell$ of $Q$ of type 1C, together with an additional cut (an edge of $Q$ incident to the edge containing $\ell$ and sharing an endpoint with $\ell$) induces a 2C subpolygon. Since we consider an i-partition induced by a cutset consisting of v-cuts, we have the following lemma.

**Lemma 7.** *There are $O(n)$ subpolygons of type* 1C*.*

*Proof.* One endpoint of the boundary cut of $Q$ of type 1C lies on a reflex vertex of $P$. Each reflex vertex of $P$ can be used as an endpoint of the boundary cut for at most two different subpolygons of type 1C. There are $O(n)$ reflex vertices of $P$, and thus there are $O(n)$ subpolygons of type 1C. ◻

For each subpolygon $Q$ of type 1C and its origin point $o$, there are $O(n^2)$ partner points $p$ such that $(Q, o, p)$ satisfies rule 1 in Lemma 3, because there are $O(n^2)$ grid points of G. This, together with Lemma 7, bounds the number of candidate triplets of type 1C.

**Lemma 8.** *There are $O(n^3)$ candidate triplets of type* 1C*.*

#### 4.1.3 Candidate triplets of type 1R.

We characterize each subpolygon of type 1R by its cut direction and the local placement of the subpolygon around the boundary cut. Let $H^+$ be the set of the subpolygons $Q$ of type 1R such that the boundary cut $\ell$ of $Q$ is horizontal and $Q$ lies above $\ell$ locally.

**Lemma 9.** *For $Q$ in $H^+$, $R_{op}$ appears in no i-partition of $Q$ if both top corners of $R_{op}$ have their vertical projections onto $\mathsf{int}(\ell)$.*

*Proof.* If both top corners of $R_{op}$ have their vertical projections onto $\mathsf{int}(\ell)$, one of the vertical sides of $R_{op}$ is not on a $\mathsf{v}$-cut, so there is no uni-rectangle partition induced by the triplet $(Q, o, p)$. See Figure 7(a). □

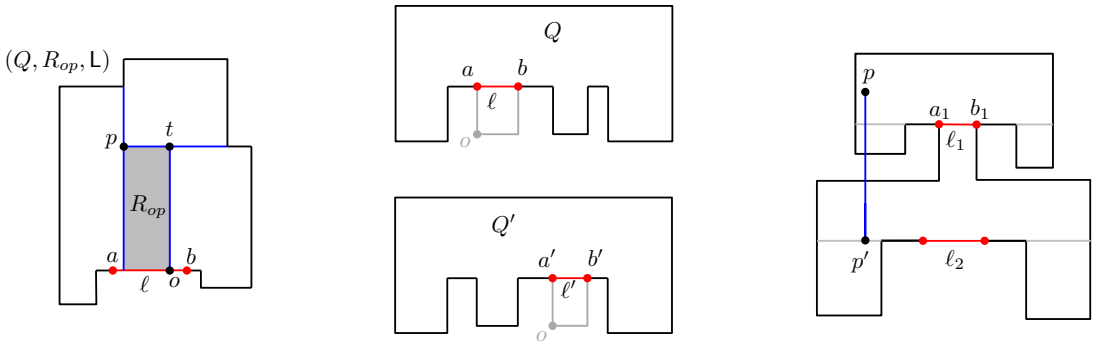



Figure 7: (a) If both top corners $p$, $t$ of $R_{op}$ have their vertical projections onto $\mathsf{int}(\ell)$, one vertical side ($ot$ in the figure) of $R_{op}$ is not on a $\mathsf{v}$-cut in $(Q, R_{op}, \mathsf{L})$. (b) If both $Q$ and $Q'$ are considered, $o \in P(a, b)$ and $o \in P(a', b')$. This is not possible because $P(a, b) \cap P(a', b') = \emptyset$. (c) Proof of Lemma 10.

**Lemma 10.** *For each grid point $p \in \mathsf{G}$, the algorithm considers at most one subpolygon $Q$ in $H^+$ such that $pp^o \subset Q$ and $p^o \notin \ell$ but $p^o c \subset Q$ for an endpoint $c$ of $\ell$ and the vertical projection $p^o$ of $p$ onto the line through $\ell$.*

*Proof.* For any subpolygon of $P$ (including $P$) and two points $a$ and $b$ on $\partial P$, we denote by $P(a, b)$ the subchain on $\partial P$ in the clockwise direction from $a$ to $b$.

We first show that among all subpolygons in $H^+$ and boundary cuts lying on the same horizontal grid line of $\mathsf{G}$, at most one of them is considered by the algorithm. Suppose there are two such subpolygons $Q$ and $Q'$ with (horizontal) boundary cuts $\ell = ab$ and $\ell' = a'b'$ considered by the algorithm, respectively. Then no vertical line intersects both $\ell$ and $\ell'$. Moreover, $o \in P(a, b)$ and $o \in P(a', b')$ for the origin point $o$ used in the first recursive step of the algorithm. This is not possible because $P(a, b) \cap P(a', b') = \emptyset$. Thus, there is at most one subpolygon considered by the algorithm among subpolygons in $H^+$, with boundary cuts lying on the same horizontal grid line. See Figure 7(b).

Suppose there exist two subpolygons $Q_1$ and $Q_2$ satisfying the lemma statement, with horizontal boundary cuts $\ell_1 = a_1 b_1$ and $\ell_2$ of $Q_1$ and $Q_2$, respectively, such that $\ell_1$ lies above $\ell_2$. Then $pp'$ intersects the boundary of $Q_2$ at a point in $Q_2(a_1, b_1)$, contradicting $pp' \subset Q_2$ for the vertical projection $p'$ of $p$ onto the line through $\ell_2$. See Figure 7(c). □

With an argument similar to the one in the proof of Lemma 7, we can bound the number of subpolygons of type $\mathsf{1R}$.

**Lemma 11.** *There are $O(n)$ subpolygons of type $\mathsf{1R}$.*

With Lemmas 9, 10, and 11, we can bound the number of candidate triplets of type $\mathsf{1R}$.

**Lemma 12.** *There are $O(n^3)$ candidate triplets of type $\mathsf{1R}$.*

*Proof.* We consider only the subpolygons in $H^+$. The other cases can be handled similarly. By Lemma 9, we count the candidate triplets $(Q, o, p)$ of type 1R such that at least one of the top corners of $R_{op}$, denoted $s$, satisfies $ss^o \subset Q$, $s^o \notin \mathsf{int}(\ell)$, and $s^o a \subset Q$ for an endpoint $a$ of $\ell$, where $s^o$ is the vertical projection of $s$ onto the line through $\ell$.

First, we count the candidate triplets such that $s^o \notin \ell$. By Lemma 10, there is at most one subpolygon of type 1R for $s$. By checking all $O(n)$ origin points for the subpolygon, there are $O(n)$ candidate triplets for each grid point of G. Since there are $O(n^2)$ grid points of G, there are $O(n^3)$ such candidate triplets.

Then, we count the candidate triplets $(Q, o, p)$ such that $s^o$ lies on an endpoint of $\ell$. For each $Q$ of type 1R, there are $O(n^2)$ such candidate triplets because there are $O(n)$ horizontal grid lines of G for the choice of $s$ and $O(n)$ vertical grid lines of G for the choice of $o$. By Lemma 11, there are $O(n^3)$ such candidate triplets in total.

So, the total number of candidate triplets of type 1R is $O(n^3)$. □

#### 4.1.4 Candidate triplets of type 2R.

We bound the number of candidate triplets satisfying condition (1) of rule 2 in Lemma 3 to $O(n^3)$ in a way similar to the one in Section 4.1.1. For the candidate triplets satisfying condition (2), we classify them into two subtypes depending on the position of $p$, and then bound the number of candidate triplets to be checked by the algorithm to $O(n^3)$ for each type.

We assume that the cut direction of $Q$ is up-right. The four regions of the plane subdivided by the vertical line and the horizontal line through $o$ are called the *o-quadrants*, labeled from $o_{\mathsf{I}}$ (top-right region) to $o_{\mathsf{IV}}$ (bottom-right region), in counterclockwise direction around $o$. By condition (2) of rule 2 in Lemma 3, the partner point $p$ from a candidate triplet $(Q, o, p)$ of type 2R lies either in $o_{\mathsf{II}}$ or $o_{\mathsf{IV}}$.

Given a candidate triplet $(Q, o, p)$ of type 2R, $p$ is *cut-visible* in $Q$ ($(Q, o, p)$ of type 2Rv) if $p$ has an orthogonal projection on $\ell_v$ or on $\ell_h$. It is *cut-invisible* ($(Q, o, p)$ of type 2Ri) otherwise.

**Candidate Triplets $(Q, o, p)$ with $p$ cut-invisible (2Ri).** For each partner point $p$, there is at most one subpolygon of type 2R per grid line, among those with the same cut direction and cut-invisible $p$.

**Lemma 13.** *For each grid point $p$, there is at most one subpolygon $Q$ of type 2R with origin point $o$ such that $p \in o_{\mathsf{IV}}$ for the candidate triplet $(Q, o, p)$ among the subpolygons with up-right cut direction, origin points lying on the same vertical grid line, and $p$ cut-invisible.*

*Proof.* The proof is similar to the one for Lemma 5. Suppose that there are two candidate triplets $(Q, o, p)$ and $(Q', o', p)$ satisfying the conditions in the lemma statement. Assume that $y(o') < y(o)$. Let $u$ be the right endpoint of the horizontal boundary cut $\ell'_h$ of $Q'$. If $p \in o'_{\mathsf{IV}}$ then $x(p) > x(u) > x(o') = x(o)$ and $y(p) < y(u) = y(o') < y(o)$. Thus, $u \in \mathsf{int}(R_{op})$ and $R_{op} \not\subset Q$ because $u$ is a reflex vertex of $Q$. This contradicts that $(Q, o, p)$ is a candidate triplet. □

Lemma 13 also holds for grid point $p$ in $o_{\mathsf{II}}$ and origin points lying on the same horizontal grid line. The number of candidate triplets of type 2Ri per grid line is thus $O(n^2)$ by the choice of $p$, which proves $\mathsf{T}_{\mathsf{2Ri}} = O(n^3)$.

