Online Hitting Set for Axis-Aligned Squares

Minati De
Department of Mathematics, Indian Institute of Technology Delhi, New Delhi, India

Satyam Singh
Department of Computer Science, Aalto University, Espoo, Finland

Csaba D. Tóth
Department of Mathematics, California State University Northridge, Los Angeles, CA, USA
Department of Computer Science, Tufts University, Medford, MA, USA

Abstract

Given a set P of n points in the plane and a sequence of axis-aligned squares that arrive in an online fashion, the online hitting set problem consists of maintaining, by adding new points from P if necessary, a hitting set $H \subseteq P$, which contains at least one point in every input square that has already arrived. We present an $O(\log n)$-competitive deterministic algorithm for this problem. The competitive ratio is the best possible, apart from constant factors. In fact, this is the first $O(\log n)$-competitive algorithm for the online hitting set problem that works for geometric objects of arbitrary sizes (i.e., unbounded scaling factors) in the plane. We further generalize this result to positive homothets of a polygon with $k \geq 3$ vertices in the plane and provide an $O(k^2 \log n)$-competitive algorithm.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Online algorithms

Keywords and phrases axis-aligned squares, hitting set, homothets of a polygon, online algorithm

Digital Object Identifier 10.4230/LIPIcs.SWAT.2026.16

Related Version *Previous Version*: https://arxiv.org/pdf/2510.23107

Funding *Satyam Singh*: Research on this paper was supported by the Research Council of Finland, Grant 363444.
Csaba D. Tóth: Research on this paper was supported, in part, by the NSF award DMS-2154347.

Acknowledgements We thank an anonymous reviewer for a suggestion that helped improve the competitive ratio in Theorem 1 from $O(\varrho \log n)$ to $O((1 + \log \varrho) \log n)$.

1 Introduction

The ***minimum hitting set*** problem is one of Karp's 21 classic NP-hard problems [24]. Given a set P of elements and a collection $\mathcal{C} = \{S_1, \ldots, S_m\}$ of subsets of P, referred to as ***ranges***, we need to find a set $H \subseteq P$ (***hitting set***) of minimal size such that every set $S_i \in \mathcal{C}$ contains at least one element of H. Motivated by numerous applications in VLSI design, resource allocation, and wireless networks, researchers have extensively studied the problem for geometric range spaces. In the ***geometric hitting set*** problem, we have $P \subseteq \mathbb{R}^d$ for some constant dimension d, and the sets in $\mathcal{C}$ are geometric objects of some type, such as balls, unit balls, simplices, hypercubes, or hyper-rectangles. Note that the minimum hitting set problem is dual to the minimum set cover problem in the abstract setting, but this duality in general does not extend to the geometric setting.

In this paper, we study the online hitting set problem for geometric objects. In the ***online geometric hitting set*** problem, the point set P is known in advance, while the objects in $\mathcal{C}$ arrive one at a time (without prior knowledge). We need to maintain a hitting set $H_i \subseteq P$ for the first i objects for all $i \geq 1$. Importantly, in the online model, points may be added to

the hitting set as new objects arrive, but they cannot be removed (i.e., $H_i \subseteq H_j$ for $i \leq j$). Upon the arrival of a new object $S_i \in \mathcal{C}$, any number of points can be added to the hitting set. Depending on whether P is finite [18, 20, 23, 26] or infinite [2, 14, 17, 19, 21, 22], there are different versions of the online geometric hitting set problem. In this paper, we consider P to be a finite set of points in $\mathbb{R}^2$.

The quality measure for online algorithms is the competitive ratio, which quantifies the loss of performance due to irrevocable decisions made based on incomplete information. Let ALG be an algorithm for the online hitting set problem on an instance $(P, \mathcal{C})$. The ***competitive ratio*** of ALG, denoted by $\rho(\mathsf{ALG})$, is the supremum, over all possible input sequences σ, of the ratio between the size $\mathsf{ALG}(\sigma)$ of the hitting set obtained by the online algorithm and the minimum size $\mathsf{OPT}(\sigma)$ of a hitting set for the same input:

$$\rho(\mathsf{ALG}) = \sup_{\sigma} \left[\frac{\mathsf{ALG}(\sigma)}{\mathsf{OPT}(\sigma)} \right].$$

1.1 Related Previous Work

Alon et al. [3] initiated the study of the online hitting set problem and presented a deterministic algorithm with a competitive ratio of $O(\log |P| \log |\mathcal{C}|)$ and obtained an almost matching lower bound of $\Omega\left(\frac{\log |P| \log |\mathcal{C}|}{\log \log |P| + \log \log |\mathcal{C}|}\right)$. While their work addresses the general setting, Even and Smorodinsky [23] initiated the study of the online geometric hitting set problem for various geometric objects. They established an optimal competitive ratio of $\Theta(\log |P|)$ when P is a finite subset of $\mathbb{R}$, and the objects are intervals in $\mathbb{R}$. They also established an optimal competitive ratio of $\Theta(\log |P|)$ when P is a finite subset of $\mathbb{R}^2$, and the objects are half-planes or congruent disks in the plane.

Later, Khan et al. [26] examined the problem for the special case of a finite set of ***integer*** points $P \subseteq [0, N)^2 \cap \mathbb{Z}^2$ and a collection $\mathcal{C}$ of axis-aligned squares $S \subseteq [0, N)^2$ with integer coordinates, for $N > 0$. They developed an $O(\log N)$-competitive deterministic algorithm for this variant. They also established a randomized lower bound of $\Omega(\log |P|)$, where $P \subset \mathbb{R}^2$ is finite and $\mathcal{C}$ consists of congruent axis-aligned squares. De et al. [18, 20] further investigated the problem for an arbitrary finite set $P \subset \mathbb{R}^2$, where $\mathcal{C}$ consists of homothetic geometric objects with scaling factors (e.g., diameters) in the interval $[1, M]$ for some parameter $M > 0$. In [18], they considered homothetic copies of a regular k-gon (for $k \geq 4$) and developed a randomized algorithm with expected competitive ratio $O(k^2 \log M \log |P|)$. Although regular k-gons can approximate disks as $k \to \infty$, this result does not imply a competitive algorithm for disks with radii in $[1, M]$. In [20], they addressed this gap by presenting an $O(\log M \log |P|)$-competitive deterministic algorithm for homothetic disks, and further generalized their result to positive homothets of any convex body in the plane with scaling factors in $[1, M]$.

However, it remained an open problem whether the dependence on the parameter M is necessary, or whether there is an $O(\log |P|)$-competitive online hitting set algorithm for arbitrary homothets of a polygon or for arbitrary disks in the plane. In fact, the full version of Khan et al. [26] posed this as an open problem specifically for axis-aligned squares.

1.2 Our Contribution

We present the first $O(\log n)$-competitive algorithm for the online hitting set problem for a set P of $n = |P|$ points and geometric objects of arbitrary sizes in the plane; Table 1 summarizes both previous and new results. Our algorithm works for axis-aligned squares of

arbitrary sizes, and it generalizes to axis-aligned rectangles of bounded aspect ratio. The ***aspect ratio*** of a rectangle is the ratio of the length of the longer to that of the shorter side (e.g., the aspect ratio of a square is 1, and the aspect ratio of a 1×2 or a 2×1 rectangle is 2).

Table 1 Summary of known and new results for the geometric online hitting set problem where $|P| = n$ for some $n \in \mathbb{N}$. (#) indicates lower bounds for randomized algorithms. The results presented in this paper are listed in the last three rows.