**Lemma 14.** *There are $O(n^3)$ candidate triplets of type 2Ri.*

*Proof.* We can group the candidate triplets $(Q, o, p)$ with $p \in o_{\mathsf{II}}$ and $p$ cut-invisible in the subpolygon by the vertical grid lines of G where their origin points lie. Similarly, we can group the candidate triplets $(Q, o, p)$ with $p \in o_{\mathsf{IV}}$ and $p$ cut-invisible in the subpolygon by the horizontal

grid lines of $\mathsf{G}$ where their origin points lie. By Lemma 13, there are $O(n^3)$ candidate triplets grouped by grid lines. This number is linear to the number of candidate triplets of type 2Ri because there are $O(n^2)$ grid points and $O(n)$ grid lines of $\mathsf{G}$. □

**Candidate Triplets $(Q, o, p)$ with $p$ cut-visible (2Rv).** Assume that the vertical grid lines are indexed from left to right, and the horizontal grid lines are indexed from bottom to top. Let $\mathsf{G}(i, j)$ (or simply $(i, j)$ if understood in context) denote the grid point which is the intersection point of the $i$-th vertical grid line and the $j$-th horizontal grid line. Observe that there is a unique cutset (consisting of v-cuts) that determines $R_{op}$ for each candidate triplet $(Q, o, p)$ of type 2Rv, unless a corner of $R_{op}$ coincides with the right endpoint of $\ell_h$ or the upper endpoint of $\ell_v$ of $Q$. If a corner coincides with such an endpoint of the boundary cuts, there can be at most two cutsets, but this does not affect the asymptotic time complexity of the algorithm. So we assume that there is no such case.

Observe that $p$ has an orthogonal projection in the interior of a boundary cut, and there is a one-to-one correspondence between candidate triplets $(Q, o, p)$ and uni-rectangle partitions $\mathsf{U} = (Q, R_{op}, \mathsf{L})$ of type 2Rv. See Figure 8(a). Abusing the notation, let $\mathsf{lnk}(Q, o, p) := \mathsf{lnk}(Q, R_{op}, \mathsf{L})$. We let $\mathsf{lnk}(Q, o, p) = \infty$ if $p$ is not a partner point of $o$. For a subpolygon $Q$ of type 2Rv, we call an element $p^*$ of a set $S$ of grid points of $\mathsf{G}$ an *optimal partner point* of $o$ in $S$ if $p^* \in \arg\min_{p \in S} \mathsf{lnk}(Q, o, p)$.

**Lemma 15.** *Let $Q$ and $Q'$ be subpolygons of type 2R with up-right cut direction and origin points $o = (i, j)$ and $o' = (i, j + 1)$, respectively. Let $S = \{(i', j') \mid j' < j\}$ for any fixed integer $i' > i$. If $p$ is an optimal partner point of $o$ in $S$, then $p$ is also an optimal partner point of $o'$ in $S$.*

*Proof.* We show that

$$\begin{aligned} \mathsf{lnk}(Q, o, p) < \mathsf{lnk}(Q, o, q) &\Rightarrow \mathsf{lnk}(Q', o', p) <= \mathsf{lnk}(Q', o', q), \\ \mathsf{lnk}(Q, o, p) = \mathsf{lnk}(Q, o, q) &\Rightarrow \mathsf{lnk}(Q', o', p) = \mathsf{lnk}(Q', o', q) \end{aligned} \tag{1}$$

for any grid points $p, q \in S$.

Let $\mathsf{U} = (Q, R_{op}, \mathsf{L})$ and $\mathsf{U}' = (Q', R_{o'p}, \mathsf{L}')$ be the uni-rectangle partitions induced by the triplet $(Q, o, p)$ and $(Q', o', p)$, respectively. Among the cuts in $\mathsf{L}$, we denote by $\ell_1$ the vertical cut incident to $o$, and by $\ell_2$ the vertical cut other than $\ell_1$ and incident to the horizontal boundary cut $\ell_h$ of $Q$. Let $Q_1$ be the subpolygon in $\mathsf{U} \setminus \{R_{op}\}$ whose boundary intersects both $\ell_2$ and $\ell_h$ in (nondegenerate) line segments. Similarly, we define $\ell'_1$ and $\ell'_2$ among cuts in $\mathsf{L}'$, and $Q'_1$ in $\mathsf{U}' \setminus \{R_{o'p}\}$. See Figure 8(b).

Because $Q$, $o$ and $i'$ are fixed, we can consider each of $\mathsf{lnk}(Q, o, (i', j'))$, $\mathsf{len}(\ell_1)$, $\mathsf{len}(\ell_2)$, and $\mathsf{lnk}(Q_1)$ as a (discrete) function of $j'$ with domain $\{1, ..., j - 1\}$. Similarly, we can consider each of $\mathsf{lnk}(Q', o', (i', j'))$, $\mathsf{len}(\ell'_1)$, $\mathsf{len}(\ell'_2)$, and $\mathsf{lnk}(Q'_1)$ as a (discrete) function of $j'$ with domain $\{1, ..., j - 1\}$.

Observe that $\mathsf{U}' = (\mathsf{U} \setminus \{Q_1, R_{op}\}) \cup \{Q'_1, R_{o'p}\}$ and $\mathsf{L}' = (\mathsf{L} \setminus \{\ell_1, \ell_2\}) \cup \{\ell'_1, \ell'_2\}$. So we have

$$\begin{aligned} \mathsf{lnk}(Q', o', p) &:= \sum_{\ell \in \mathsf{L}'} \mathsf{len}(\ell) + \sum_{Q' \in \mathsf{U}'} \mathsf{lnk}(Q') \\ &= \mathsf{lnk}(Q, o, p) - \mathsf{len}(\ell_1) - \mathsf{len}(\ell_2) - \mathsf{lnk}(Q_1) + \mathsf{len}(\ell'_1) + \mathsf{len}(\ell'_2) + \mathsf{lnk}(Q'_1) \\ &= \mathsf{lnk}(Q, o, p) + c. \end{aligned}$$

as a function of $j'$ for some constant $c$. This is because $\mathsf{len}(\ell'_1) - \mathsf{len}(\ell_1)$, $\mathsf{len}(\ell'_2) - \mathsf{len}(\ell_2)$, $\mathsf{lnk}(Q'_1)$, and $\mathsf{lnk}(Q_1)$ are all constants with respect to $j'$. Thus, both equations in (1) are satisfied.

A degenerate case may occur when the right endpoint of the horizontal boundary cut $\ell_h$ of $Q$ is a reflex vertex of $P$ and it coincides with the upper endpoint of a vertical edge of $Q$. In

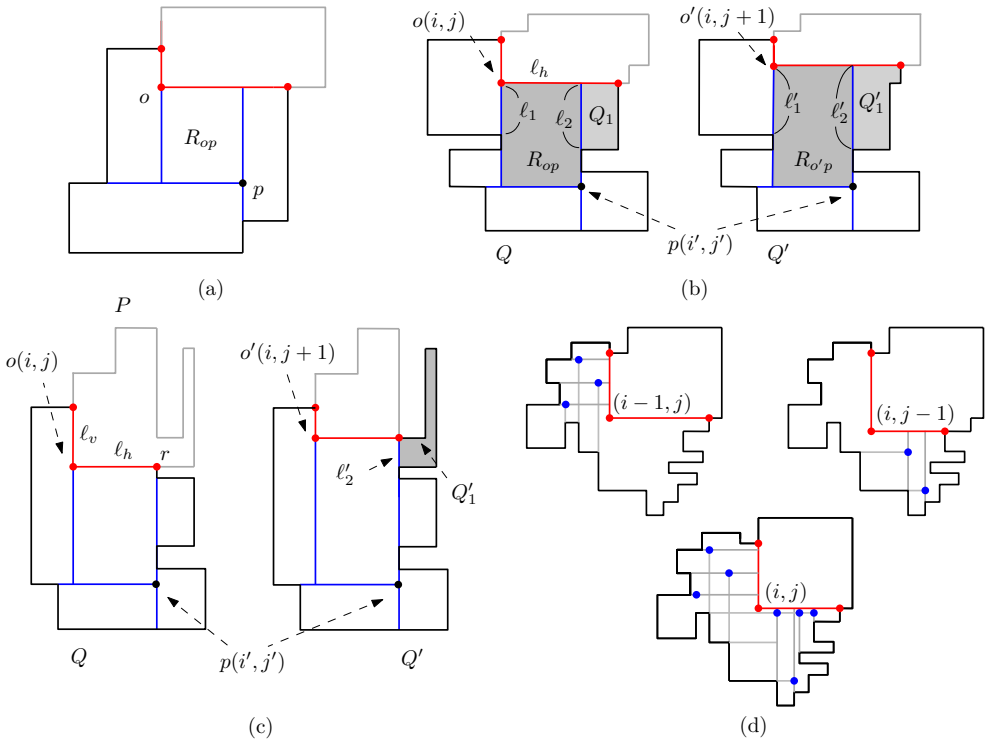



Figure 8: (a) Unique cutset $\mathsf{L}$ that determines $R_{op}$ for any candidate triplet $(Q, o, p)$ of type 2Rv satisfying condition (2) of rule 2 in Lemma 3. (b) $\mathsf{U}' = (\mathsf{U} \setminus \{Q_1, R_{op}\}) \cup \{Q'_1, R_{o'p}\}$ and $\mathsf{L}' = (\mathsf{L} \setminus \{\ell_1, \ell_2\}) \cup \{\ell'_1, \ell'_2\}$. (c) A reflex vertex $r$ of $P$ is the right endpoint of $\ell_h$ and the upper endpoint of a vertical edge of $Q$. We set $\mathsf{len}(\ell_2) = 0$ and $\mathsf{lnk}(Q_1) = 0$. (d) We update $A(i,j)$ using $A(i-1,j)$ and the grid points on the $(i-1)$-th vertical grid line, and then update $B(i,j)$ using $B(i,j-1)$ and the grid points on the $(j-1)$-th horizontal grid line in $O(n)$ time.

this case, we set $\mathsf{len}(\ell_2) = 0$ and $\mathsf{lnk}(Q_1) = 0$. Then, both equations in (1) are satisfied. See Figure 8(c).