Finite Point Set	Objects	Lower Bound	Upper Bound
$P \subset \mathbb{R}$	Intervals in $\mathbb{R}$	$\Omega(\log \lvert P\rvert)$ [23]	$O(\log \lvert P\rvert)$ [23]
$P \subset \mathbb{R}^2$	Half-planes in $\mathbb{R}^2$	$\Omega(\log \lvert P\rvert)$ [23]	$O(\log \lvert P\rvert)$ [23]
$P \subset \mathbb{R}^2$	Congruent disks in $\mathbb{R}^2$	$\Omega(\log \lvert P\rvert)$ [23]	$O(\log \lvert P\rvert)$ [23]
$P \subseteq [0, N)^2 \cap \mathbb{Z}^2$	Axis-aligned squares with integral vertices	$\Omega(\log \lvert P\rvert)$ [26] (#)	$O(\log N)$ [26]
$P \subseteq [0, N)^2 \cap \mathbb{Z}^2$	Bottomless rectangles (of the form $[a, b] \times (-\infty, c]$)	$\Omega(\log \lvert P\rvert)$ [23]	$O(\log N)$ [20]
$P \subset \mathbb{R}^2$	Positive homothets of an arbitrary convex body in $\mathbb{R}^2$ with scaling factors in $[1, M]$	$\Omega(\log \lvert P\rvert)$ [26] (#)	$O(\log M \log \lvert P\rvert)$ [20]
$P \subset \mathbb{R}^2$	Axis-aligned squares	$\Omega(\log \lvert P\rvert)$ [26] (#)	$O(\log \lvert P\rvert)$ [Theorem 1]
$P \subset \mathbb{R}^2$	Axis-aligned rectangles of aspect ratio at most $\varrho \geq 1$	$\Omega(\log \lvert P\rvert)$ [26] (#)	$O((1 + \log \varrho) \log \lvert P\rvert)$ [Theorem 1]
$P \subset \mathbb{R}^2$	Positive homothets of a polygon with $k \geq 3$ vertices	$\Omega(\log \lvert P\rvert)$ [26] (#)	$O(k^2 \log \lvert P\rvert)$ [Theorem 2]

▶ **Theorem 1.** *For every $\varrho \geq 1$, there is an $O((1 + \log \varrho) \log n)$-competitive deterministic algorithm for the online hitting set problem for any set of n points in the plane and a sequence of axis-aligned rectangles of aspect ratio at most ϱ.*

We further generalize Theorem 1 to positive homothets of a polygon.

▶ **Theorem 2.** *Let M be a polygon with $k \geq 3$ vertices. Then there is an $O(k^2 \log n)$-competitive deterministic algorithm for the online hitting set problem for any set of n points in the plane, and a sequence of positive homothets of M.*

The previous best competitive ratio for these problems was $O(\log^2 n)$, by Alon et al. [3], which holds more generally for any collection of sets of polynomial size, including any collection of geometric objects of bounded VC-dimension. As noted above, De et al. [18] designed an $O(k^2 \log M \log n)$-competitive randomized algorithm for homothets of regular k-gons with scaling factors in the range $[1, M]$. Theorem 2 provides a deterministic algorithm for homothets of arbitrary convex k-gons of arbitrary sizes. However, it is unclear whether the dependence on k is necessary. Even and Smorodinsky [23] asked whether there is an online hitting set algorithm with an $O(\log n)$-competitive ratio for any set system on n points with bounded VC-dimension. Khan et al. [26] asked whether there is an $O(\log n)$-competitive algorithm for the online hitting set problem with squares, as their algorithm is restricted to integer points in $[0, N)^2 \cap \mathbb{Z}^2$ and it is $O(\log N)$-competitive even if $|P| \ll N$. Our

result (Theorem 1) gives an affirmative answer to their question. It is unclear whether $O(\log n)$-competitive online algorithms are possible for any other families of geometric set systems. In Section 6, we briefly discuss roadblocks to possible generalizations to disks (of arbitrary radii) in $\mathbb{R}^2$ or to cubes in $\mathbb{R}^3$, as well as possible extensions to the dual problem of online geometric set cover.