Another degenerate case may occur when $R_{o'p} \notin Q'$ for every $p \in S$. In this case, $\mathsf{lnk}(Q', o', p) = \infty$ for every $p \in S$, and both equations in (1) are trivially satisfied. □

Indeed, there may exist origin points with which no subpolygon of type 2R with cut direction up-right is defined. We can handle such cases without increasing the running time of the algorithm.

By Lemma 15, each grid point $p$ of $\mathsf{G}$ is compared at most once as the partner point in the candidate triplets $(Q, o, p)$ among the subpolygons $Q$ of type 2R with up-right cut direction and origin points lying on the same grid line. To efficiently compute a partner point that minimizes $\mathsf{lnk}(Q, o, p)$ for each subpolygon $Q$ of type 2R with up-right cut direction, we define another grid $\mathsf{G}'$ coarser than $\mathsf{G}$ such that for every subpolygon of type 2R with up-right cut direction, its origin point is a grid point of $\mathsf{G}'$. $\mathsf{G}'$ is induced by half-lines, each of which is a ray going either leftward horizontally or downward vertically from a reflex vertex of $P$. Thus, every subpolygon of type 2R with up-right cut direction has its origin point on a grid point of $\mathsf{G}'$. Moreover, every grid point contained in $\mathsf{int}(P)$ is an origin point of a subpolygon of type 2R with up-right cut

direction. The vertical grid lines of $\mathsf{G}'$ are indexed from left to right, and the horizontal grid lines of $\mathsf{G}'$ are indexed from bottom to top.

We maintain two arrays, $A(i,j)$ and $B(i,j)$, for each grid point $o = \mathsf{G}'(i,j)$. $A(i,j)$ stores the partner points in quadrants $o_{\mathsf{II}}$ that minimize $\mathsf{lnk}(Q, o(i,j), p)$, one for each horizontal grid line of $\mathsf{G}'$. Similarly, $B(i,j)$ stores the partner points in quadrants $o_{\mathsf{IV}}$ that minimize $\mathsf{lnk}(Q, o(i,j), p)$, one for each vertical grid line of $\mathsf{G}'$. We update $A(i,j)$ using $A(i-1,j)$ and the grid points on the $(i-1)$-th vertical grid line of $\mathsf{G}'$, and then update $B(i,j)$ using $B(i,j-1)$ and the grid points on the $(j-1)$-th horizontal grid line of $\mathsf{G}'$ in $O(n)$ time. Then, we compute a partner point that leads to an optimal uni-rectangle partition, by comparing the partner points in $A(i,j)$ and $B(i,j)$ in $O(n)$ time. See Figure 8(d). The base cases are the grid points $\mathsf{G}'(i,j)$ such that $\mathsf{G}'(i,j)$ is contained in $\mathsf{int}(P)$ and both $\mathsf{G}'(i-1,j)$ and $\mathsf{G}'(i,j-1)$ are on $\partial P$. See Figure 9.

Since there are $O(1)$ cut directions and $O(n^2)$ grid points of $\mathsf{G}$, we have Lemma 16.

**Lemma 16.** *We can obtain $\mathsf{lnk}(\cdot)$ for all subpolygons of type $\mathsf{2R}$ satisfying rule 2 in Lemma 3, by checking $O(n^3)$ uni-rectangle partitions.*

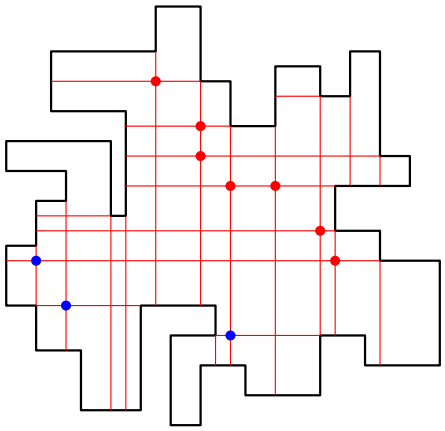


Figure 9: Red grid points are on a staircase chain of grid points. Blue grid points represent base cases.

With the bounds on the numbers of candidate triplets of types $\mathsf{1C}$, $\mathsf{1R}$, $\mathsf{2C}$ and $\mathsf{2R}$, to be considered by the algorithm to obtain an $\mathsf{i}$-partition of $P$, we have a main result.

**Theorem 1.** *We can compute an $\mathsf{i}$-partition of a rectilinear polygon with $n$ vertices and no holes in the plane in $O(n^3)$ time using $O(n^2)$ space.*

*Proof.* We analyze the time complexity of our algorithm. The preprocessing step (which will be given in Section 4.2) takes $O(n^2)$ time. By Lemmas 6, 8, 12, and 16, there are $O(n^3)$ uni-rectangle partitions to be considered in order to obtain an $\mathsf{i}$-partition. Since $\mathsf{lnk}(\cdot)$ value for each uni-rectangle partition can be obtain in $O(1)$ time after preprocessing, all the subpolygons encountered can be handled in $O(n^3)$ time in total. After computing the cuts inducing an $\mathsf{i}$-partition of $P$, we can construct the $\mathsf{i}$-partition in $O(n)$ time as the number of cuts is $O(n)$ by Lemma 2. So our algorithm runs in $O(n^3)$ time.

We analyze the space complexity of our algorithm. The preprocessing step for the algorithm (which will be given in Section 4.2) uses $O(n^2)$ space. The algorithm uses $O(1)$ space for each subpolygon of type $\mathsf{2C}$, $\mathsf{1C}$, or $\mathsf{1R}$, since it stores the best partner point found so far for each origin point. Since there are $O(n^2)$ such subpolygons, it uses $O(n^2)$ space for them in total.

Then, we analyze the space that the algorithm uses for subpolygons of type 2R. For each origin point $\mathsf{G}'(i,j)$ of a subpolygon of type 2R, our algorithm uses two arrays, one for the origin point $\mathsf{G}'(i-1,j)$ and one for origin point $\mathsf{G}'(i,j-1)$. Thus, it suffices to store the arrays corresponding to certain grid points of $\mathsf{G}'$ which are on a staircase chain of line segments on the grid lines of $\mathsf{G}'$. Since the size of each array is $O(n)$ and a staircase chain consists of $O(n)$ grid points at the same time, the algorithm uses $O(n^2)$ space for subpolygons of type 2R.

When handling each subpolygon and its origin point, the algorithm stores not only $\mathsf{lnk}(\cdot)$ value of a uni-rectangle partition but also the uni-rectangle partition itself. Because each uni-rectangle partition consists of $O(n)$ subpolygons with $O(n)$ vertices in total on their boundaries, the algorithm does not store it explicitly using a list of vertices. Instead, it stores each uni-rectangle partition using a pair of grid points corresponding to the origin and partner point, and a single number corresponding to the shape of cutsets. There are $O(1)$ distinct cutsets (consisting of v-cuts), each of which induces a uni-rectangle partition, for a given candidate triplet. Thus, we use $O(1)$ space for storing each uni-rectangle partition. Since the algorithm stores an optimal uni-rectangle partition for each $\leq$2-cut subpolygon, it requires $O(n^2)$ space in total for storing them in total. Finally, it uses $O(n)$ space to store an i-partition for $P$, because the total number of cuts in the i-partition is $O(n)$ by Lemma 2. □

Gonzalez and Zheng [11] showed that the i-partition algorithm by Lingas et al. [18] can be extended to yield an approximation algorithm for the i-partition partition problem on a rectangle containing $m$ point holes. Their approximation algorithm first transforms the rectangle with point holes into a weakly-simple rectilinear polygon $P'$ without point holes in $O(m^2)$ time by connecting the point holes with layered staircase chains to the boundary of the rectangle. Then, they apply the i-partition algorithm by Lingas et al. to $P'$ and compute an i-partition in $O(m^4)$ time using $O(m^2)$ space. By applying our i-partition algorithm instead of the one by Lingas et al., the time complexity gets improved.

**Corollary 2.** *Given a rectangle $R$ with $m$ point holes, we can compute a rectangular partition of $R$ in $O(m^3)$ time using $O(m^2)$ space such that no rectangle in the partition contains a point hole in its interior and the total length of the line segments used for the partition is within three times the optimal.*

### 4.2 i-partition algorithm made clear

Lingas et al. [18] claimed that their i-partition algorithm takes $O(n^4)$ time. The algorithm considers all candidate triplets for each subpolygon $Q$ of $P$. There are $O(1)$ uni-rectangle partitions $(Q, R_{op}, \mathsf{L})$ for a candidate triplet. For example, any triplet $(Q, o, p)$ of type 2C can have at most three different cutsets, each forming a uni-rectangle partition induced by the triplet. See Figure 10 for an illustration.

They claimed that given a triplet $(Q, o, p)$, they can check if $R_{op} \subset Q$ and compute $\mathsf{lnk}(\mathsf{U})$ for each of the uni-rectangle partitions $\mathsf{U} = (Q, R_{op}, \mathsf{L})$ induced by the triplet in $O(1)$ time. However, they did not provide any details on how to do this in $O(1)$ time.

We show that this can be done $O(1)$ time with some preprocessing. We do not compute the uni-rectangle partitions explicitly, because each partition may have $O(n)$ subpolygons. Instead, we observe that there are $O(1)$ subpolygons in partition $(Q, R_{op}, \mathsf{L})$, each sharing a corner with $R_{op}$, and $O(n)$ subpolygons, each sharing a side (but sharing no corner) with $R_{op}$. We compute $\mathsf{lnk}(\cdot)$ for the subpolygons sharing a corner with $R_{op}$ in a brute-force way, and use range-sum queries to compute the sum of $\mathsf{lnk}(\cdot)$ values for the subpolygons sharing a side with $R_{op}$. This takes $O(1)$ time in total.