Technical highlights. We present the key ideas for axis-aligned squares (of arbitrary size), since all technical tools readily generalize to axis-aligned rectangles of bounded aspect ratio. Recall that the point set $P \subset \mathbb{R}^2$ is given in advance, that axis-aligned squares $\mathcal{C} = \{S_1, S_2 \ldots\}$ arrive one-by-one, and that we need to construct a hitting set $H \subset P$ of size $|H| \leq O(|\mathsf{OPT}| \log n)$ where $\mathsf{OPT} \subset P$ is an optimal hitting set for $\mathcal{C}$. Each square $S_i \in \mathcal{C}$ is hit by an unknown point $p \in \mathsf{OPT}$ in the offline optimum; we would like to add points to H that also hit many other potential squares that contain (the unknown) point p. Our general strategy is to preprocess P into a hierarchy of depth $O(\log n)$, and then use this hierarchy to narrow down the possible location of the points in OPT. In one dimension, where $P \subset \mathbb{R}$ and $\mathcal{C}$ is a sequence of intervals, a simple binary hierarchy is sufficient [23].

In the plane, we preprocess $P \subset \mathbb{R}^2$ with the classical BBD tree data structure by Arya et al. [5], which has found a wide range of applications in computational geometry, including binary space partitions, range searching, and many other problems [1, 4, 5, 11, 25]; this data structure was previously used for the online geometric set cover problem by Khan et al. [26]. For a set $P \subset \mathbb{R}^2$ of n points, the BBD tree is a hierarchical space partition of depth $O(\log n)$, where all cells of the partition are "fat" (in the sense defined below). When a square S_i arrives online, we choose hitting points via a top-down traversal of the BBD tree. Specifically, we "activate" the $O(1)$ lowest inactive cells of the BBD tree that intersect S_i (see Section 4 for details), and add $O(1)$ extremal points in each of these cells to H. The extremal points of a rectangular cell C are simply the leftmost, rightmost, topmost, and bottommost points in $P \cap C$ (i.e., extremal with respect to axis-parallel directions). Extremal points have previously been used in computational geometry for many different problems, including guarding set or shortest path problems [8, 10, 16, 30], as well as restricted versions of the online hitting set problem in combination with quadtrees [26].

Unfortunately, the cells in the BBD tree are not necessarily convex – they include rectangles with a rectangular hole – so we need to define extremal points much more carefully. The intersection $C_v \cap S_i$ of a cell C_v of the BBD tree and a square S_i can be reduced to axis-aligned half-spaces, and we show that $C_v \cap S_i \cap P$ is either empty or contains at least one extremal point of C_v (see Section 3.2 for details). Note that these techniques work for arbitrary axis-aligned fat rectangles (the fatness of a rectangle is quantified by the aspect ratio ϱ): extremal points in the four axis-parallel directions are sufficient if S_i is an axis-aligned rectangle, and the intersection $C_v \cap S_i$ reduces to halfplanes if both C_v and S_i are fat.

Each (unknown) point $p \in \mathsf{OPT}$ is contained in $O(\log n)$ cells of the BBD tree. After $O(\log n)$ steps, in which a square S_i arrives that contains the point $p \in \mathsf{OPT}$ but our algorithm adds $O(1)$ new points to H, the BBD tree is "saturated", in the sense that all cells containing p are activated – at this point, the extremal points in H already hit any subsequent square S_i that contains p. We prove that our algorithm uses only $O(\log n)$ points for any sequence of squares that can be optimally hit by a single point; see Section 4.2 for further details.

Relation to online set cover for squares. Consider the ***online geometric set cover*** problem: given a set $\mathcal{S}$ of n geometric objects in $\mathbb{R}^d$, and a sequence of points arriving one by one, we need to maintain a set cover $C_i \subset \mathcal{S}$ for the first i points in an online fashion.

Note, however, that the hitting set and the set cover problems are not equivalent in the geometric setting: points in the plane form a 2-parameter family (parameterized by x- and y-coordinates), while axis-aligned squares form a 3-parameter family (parameterized by, e.g., a lower-left corner and a side length). In the online hitting set problem, the 2-parameter family is known in advance, and the 3-parameter family arrives without prior knowledge – the roles are reversed in online set cover. That is, an online hitting set algorithm makes decisions based on less information (i.e., more uncertainty) than an analogous online set cover algorithm. Importantly, these problems are not equivalent, and techniques developed for one problem do not easily carry over to the other. Khan et al. [26] studied a restricted version of the online set cover problem for axis-aligned squares. In their model, both a collection $\mathcal{S}$ of m axis-aligned squares and a set $\tilde{P}$ of n points are given in advance. Points from $\tilde{P}$ arrive online, and we need to maintain a set of squares $C_i \subset \mathcal{S}$ that cover the first i points. Using a BBD tree decomposition over the point set $\tilde{P}$, they obtain a deterministic $O(\log n)$-competitive algorithm.

We remark that their result yields an $O(\log m)$-competitive algorithm for the ***unrestricted*** version of the online set cover problem for axis-aligned squares (where $\tilde{P}$ is not given in advance). An arrangement of m squares in the plane determines $O(m^2)$ cells: points in the same cell are contained in the same subset of squares (they are combinatorially indistinguishable). We can define $\tilde{P}$ as a set of $O(m^2)$ representative points, one from each cell. When a sequence of arbitrary points $p_1, p_2, \ldots$ arrive online, we can replace each point p_i with the representative of its cell, and invoke the algorithm by Khan et al. [26], with competitive ratio $O(\log |\tilde{P}|) = O(\log m^2) = O(\log m)$.

Organization. Section 2 begins by introducing the necessary definitions and then reviews a classical space partition data structure, the Balanced Box Decomposition Tree (BBD tree) [6]. Section 3 presents several key properties of BBD trees. Section 4 describes our online algorithm for hitting axis-aligned rectangles, and analyzes its competitive ratio. Next, Section 5 generalizes the main result from axis-aligned squares to positive homothets of an arbitrary polygon. Finally, Section 6 concludes with a discussion of future research directions.

2 Notation and Preliminaries

Unless stated otherwise, the term ***object*** refers to a compact set in $\mathbb{R}^d$ with a nonempty interior. Let σ denote such an object. For a scaling parameter $\lambda \in \mathbb{R}$ and a translation vector $b \in \mathbb{R}^d$, the set $\lambda\sigma + b = \{\lambda x + b : x \in \sigma\}$ is called a ***homothet*** or ***homothetic copy*** of σ; and it is a ***positive homothet*** if $\lambda > 0$.

BBD Trees. Arya et al. [6] introduced the ***Balanced Box Decomposition Tree*** (***BBD tree***, for short), which is a binary space partition tree for a set of n points in $\mathbb{R}^d$. Since its introduction in the 1990s, BBD trees have become a widely used data structure for processing and classifying spatial data in computational geometry and related fields. In contrast to the quadtree (or compressed quadtree), the depth of the BBD tree is $O(\log n)$, and the nodes correspond to "fat" regions; the precise definition is given below.

For a set P of n points in an axis-aligned square (the bounding box), the BBD tree is a binary tree T, where the nodes correspond to regions, called ***cells*** (with the root corresponding to the bounding box). The parent-child relation corresponds to containment between the corresponding cells with the following properties:

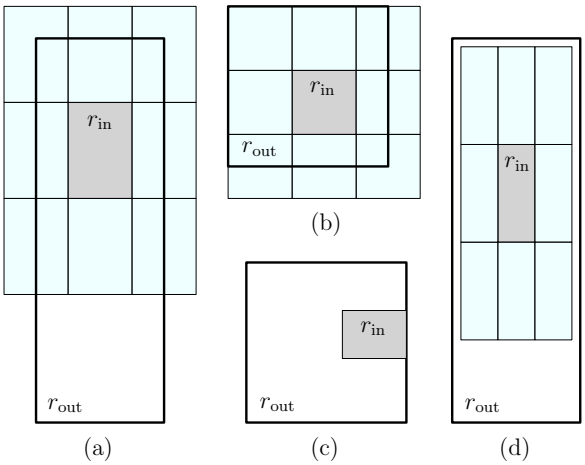

Figure 1 The rectangle r_{in} is not sticky for the rectangle r_{out} in (a) and (b), and is sticky in (c) and (d).