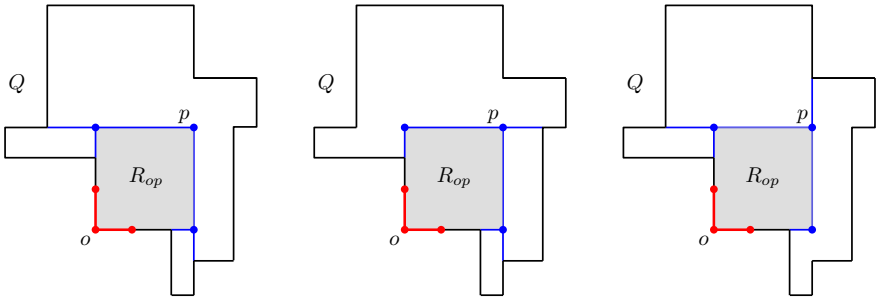



Figure 10: Subpolygon $Q$ of type 2C and uni-rectangle partitions induced by the triplet $(Q, o, p)$. There are three distinct cutsets each of which induces a uni-rectangle partition.

**Preprocessing.** We preprocess $P$ such that given two grid points $o, p \in \mathsf{G}$, one can determine if $R_{op} \subseteq P$ or not in $O(1)$ time. If $R_{op} \subseteq P$, $\mathsf{Ink}(\mathsf{U})$ for $\mathsf{U} = (Q, R_{op}, \mathsf{L})$ can also be obtained in $O(1)$ time. We use a line sweep algorithm [28] to construct a query structure (two matrices of size $O(n^2)$ and $O(n)$ arrays of length $O(n)$) in the preprocessing.

A $\leq$2-cut subpolygon of $P$ can be specified by a grid point $g \in \mathsf{G}$, the cut directions $d$ from $g$, and a truth value $t$. See Figure 11(a,b). For a 1-cut subpolygon with boundary cut $\ell$, $g$ is the right endpoint (if $\ell$ is horizontal) or the upper endpoint (if $\ell$ is vertical), and $d$ indicates whether $P$ lies to the right of the ray emanating from $b$ and containing $\ell$. For a 2-cut subpolygon, $g$ is the endpoint of shared by the two boundary cuts, and $d$ indicates whether $g$ is a convex vertex or a reflex vertex of the subpolygon.

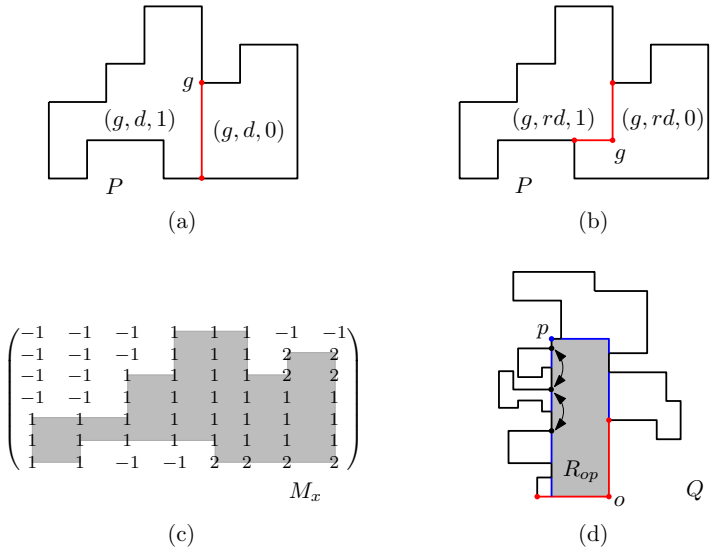



Figure 11: (a) A rectilinear polygon $P$ and two 1-cut subpolygons induced by a cut incident to $g$. (b) Two 2-cut subpolygons of $P$ induced by two cuts incident to $g$. (c) Each element of $M_x$ corresponds to a grid point of $\mathsf{G}$. The gray region represents $P$. (d) For subpolygons of type 1C with boundary cuts lying on the same grid line of $\mathsf{G}$, we store pointers between two consecutive subpolygons along the grid line.

With a line sweep, we construct a matrix $M_x$ ($M_y$) such that each element corresponds to a grid point and two elements in the same row (column) have the same value if and only if the line segment connecting the grid points of $\mathsf{G}$ corresponding to the two elements is contained in $P$. See Figure 11(c) for an illustration. The line sweep takes $O(n^2)$ time and space. Because $P$ has no holes, $R_{op}$ is contained in $P$ if and only if its four sides are all contained in $P$. By comparing the elements in $M_x$ ($M_y$) corresponding to the two horizontal (vertical) sides of $R_{op}$, we can check if $R_{op} \subset P$ in $O(1)$ time.

To compute $\mathsf{lnk}(\mathsf{R_U})$ for an optimal refinement $\mathsf{R_U}$ for a given uni-rectangle partition $\mathsf{U} = (Q, R_{op}, \mathsf{L})$ in $O(1)$ time, we apply another line sweep that takes $O(n^2)$ time and space. During the line sweep, we consider the subpolygons of type $\mathsf{1C}$ with boundary cuts lying on the grid line of $\mathsf{G}$ containing the boundary cut of $Q$, and store pointers between two consecutive subpolygons along the grid line. When a subpolygon $Q$ of type $\mathsf{1C}$ is handled during the algorithm, the algorithm computes $\mathsf{lnk}(\cdot)$ value of $Q$ and $\mathsf{lnk}(\cdot)$ values of the subpolygons linked by the pointers. See Figure 11(d). Then, we store $\mathsf{lnk}(\cdot)$ values of such subpolygons in an array for each grid line of $\mathsf{G}$. Also, we store the lengths of collinear edges of $P$ in an array for each grid line of $\mathsf{G}$.

Once we stored the lengths and $\mathsf{lnk}(\cdot)$ values in arrays, we compute $\mathsf{lnk}(\cdot)$ values of the subpolygons in a uni-rectangle partition $\mathsf{U} = (Q, R_{op}, \mathsf{L})$ as follows. There are $O(n)$ subpolygons that share a side (sharing no corner) with $R_{op}$. We compute $\mathsf{lnk}(\cdot)$ values for these subpolygons by performing four range-sum queries, each for a side of $R_{op}$. There are $O(1)$ subpolygons that share a corner with $R_{op}$. We compute their $\mathsf{lnk}(\cdot)$ values in a brute-force way. Finally, we compute the total length of the line segments in $\mathsf{L}$ by performing four range-sum queries to obtain the length of the intersection between $\partial P$ and the sides of $R_{op}$. All these queries and computations take $O(1)$ time in total.

Observe that $\sum_{\ell \in \mathsf{L}} \mathsf{len}(\ell)$ is the perimeter of $R_{op}$ minus $\sum_{\ell \in \partial P \cap R_{op}} \mathsf{len}(\ell)$ plus the lengths of at most four line segments, each sharing a corner with $R_{op}$, which can be computed in $O(1)$ time. Because $\mathsf{lnk}(\mathsf{U}) := \sum_{\ell \in \mathsf{L}} \mathsf{len}(\ell) + \sum_{Q' \in \mathsf{U}} \mathsf{lnk}(Q')$ by definition, we can compute $\mathsf{lnk}(\mathsf{U})$ in $O(1)$ time.

# 5 t-partition algorithm

## 5.1 Rectilinear polygon with no holes

We give a t-partition algorithm for a rectilinear polygon $P$ with $n$ vertices and no holes under the vertex incidence. Our algorithm is based on the vt-algo with $O(n^5)$ time and $O(n^4)$ space by O'Rourke and Tewari [24]. But with some modifications in a way similar to the one in Section 4, our algorithm returns a vt-partition in $O(n^3 \log^2 n)$ time using $O(n^3)$ space.

Our algorithm classifies subpolygons into four types and enumerates the uni-rectangle partitions of each type without duplicates. We define $\mathsf{Width}(\cdot)$ functions as follows.

- $\mathsf{Width}(R) := \min(a, b)$, for a rectangle $R$ with side lengths $a$ and $b$.
- $\mathsf{Width}(\mathsf{R}) := \min_{R \in \mathsf{R}} \mathsf{Width}(R)$, for a rectangular partition $\mathsf{R}$.
- $\mathsf{Width}(Q)$ is the maximum of $\mathsf{Width}(\mathsf{R})$ over all rectangular partitions $\mathsf{R}$ of a rectilinear polygon $Q$.
- $\mathsf{Width}(Q, R_{ab}, \mathsf{L}) := \min_{Q' \in \mathsf{U}} \mathsf{Width}(Q')$ for a uni-rectangle partition $\mathsf{U} = (Q, R_{ab}, \mathsf{L})$ of a rectilinear polygon $Q$.

The running time of our algorithm is proportional to the number of uni-rectangle partitions for which we compute $\mathsf{Width}(\cdot)$ values. The rules in Lemma 3 still hold for $\leq$2-cut subpolygons,

even if we seek for a vt-partition. Hence, the number of candidate triplets considered by our algorithm for each $\leq$2-cut subpolygon remains the same for each subtype.

**Lemma 17.** *There are $O(n^3)$ candidate triplets $(Q, o, p)$ of types* 1C, 1R, 2C, *and* 2Ri.

We can preprocess $P$ so that $\mathsf{Width}(\cdot)$ value and the size of a given uni-rectangle partition can be computed efficiently, as in the following lemma. Recall that $\mathsf{card}(\mathsf{P})$ denotes the number of subpolygons in a partition P of a rectilinear polygon.