- Each node $v \in V(T)$ corresponds to a ***cell*** $C_v = r_{\text{out}}(v) \setminus r_{\text{in}}(v)$, where $r_{\text{in}}(v)$ and $r_{\text{out}}(v)$ are axis-aligned rectangles such that $r_{\text{in}}(v) \subset r_{\text{out}}(v)$, and where possibly $r_{\text{in}}(v) = \emptyset$.
- the aspect ratio of $r_{\text{out}}(v)$ and $r_{\text{in}}(v)$ (if $r_{\text{in}} \neq \emptyset$) is at most 3;
- if $r_{\text{in}}(v) \neq \emptyset$, then it is ***sticky***, which means that the vertical (respectively, horizontal) distance between $r_{\text{in}}(v)$ and the boundary of $r_{\text{out}}(v)$ is either 0 or at least the side length of $r_{\text{in}}(v)$. An equivalent condition for stickiness can be obtained by considering the regular grid consisting of 3^2 translated copies of $r_{\text{in}}(v)$, centered around $r_{\text{in}}(v)$. The rectangle $r_{\text{in}}(v)$ is sticky for $r_{\text{out}}(v)$ if and only if every copy of $r_{\text{in}}(v)$ in this grid either lies entirely within $r_{\text{out}}(v)$ or is disjoint from the interior of $r_{\text{out}}(v)$, see Figure 1;
- the cells $\{C_v : v \in V(T)\}$ form a laminar set system, that is, if u is a descendant of v, then $C_u \subset C_v$, otherwise $\text{int}(C_u) \cap \text{int}(C_v) = \emptyset$;
- for each leaf node $v \in V(T)$, the region C_v contains at most one point of P.

Each internal node has exactly two children, generated by one of two operations: a ***fair split***, which decomposes a cell C_v along an axis-parallel line into two cells, or a ***shrink***, which introduces a new box R such that $r_{\text{in}}(v) \subseteq R \subseteq r_{\text{out}}(v)$ and decomposes $C_v = r_{\text{out}}(v) \setminus r_{\text{in}}(v)$ into $r_{\text{out}}(v) \setminus R$ and $R \setminus r_{\text{in}}(v)$. Furthermore, the number of nodes in T is $O(n)$, the depth of T is $O(\log n)$, and the entire structure can be constructed in $O(n \log n)$ time [6].

3 A Rectangle amid Cells of the BBD Tree

In this section, we present some important properties of BBD trees and discuss the relative position of an arbitrary axis-aligned rectangle of bounded aspect ratio with respect to the cells of the BBD tree. These properties play a crucial role in the design and analysis of our algorithms in Sections 4 and 5.

3.1 Crossing Between Rectangles and Cells of the BBD Tree

Let T be a BBD tree for a finite point set P. We say that an axis-aligned rectangle R ***crosses*** a cell C_v, $v \in V(T)$, if $C_v \cap R \neq \emptyset$ but C_v does not contain any vertex of R and R does not contain any vertex of C_v (i.e., vertices of $r_{\text{out}}(v)$ and vertices of $r_{\text{in}}(v)$ if any). See Figure 2 for examples.

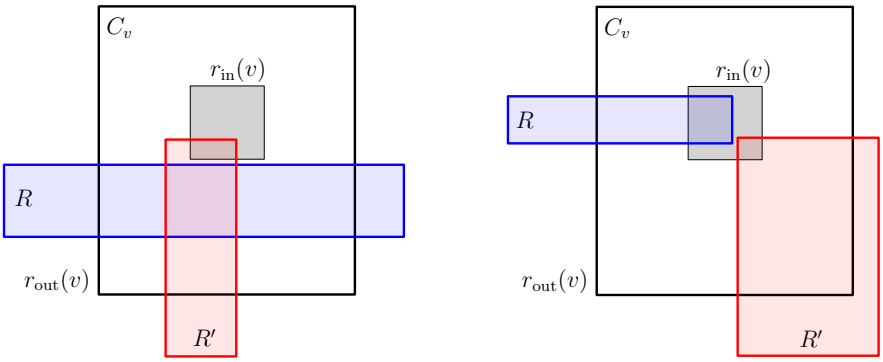

Figure 2 In both (a) and (b), the rectangle R crosses the cell C_v, but R' does not.