**Lemma 18.** *Given a uni-rectangle partition $\mathsf{U} = (Q, R_{op}, \mathsf{L})$, we can compute $\mathsf{Width}(\mathsf{U})$ and $\mathsf{card}(\mathsf{U})$ in $O(\log n)$ time, after preprocessing of $P$ in $O(n^2 \log n)$ time using $O(n^2)$ space.*

*Proof.* We preprocess $P$ in a way similar to the one in Section 4.2. But when we store $\mathsf{Width}(\cdot)$ values of the subpolygons of type 1C, we use segment trees instead of arrays. Using segment trees (Section 9.2.2 of [14]), we can answer a range minimum query of $n$ numbers in $O(\log n)$ time. Thus, we can obtain $\mathsf{Width}(\mathsf{U})$ in $O(\log n)$ time by applying range minimum queries on the four sides of $R_{op}$ and compare the result with $\mathsf{Width}(\cdot)$ values of $O(1)$ subpolygons in U sharing a corner with $R_{op}$ in a brute-force manner.

For each segment tree storing $\mathsf{Width}(\cdot)$ values of subpolygons, we maintain an array storing the number of rectangles in a vt-partition of each subpolygon. With this, we can carry out sum queries to compute $\mathsf{card}(\mathsf{U})$ in constant time.

Each segment tree can be constructed in $O(n \log n)$ time using $O(n)$ space, as $O(1)$ values are stored in each of the $O(n)$ nodes. For each array, we use $O(n)$ time and space. Since there are $O(n)$ grid lines of G, the preprocessing takes $O(n^2 \log n)$ time and $O(n^2)$ space. □

By Lemmas 17 and 18, we have the following corollary.

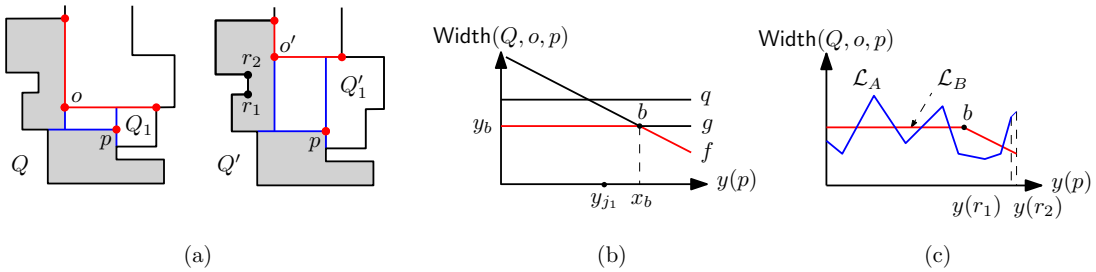
**Corollary 3.** *We can compute $\mathsf{Width}(\cdot)$ values of all uni-rectangle partitions of types* 1C, 1R, 2C, *and* 2Ri *in $O(n^3 \log n)$ time using $O(n^2)$ space.*

Our algorithm handles the uni-rectangle partitions of subtype 2Rv using the coherence between them as in Section 4.1.4. Since each candidate triplet of type 2Rv induces only one uni-rectangle partition, we abuse the notation for a candidate triplet to represent its corresponding uni-rectangle partition. Given a subpolygon $Q$ of type 2Rv, its origin point $o$, and a set S of grid points of G, we call an element $p^* \in S$ an *optimal partner point* of $o$ in $S$ if $p^* \in \arg\max_{p \in S} \mathsf{Width}(Q, o, p)$ and $\mathsf{card}(Q, R_{op^*}, \mathsf{L}) \leq \mathsf{card}(Q, R_{op'}, \mathsf{L})$ for any $p' \in \arg\max_{p \in S} \mathsf{Width}(Q, o, p)$.

We show that for the subpolygons of type 2Rv with up-right cut direction and origin points lying on a same vertical grid line of G, we can compute an optimal partner point for each origin point among the partner points lying on another vertical grid line of G efficiently.

Let $Q$ be a subpolygon of type 2R with up-right cut direction and origin point $o = \mathsf{G}(i, j_1)$. By the vertex incidence, $p = \mathsf{G}(i', j')$ for $i' > i$ and $j' < j_1$ can be a partner point of $o$ only if there is a reflex vertex of $P$ lying on the $j_1$-th line of G and there is a horizontal projection from the reflex vertex onto the left side of $R_{op}$. Hence, in the following lemma, we let $\mathsf{Width}(Q, o, p) := 0$ if $P$ does not contain such a reflex vertex lying on the same horizontal grid line of G with $p$.

**Lemma 19.** *Let $Q$ and $Q'$ be subpolygons of type* 2R *with up-right cut direction and origin points $o = \mathsf{G}(i, j_1)$ and $o' = \mathsf{G}(i, j_2)$, respectively. Let $S = \{\mathsf{G}(i', j') \mid j' < j_1\}$ and $S' = \{\mathsf{G}(i', j') \mid j' < j_2\}$ for any fixed integer $i' > i$. We can preprocess the uni-rectangle partition $(Q, R_{op}, \mathsf{L})$ for an optimal partner point $p$ of $o$ in $S$ such that an optimal partner point of $o'$ in $S'$ can be computed in $O(k \log^2 n)$ time, where $k$ is the number of reflex vertices of $Q'$ that lie in $o_{\mathsf{II}} \cap o'_{\mathsf{III}}$, each having a horizontal projection onto the left side of $R_{o'p}$.*




(a) (b) (c)

Figure 12: (a) Uni-rectangle partitions $(Q, R_{op}, \mathsf{L})$ with $o = (i, j_1)$ and $(Q', R_{o'p}, \mathsf{L}')$ with $o' = (i, j_2)$. The gray regions are $\mathsf{Q}(p)$. $r_1, r_2$ are reflex vertices contained in $o_{\mathsf{II}} \cap o'_{\mathsf{III}}$ that have a horizontal projection onto the left side of $R_{o'p}$. (b) Since $g$ and $q$ are constant for $y(p)$ and $f$ is decreasing linearly for $y(p)$, the lower envelope of the three functions has only one bending point $b$ at which its slope changes. (c) As the origin point changes from $o$ to $o'$, one can update the polygonal chain (blue graph) in $O(k \log^2 n)$ time using a dynamic ray shooting data structure.

*Proof.* Let $\mathsf{U} = (Q, R_{op}, \mathsf{L})$ and $\mathsf{U}' = (Q', R_{o'p}, \mathsf{L}')$ be the uni-rectangle partitions induced by the triplets $(Q, o, p)$ and $(Q', o', p)$, respectively. We define $Q_1$ and $Q'_1$ as in the proof of Lemma 15. See Figure 12(a). Let $\mathsf{W}(p) := \min_{P' \in \mathsf{Q}(p)} \mathsf{Width}(P')$ for $\mathsf{Q}(p) = \mathsf{U} \setminus \{R_{op}, Q_1\} = \mathsf{U}' \setminus \{R_{o'p}, Q'_1\}$. We use $x_a$ to denote the $x$-coordinate of the $a$-th vertical grid line of $\mathsf{G}$, $y_a$ to denote the $y$-coordinate of the $a$-th horizontal grid line of $\mathsf{G}$, and $|t|$ to denote the absolute value of a real number $t$. Then we have the following equations.

$$\mathsf{Width}(Q, o, p) = \min_{P' \in \mathsf{U}} \mathsf{Width}(P') = \min\{|y_{j_1} - y_{j'}|, |x_i - x_{i'}|, \mathsf{Width}(Q_1), \mathsf{W}(p)\}.$$
$$\mathsf{Width}(Q', o', p) = \min_{P' \in \mathsf{U}'} \mathsf{Width}(P') = \min\{|y_{j_2} - y_{j'}|, |x_i - x_{i'}|, \mathsf{Width}(Q'_1), \mathsf{W}(p)\}$$

Observe that $\mathsf{W}(p)$ is used in both equations. Thus, once we compute $\mathsf{W}(p)$ for grid points $p \in S$, we use them to obtain an optimal partner point of $o'$ among grid points in $S'$.

We store $\mathsf{W}(p)$ for grid points $p \in S$ in two different ways; as a segment tree [3] and as a polygonal chain. The segment tree is a balanced binary search tree with keys $\{1, ..., n\}$ at the leaf nodes. For $j' \in \{1, ..., j-1\}$, the $j'$-th leaf node stores $\mathsf{W}(p)$ for $p = \mathsf{G}(i', j')$. The leaf node also stores $\mathsf{card}(\mathsf{Q}(p))$ for $p = \mathsf{G}(i', j')$. For $j' \in \{j, ..., n\}$, the $j'$-th leaf node stores the value $-\infty$. The segment tree supports range maximum queries $\max(a, b)$ for integers $a, b$ with $1 \le a \le b \le n$, which returns the index of the leaf node minimizing $\mathsf{card}(\mathsf{Q}(p))$ among those maximizing $\mathsf{W}(p)$ for $p = \mathsf{G}(i', j')$ with $j' \in \{a, \ldots, b\}$. If there are more than one leaf node minimizing $\mathsf{card}(\mathsf{Q}(p))$ and maximizing $\mathsf{W}(p)$, the segment tree returns the index of any such leaf node.

Let $\mathcal{L}_A$ denote the graph constructed by connecting every two consecutive points $(y(p), \mathsf{W}(p))$ along the $y(p)$-axis by a line segment for grid points $p \in S$. See the blue graph in Figure 12(c).

Before computing an optimal partner point of $o'$ in $S'$, we update $\mathcal{L}_A$ and the segment tree. We update $\mathcal{L}_A$ by adding points $(y(p), \mathsf{W}(p))$ corresponding to the partner points $p \in S' \setminus S$. We add a line segment connecting every two consecutive points. We update the segment tree by updating leaf nodes corresponding to the partner points $p \in S' \setminus S$, and by updating some internal nodes.

Then, we obtain an optimal partner point of $o'$ in $S'$. Because such an optimal partner $p = \mathsf{G}(i', j')$ maximizes $\mathsf{Width}(Q', o', p)$, we can obtain it by computing the highest point on the lower envelope of $\mathcal{L}_A$ and $\mathcal{L}_B$, where $\mathcal{L}_B$ denotes the lower envelope of the discrete functions $|y_{j_2} - y_{j'}|$, $|x_i - x_{i'}|$, and $\mathsf{Width}(Q'_1)$. To make the computation easier, we again interpolate each

of the three functions by adding edges between two neighboring vertices, and call them $f$, $g$, and $q$ in order. For example, $f$ is the interpolation of $|y_{j_2} - y_{j'}|$.

Because $g$ and $q$ are constant functions of $y(p)$ and $f$ is a linearly decreasing function of $y(p)$, The lower envelope of $f$, $g$, and $q$, denoted $\mathcal{L}_B$, has at most one *bending point*, at which its slope changes. If $\mathcal{L}_B$ has no bending point, then the partner point $\mathsf{G}(i', j')$ with $j' = \max(1, j_2)$ is an optimal partner point of $o'$ in $S'$. So, suppose $\mathcal{L}_B$ has a bending point $b$. Denote by $j_1$ the index of the bottommost horizontal grid line of $\mathsf{G}$ among the horizontal grid lines satisfying $y_{j_1} \leq x_b$. See Figure 12(b). We first query $\max(1, j_b)$ on the segment tree. If $y_b \leq \mathsf{W}(p)$ for the partner point $\mathsf{G}(i', j')$ where $j' = \max(1, j_b)$, then the partner point $\mathsf{G}(i', j')$ is an optimal partner point of $o'$ in $S'$. Otherwise, $y_b > \mathsf{W}(p)$, and we carry out a ray shooting query onto the polygonal chain where the ray emanates from $b$ and proceeds along $f'$, to find out the edge $e$ of the polygonal chain which is hit first by the ray. If there is such an edge $e$, the partner point $\mathsf{G}(i', j')$ with $j' = \max(1, j_e)$ is an optimal partner point of $o'$ in $S'$, where $j_e$ is the index of the bottommost horizontal grid line of $\mathsf{G}$ among the horizontal grid lines whose $y$-coordinates are at most that of the right endpoint of $e$ on the polygonal chain. Otherwise, there is no edge on the polygonal chain hit by the ray, and the partner point $\mathsf{G}(i', j')$ with $j' = \max(1, j_2)$ is an optimal partner point of $o'$ in $S'$.

We now analyze the time complexity. Let $k$ denote the number of reflex vertices of $Q'$ that lie in $o_{\mathsf{II}} \cap o'_{\mathsf{III}}$, each having a horizontal projection onto the left side of $R_{o'p}$. It takes $O(\log^2 n)$ time to add a vertex or an edge on the polygonal chain, using a dynamic ray-shooting data structure for connected planar subdivisions [12]. The algorithm performs $O(k)$ such operations in total, so the total time for updating the polygonal chain is $O(k \log^2 n)$. We give at most one ray shooting query on the polygonal chain, which takes $O(\log^2 n)$ time.

It takes $O(\log n)$ time to update a node in the segment tree [3]. We update $O(k)$ nodes in total, so the total time for updating the segment tree is $O(k \log n)$. We give at most two $\max(\cdot, \cdot)$ queries on the segment tree, which take $O(\log n)$ time in total.

Thus, in total, it takes $O(k \log^2 n)$ time to compute an optimal partner point of $o'$ in $S'$. □

Using coherence among the candidate triplets, we can compare them efficiently and obtain the following lemma.

**Lemma 20.** *We can compute optimal partner points of $o$ for all candidate triplets $(Q, o, p)$ of type* 2Rv*, one for each, in $O(n^3 \log^2 n)$ time using $O(n^3)$ space.*

*Proof.* We consider only the candidate triplets of type 2Rv with up-right cut direction. For the candidate triplets of type 2Rv with other cut directions, we handle them analogously.

By Lemma 19, for origin points lying on the same vertical grid line of $\mathsf{G}$, we can compute their optimal partner points among the grid points of $\mathsf{G}$ on the same vertical grid line in $O(n \log^2 n)$ time, because there are $O(n)$ reflex vertices of $P$. We do this for $O(n^2)$ pairs of vertical grid lines on which origin and partner point lie, which takes $O(n^3 \log^2 n)$ time in total. Before computing the optimal partner points, we initialize each of the $O(n^2)$ segment trees such that all leaf nodes store the value $-\infty$. It takes $O(n \log n)$ time to initialize a segment tree, thus $O(n^3 \log n)$ time in total.

A degenerate case may occur when a *reflex* edge (whose both endpoints are reflex vertices of $P$) appears on the $i$-th or $i'$-th vertical grid line of $\mathsf{G}$. In this case, we update the relevant portion of the segment tree and the polygonal chain. It takes $O(n \log^2 n)$ time to update each of them, as we change $O(n)$ vertices and entries in the polygonal chain and the segment tree, respectively. Since there are $O(n)$ reflex edges of $P$ and each reflex edge induces $O(n)$ degenerate cases, we can handle all such degenerate cases in $O(n^3 \log^2 n)$ time in total.

We maintain a polygonal chain and a segment tree for each pair of vertical grid lines of G on which origin point and partner point lie. Because the size of each polygonal chain or segment tree is $O(n)$, the total space complexity is $O(n^3)$. □

By Corollary 3 and Lemma 20, it takes $O(n^3 \log^2 n)$ time and $O(n^3)$ space to check all uni-rectangle partitions and to obtain a vt-partition. After computing the cuts inducing a vt-partition, we can compute the vt-partition in $O(n)$ time and space as there are $O(n)$ cuts by Lemma 2. So we have another main result.

**Theorem 2.** *We can compute a vt-partition of a rectilinear polygon with n vertices and no holes in the plane in $O(n^3 \log^2 n)$ time using $O(n^3)$ space.*

We use *at-partition* to refer to a t-partition whose cutset consists of a-cuts.

**Lemma 21** (Theorem 6 of [24])**.** *There is an at-partition of $P$ such that every cut in the cutset lies on the canonical grid of $P$ or at fractions $\{\frac{1}{3}, \frac{1}{2}, \frac{2}{3}\}$ between two edges of $P$.*

By Lemma 21, there are $O(n^2)$ points on the boundary of $P$ where cuts can be incident. By treating those points as vertices, O'Rourke and Tewari obtained an $O(n^{10})$-time algorithm that computes an at-partition. By using our vt-partition algorithm, we have the following result.

**Corollary 4.** *We can compute an at-partition of a rectilinear polygon with no holes in the plane in $O(n^6 \log^2 n)$ time using $O(n^6)$ space.*

### 5.2 Rectilinear polygon with holes

We show that the decision version of the vt-partition problem for rectilinear polygons with holes is NP-complete.

**Problem statement.** $\mathsf{TH}(P, \delta, k)$ := Given a rectilinear polygon $P$ with $n$ vertices, including the vertices of holes, and a positive real value $\delta$ and a positive integer $k$, decide whether there exists a rectangular partition P of $P$ under the vertex incidence such that (1) $\mathsf{Width}(\mathsf{P}) \geq \delta$ and (2) $\mathsf{card}(\mathsf{P}) \leq k$.

We first show that TH is in NP. For a problem instance $(P, \delta, k)$, suppose that we are given a rectangular partition P of $P$ as a solution set. Because $\mathsf{card}(\mathsf{P}) = O(n)$, we can check if $\mathsf{card}(\mathsf{P}) \leq k$ and $\mathsf{Width}(\mathsf{P}) \geq \delta$ in $O(n)$ time.

To prove that TH is NP-hard, we use a reduction from P3SAT. In P3SAT, a 3-SAT formula $F$ with a variable set $X = \{x_1, ..., x_n\}$ and a clause set $C = \{c_1, ..., c_m\}$ is given such that the graph $G(F) = (V, E)$ with $V = X \cup C$ and $E = \{(x_i, c_j) \mid x_i \text{ or } \overline{x_i} \text{ is a literal in } c_j\}$ is planar. Here, each variable in $X$ has value either 1 (`true`) or 0 (`false`), and each clause in $C$ consists of exactly three variables in $X$ (*3-CNF*).

Our reduction algorithm follows the reduction scheme from P3SAT to the i-partition problem for rectilinear polygons with holes, by Lingas et al. [18]. It takes a 3-SAT formula as an input and constructs a rectilinear polygon such that the formula is satisfiable if and only if there exists a vt-partition of the rectilinear polygon satisfying the conditions (1) and (2) in the problem statement of TH. The rectilinear polygon is represented as the union of several *gadgets*, each corresponding to a variable (*variable* gadget, v-gadget in short), a clause (*clause* gadget, c-gadget in short), or a connection between them (*connection* gadget of types *turn*, *split*, *inverter*, and *phase shifter*).

In our reduction algorithm, we use all five gadget types (v-gadget, c-gadget, split, inverter, and phase shifter) used in the reduction algorithm by Lingas et al. [18], and introduce one more gadget type, *turn*. Some of our gadgets have holes while those by Lingas et al. have no holes.

The holes in our gadgets force certain partitions in order to satisfy the condition (1) in the problem statement of TH. See Figure 13. Each hole is either a square of side length $\frac{\delta}{2}$ or a rectangle with smaller side length $\frac{\delta}{2}$, where $\delta$ is from the problem instance $(P, \delta, k)$. Note that all corners of a hole in $P$ are reflex vertices of $P$, so there is exactly one cut incident to each corner of a hole in any partition of $P$ induced by v-cuts. Thus, there are only two ways of partitioning $P$ around each square hole, each consisting of four v-cuts: a v-windmill as shown in Figure 13(a) or a h-windmill as shown in Figure 13(b). The top-left corner of a square hole is incident to a vertical cut in the v-windmill while it is incident to a horizontal cut in a h-windmill. If there are two parallel cuts, each incident to a corner of the same side of a square hole, every resulting partition has $\mathsf{Width}(\cdot)$ value at most $\frac{\delta}{2}$ and does not satisfy the conditions (1) in the problem statement of TH. See Figure 13(c).