▶ **Lemma 3.** *If R is an axis-aligned rectangle of aspect ratio at most ϱ, for some $\varrho \geq 1$, then it crosses $O(\varrho)$ interior-disjoint cells of a BBD tree.*

Proof. Let T be a BBD tree, and let $U \subset V(T)$ be a set of nodes that corresponds to a family of interior-disjoint cells crossed by R. Let $v \in U$. Recall that a cell of the BBD tree is defined as $C_v = r_{\text{out}}(v) \setminus r_{\text{in}}(v)$, where $r_{\text{in}}(v)$ may be empty. Depending on how R intersects with C_v, we distinguish between two cases.

Case 1: R crosses $C_v = r_{\text{out}}(v) \setminus r_{\text{in}}(v)$ and R intersects two opposite edges of $r_{\text{out}}(v)$. Assume w.l.o.g. that R intersects two vertical edges of $r_{\text{out}}(v)$ (i.e., R crosses $r_{\text{out}}(v)$ horizontally); see Figure 2(a). Since the aspect ratios of R and r_{out} are ϱ and at most 3, respectively, then we have

$$\text{width}(R) \leq \varrho \cdot \text{height}(R) < \varrho \cdot \text{height}(r_{\text{out}}(v)) \leq 3\varrho \cdot \text{width}(r_{\text{out}}(v)).$$

In particular, this implies $\frac{1}{3\varrho}\text{width}(R) < \text{width}(r_{\text{out}}(v))$. Note also that $\text{width}(R \cap r_{\text{out}}(v)) = \text{width}(r_{\text{out}}(v))$. If R horizontally crosses k interior-disjoint rectangles $r_{\text{out}}(v_1), \ldots, r_{\text{out}}(v_k)$, then

$$\frac{k}{3\varrho}\text{width}(R) < \sum_{i=1}^{k} \text{width}(r_{\text{out}}(v_i)) \leq \text{width}(R),$$

hence $k \leq 3\varrho$. That is, R horizontally crosses at most 3ϱ interior-disjoint cells. Similarly, R may cross at most 3ϱ cells vertically. However, if R crosses a cell C_v horizontally and another cell C_w vertically, then $\text{int}(C_v) \cap \text{int}(C_w) \neq \emptyset$. In particular, U cannot contain both v and w. Overall, R crosses at most 3ϱ interior-disjoint cells in this case.

Case 2: R crosses $C_v = r_{\text{out}}(v) \setminus r_{\text{in}}(v)$ and R intersects a pair of parallel edges in $r_{\text{out}}(v)$ and $r_{\text{in}}(v)$, respectively. Assume w.l.o.g. that R intersects the left side of both $r_{\text{in}}(v)$ and $r_{\text{out}}(v)$; see Figure 2(b). Let $\text{dist}_{\text{left}}(v)$ denote the distance between the left sides of r_{in} and r_{out}. Then we have $\text{width}(R \cap C_v) = \text{dist}_{\text{left}}(v)$, and in particular $\text{dist}_{\text{left}}(v) < \text{width}(R)$. Due to the stickiness, we also have $\text{width}(r_{\text{in}}) \leq \text{dist}_{\text{left}}(v)$. Overall, we obtain

$$\text{width}(R) \leq \varrho \cdot \text{height}(R) < \varrho \cdot \text{height}(r_{\text{in}}(v)) \leq 3\varrho \cdot \text{width}(r_{\text{in}}(v)) \leq 3\varrho \cdot \text{dist}_{\text{left}}(v).$$

By the pigeonhole principle, R crosses at most 3ϱ interior-disjoint cells between the left sides of r_{in} and r_{out}. Similarly, R crosses at most 3ϱ interior-disjoint cells between the right (respectively, top, bottom) sides of r_{in} and r_{out}. As a result, R crosses $O(\varrho)$ interior-disjoint cells in this case. ◀

3.2 Extremal Points and their Properties

For each node v of the BBD tree, we define a set Ext_v consisting of a constant number of ***extremal points*** in P.