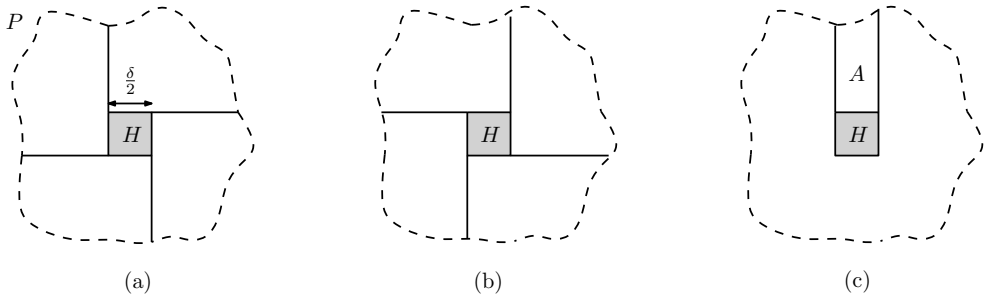



Figure 13: A rectilinear polygon $P$ and a hole $H$. $H$ is a square with side length $\frac{\delta}{2}$. (a) v-windmill. (b) h-windmill. (c) Two vertical cuts incident to the top corners of $H$. Every resulting rectangular partition has $\mathsf{Width}(\cdot)$ value at most $\mathsf{Width}(A) \leq \frac{\delta}{2}$.

We now describe the shapes of our gadgets: v-gadget, four connection gadgets (turn, split, inverter, and phase shifter), and c-gadget in order. In any gadget, every edge length is a multiple of $\frac{\delta}{2}$.

**v-gadget.** A v-gadget is a polygonal chain, which is the concatenation of two staircase chains of the same direction with edge length $\delta$, except the two edges incident to one common corner(vertex) of the staircases are of length $2\delta$. See Figure 14(a). In every polygon constructed by our reduction algorithm, a v-gadget is connected with another gadget by sharing the two endpoints of the v-gadget.

Consider the closed polygonal region bounded by a v-gadget and a horizontal line segment of length $2\delta$ which has an endpoint of the v-gadget as one of its endpoint and a (non-endpoint) vertex of the v-gadget as the other endpoint. See Figure 14(b). There is only one way to partition the polygonal region into minimum number of rectangles, and every cut inside such a partition is a horizontal cut. If there is a vertical cut, the number of rectangles in the resulting partition is not the minimum. See Figure 14(c). Similarly, we can show that the polygonal region bounded by a v-gadget and a vertical line segment of length $2\delta$ can be partitioned into minimum number of rectangles only if the cuts are all vertical. See Figure 14(d,e). The minimum partition of the region bounded by a v-gadget and a horizontal line segment corresponds to `false`, and the minimum partition of the region bounded by a v-gadget and a vertical line segment corresponds to `true`.

**Connection gadgets.** Each connection gadget consists of at most three staircase chains and at most one square hole. A turn gadget changes the direction (among the four diagonal directions) in which the truth value propagates. A split gadget receives a v-gadget and outputs two copies

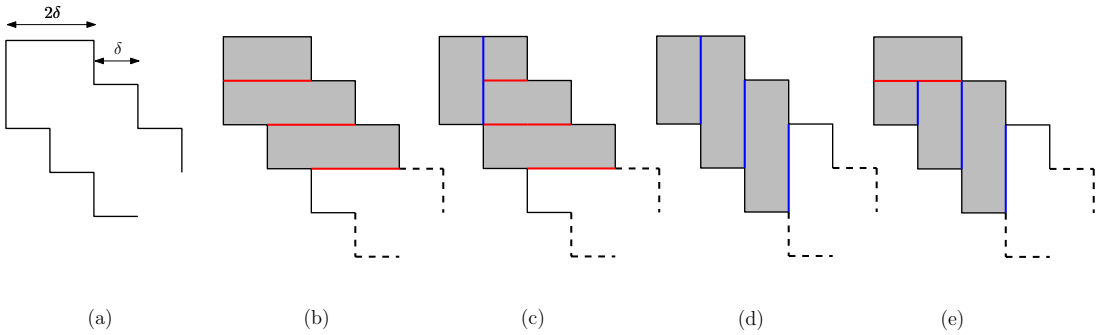


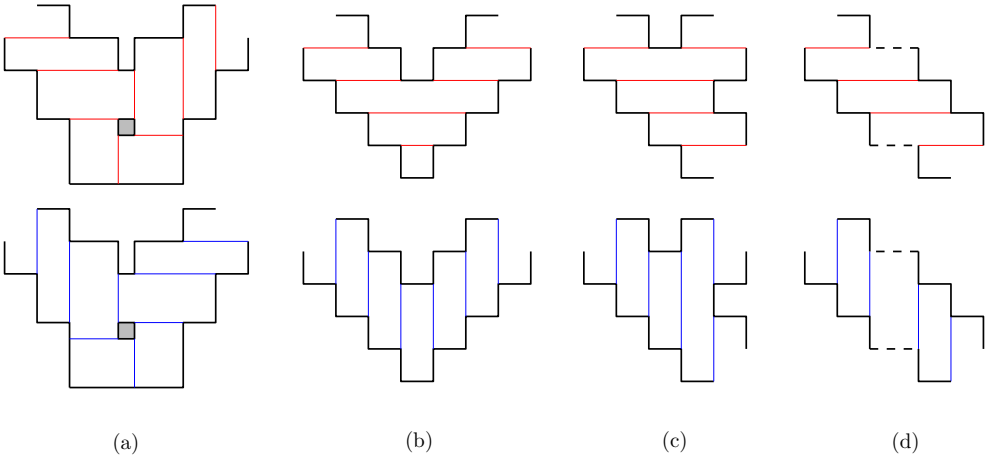

Figure 14: (a) A v-gadget. (b) A v-gadget connected by another gadget (dashed lines). The (gray) region is bounded by a v-gadget and a horizontal line segment of length $2\delta$, and its minimum partition corresponds to `false`. (c) If there is a vertical cut, the number of rectangles in the resulting partition is not the minimum. (d) The (gray) region is bounded by a v-gadget and a vertical line segment of length $2\delta$, and its minimum partition corresponds to `true`. (e) If there is a horizontal cut, the number of rectangles in the resulting partition is not the minimum.

of it. Each copied v-gadget consists of two staircase chains of the same direction, and there is no common corner of the staircases. An inverter gadget inverts the truth value transmitted from `true` to `false`, and vice versa. A phase shifter gadget connects a v-gadget and a c-gadget by matching the two endpoints of the v-gadget with two endpoints of the c-gadget (will be defined shortly), by using a pair of parallel edges of the same length, which is determined by the distance between the v-gadget and the c-gadget. See Figure 15 for an illustration of these gadgets.

**c-gadget.** A c-gadget consists of three staircase chains and four holes. A c-gadget can be connected to three v-gadgets. See Figure 16(a) for an illustration of a c-gadget. In the figure, the c-gadget is connected to v-gadgets, corresponding to three variables $x_1, x_2$, and $x_3$ in the 3-SAT formula. There are three square holes $(H_1, H_2, H_3)$ with side length $\frac{\delta}{2}$ and a rectangle hole $(H_4)$ whose smaller side length is $\frac{\delta}{2}$. Then every partition with the $\mathsf{Width}(\cdot)$ value at least $\delta$ has a windmill pattern around each square hole. Observe that there is no two horizontal cuts in such a partition, each incident to an endpoint of the same shorter side of the rectangle hole. The distance between any point on the boundary of the c-gadget and any point on the holes is larger than or equal to $\delta$. There is an inverter (including a square hole) in the top-left part of the figure, and it enforces that the c-gadget is not partitioned into minimum number of rectangles if and only if $x_1 = x_2 = x_3 = 0$. Figure 16(b) shows a partition of a c-gadget when the truth values of the variables are given as $x_1 = x_2 = x_3 = 1$. The partition has $\mathsf{Width}(\cdot)$ value at least $\delta$ and consists of the minimum number of rectangles. Figure 16(c,d) shows two different partitions of a c-gadget when the truth values of the variables are given as $x_1 = x_2 = x_3 = 0$. Every resulting partition either has $\mathsf{Width}(\cdot)$ value at most $\frac{\delta}{2}$ or does not consist of the minimum number of rectangles.

Given a 3-SAT formula $F$, we can construct the corresponding rectilinear polygon $P$ in polynomial time and space, by transforming the planar graph $G(F)$ corresponding to $F$ into a grid graph [4]. While constructing $P$, we pick a positive real value $\delta$ such that $\mathsf{Width}(P) = \delta$. For each gadget in $P$, we compute its vt-partition using any polynomial time algorithm. Let $k$ be the total number of rectangles from vt-partition of each gadget. Then, $F$ is satisfiable if and only if the answer to $\mathsf{TH}(P, \delta, k)$ is `true`. Thus, TH is NP-hard.

So we conclude with the following theorem.



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

Figure 15: (a) An inverter gadget. The size of the square hole (gray) is $\frac{\delta}{2}$. (b) A turn gadget. (c) A split gadget. (d) A phase shifter gadget. The phase shifter gadget connects a v-gadget and a c-gadget by matching the two endpoints of the v-gadget with two endpoints of the c-gadget. Dashed edges have the same length, and the length is determined by the distance between the v-gadget and the c-gadget.

**Theorem 3.** *TH is NP-complete.*

## 6 Conclusion and open problems

We gave an $O(n^3)$-time algorithm that computes an i-partition of a given rectilinear polygon without holes. We also gave an $O(n^3 \log^2 n)$-time algorithm that computes a vt-partition of a given rectilinear polygon without holes. Finally, we showed that the vt-partition problem for rectilinear polygons with holes is NP-complete. Two major open problems remain:

A. Does there exist a subcubic time algorithm that computes an i-partition of a rectilinear polygon without holes?

B. If condition (2) is removed from TH, is the problem still NP-hard?

Using coherence between uni-rectangle partitions, we could reduce the number of uni-rectangle partitions to be checked to compute an i-partition from $O(n^4)$ to $O(n^3)$, but not any further. We conjecture that any algorithm that iterates over all uni-rectangle partitions to be checked runs in $\Omega(n^3)$ time in the worst case, hence problem A is of great interest to us.

Our NP-hardness proof of TH relies on both conditions (1) and (2) of TH. There can be several partitions satisfying condition (1) for a given rectilinear polygon, but only the one correctly simulating the 3-SAT formula satisfies both (1) and (2). So, one may need a more involved way of designing gadgets to achieve a reduction such as ours. It would be surprising if problem B turns out to be polynomial-time solvable.

## References

[1] D. Avis and G. T. Toussaint. An efficient algorithm for decomposing a polygon into star-shaped polygons. *Pattern Recognition*, 13(6):395–398, 1981.

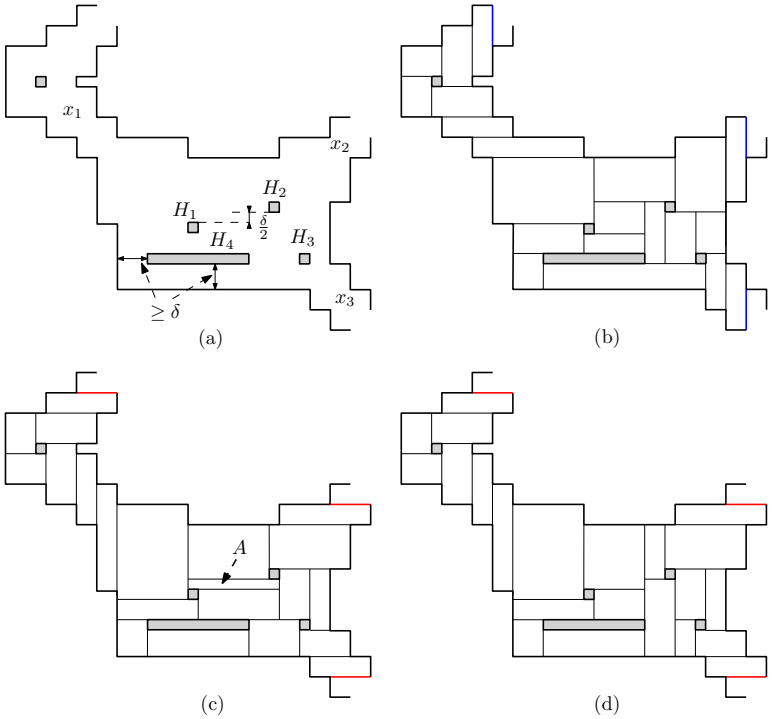



Figure 16: (a) A c-gadget corresponding to clause $(x_1 \vee x_2 \vee x_3)$. (b) A vt-partition of the c-gadget with $x_1 = x_2 = x_3 = 1$. (c) A partition of the c-gadget with $x_1 = x_2 = x_3 = 0$. The rectangle $A$ in the middle has smaller side length $\frac{\delta}{2}$. (d) Another partition of the c-gadget with $x_1 = x_2 = x_3 = 0$. Every rectangle in the partition has smaller side length less than $\delta$, but there are more rectangles in the partition than the minimum among such partitions.


[2] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6(3):485–524, 1991.

[3] J. Choi, S. Cabello, and H.-K. Ahn. Maximizing dominance in the plane and its applications. *Algorithmica*, 2021. Published online at https://link.springer.com/article/10.1007/s00453-021-00863-2.

[4] H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990.

[5] B. Delaunay. Sur la sphère vide. A la mémoire de Georges Voronoï. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, pages 793–800, 1934.

[6] D.-Z. Du, K.-I. Ko, and X. Hu. *Design and Analysis of Approximation Algorithms*. Springer Optimization and Its Applications. Springer New York, 2011.

[7] D.-Z. Du, L.-Q. Pan, and M.-T. Shing. Minimum edge length guillotine rectangular partition. Technical Report 0241886, Mathematical Sciences Research Institute, Univ. California, Berkeley, 1986.

[8] L. Ferrari, P. V. Sankar, and J. Sklansky. Minimal rectangular partitions of digitized blobs. *Computer vision, Graphics, and Image Processing*, 28(1):58–71, 1984.

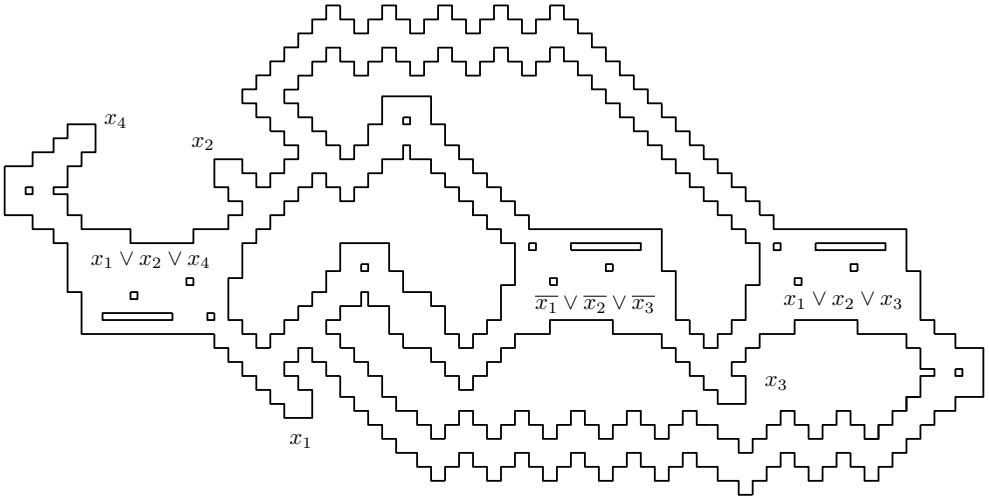



Figure 17: An example of the rectilinear polygon generated by the 3-SAT formula $(x_1 \vee x_2 \vee x_4) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee x_2 \vee x_3)$.


[9] T. Gonzalez and M. Razzazi. On the generalized channel definition problem. In *Proceedings of the 1st Great Lakes Symposium on VLSI*, GLSVLSI 1991, pages 88–91, 1991.

[10] T. Gonzalez and S.-Q. Zheng. Bounds for partitioning rectilinear polygons. In *Proceedings of the 1st Annual Symposium on Computational Geometry*, SoCG 1985, pages 281–287, 1985.

[11] T. Gonzalez and S.-Q. Zheng. Approximation algorithms for partitioning a rectangle with interior points. *Algorithmica*, 5(1):11–42, 1990.

[12] M. T. Goodrich and R. Tamassia. Dynamic ray shooting and shortest paths in planar subdivisions via balanced geodesic triangulations. *Journal of Algorithms*, 23(1):51–73, 1997.

[13] K. Gourley and D. Green. A polygon-to-rectangle conversion algorithm. *IEEE Computer Graphics and Applications*, 3(1):31–36, 1983.

[14] A. Laaksonen. *Guide to competitive programming.* Undergraduate Topics in Computer Science. Springer International Publishing, 2020.

[15] C. Levcopoulos. Fast heuristics for minimum length rectangular partitions of polygons. In *Proceedings of the 2nd Annual Symposium on Computational Geometry*, SoCG 1986, pages 100–108, 1986.

[16] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[17] A. Lingas. Heuristics for minimum edge length rectangular partitions of rectilinear figures. *Theoretical Computer Science*, pages 199–210, 1982.

[18] A. Lingas, R. Y. Pinter, R. L. Rivest, and A. Shamir. Minimum edge length partitioning of rectilinear polygons. In *Proceedings of the 20th Allerton Conference on Communication, Control, and Computing*, Allerton 1982, pages 53–63, 1982.

[19] C. Liu, P. Tu, P. Wu, H. Tang, Y. Jiang, J. Kuang, and E. F. Y. Young. An effective chemical mechanical polishing fill insertion approach. *ACM Transactions on Design Automation of Electronic Systems*, 21(3):1–21, 2016.

[20] M. A. Lopez and D. P. Mehta. Efficient decomposition of polygons into L-shapes with application to VLSI layouts. *ACM Transactions on Design Automation of Electronic Systems*, 1(3):371–395, 1996.

[21] J. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, $k$-MST, and related problems. *SIAM Journal on computing*, 28(4):1298–1309, 1999.

[22] S. A. Mohamed and M. M. Fahmy. Binary image compression using efficient partitioning into rectangular regions. *IEEE Transactions on Communications*, 43(5):1888–1893, 1995.

[23] L. Niu and Y. Song. An automatic solution of accessible information extraction from CityGMLLoD4 files. In *Proceedings of the 21st International Conference on Geoinformatics*, Geoinformatics 2013, pages 1–6, 2013.

[24] J. O'Rourke and G. Tewari. The structure of optimal partitions of orthogonal polygons into fat rectangles. *Computational Geometry*, 28(1):49–71, 2004.

[25] R. L. Rivest. The "PI"(placement and interconnect) system. In *Proceedings of the 19th Design Automation Conference*, DAC 1982, pages 475–481, 1982.

[26] M. Sato and T. Ohtsuki. Applications of computational geometry to VLSI layout pattern design. *Integration*, 5(3-4):303–317, 1987.

[27] T. Suk, C. Höschl, and J. Flusser. Decomposition of binary images–a survey and comparison. *Pattern Recognition*, 45(12):4279–4291, 2012.

[28] M. van Kreveld, O. Schwarzkopf, M. de Berg, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 2000.

[29] G. Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire. Recherches sur les parallélloèdres primitifs. *Journal für die Reine und Angewandte Mathematik*, 1908(134):198–287, 1908.

[30] G. Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Premier mémoire. Sur quelques propriétés des formes quadratiques positives parfaites. *Journal für die Reine und Angewandte Mathematik*, 1908(133):97–102, 1908.