- For an axis-aligned rectangle r, let $\text{ext}(r)$ be a subset of $P \cap r$ that consists of a point with the minimum x-coordinate, maximum x-coordinate, minimum y-coordinate, and maximum y-coordinate (ties are broken arbitrarily).
- If $r_{\text{in}}(v) = \emptyset$ (that is, $C_v = r_{\text{out}}(v)$), then let $\text{Ext}_v = \text{ext}(r_{\text{out}}(v))$; see Figure 3(a).
- If $r_{\text{in}}(v) \neq \emptyset$, then we subdivide $C_v = r_{\text{out}}(v) \setminus r_{\text{in}}(v)$ along the lines spanned by the four sides of $r_{\text{in}}(v)$ into k, $2 \leq k \leq 9$, rectangular regions $C_v = \bigcup_{i=1}^{k} r_i$ and let $\text{Ext}_v = \bigcup_{i=1}^{k} \text{ext}(r_i)$; see Figure 3(b).

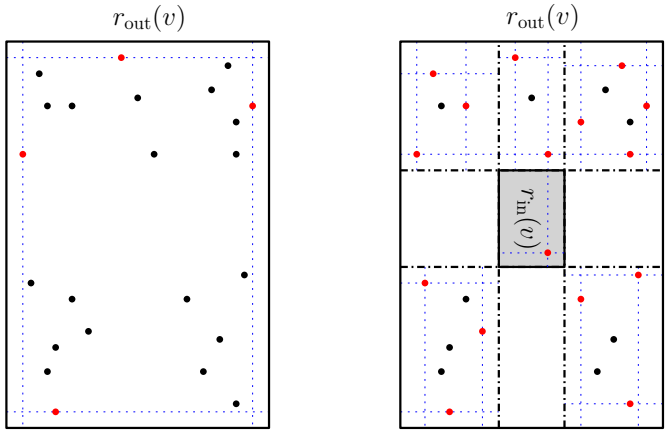

Figure 3 The extremal points Ext_v of the cell C_v are red, when (a) $r_{\text{in}}(v) = \emptyset$; (b) $r_{\text{in}}(v) \neq \emptyset$.

Properties of extremal points. We state a few properties of extremal points that will be used in the competitive analysis of our online algorithm in Section 4. Let P be a finite set of points in a bounding box (a square), and T be a BBD tree for P.

▶ **Lemma 4.** *Let r be an axis-aligned rectangle, and L^- be a half-plane bounded by an axis-parallel line L. If $P \cap r \cap L^- \neq \emptyset$, then $\text{ext}(r) \cap L^- \neq \emptyset$.*

Proof. Assume w.l.o.g. that L is the vertical line $x = a$, for some $a \in \mathbb{R}$, and let $L^- = \{(x, y) \in \mathbb{R}^2 : x \leq a\}$ be the left half-plane; see Figure 4(a). We prove the contrapositive. If $\text{ext}(r) \cap L^- = \emptyset$, then every point $p = (p_x, p_y) \in \text{ext}(P)$ satisfies $p_x > a$. In particular, we have $p_x > a$ for the leftmost point in $P \cap r$, hence we have $p_x > a$ for all points in $P \cap r$. This implies that $P \cap r \cap L^- = \emptyset$, as required. ◀

▶ **Lemma 5.** *Let $C_v = r_{\text{out}}(v) \setminus r_{\text{in}}(v)$ be a cell (where $r_{\text{in}}(v)$ may be empty), and L^- be a half-plane bounded by an axis-parallel line L. If $P \cap C_v \cap L^- \neq \emptyset$, then $\text{Ext}_v \cap L^- \neq \emptyset$.*

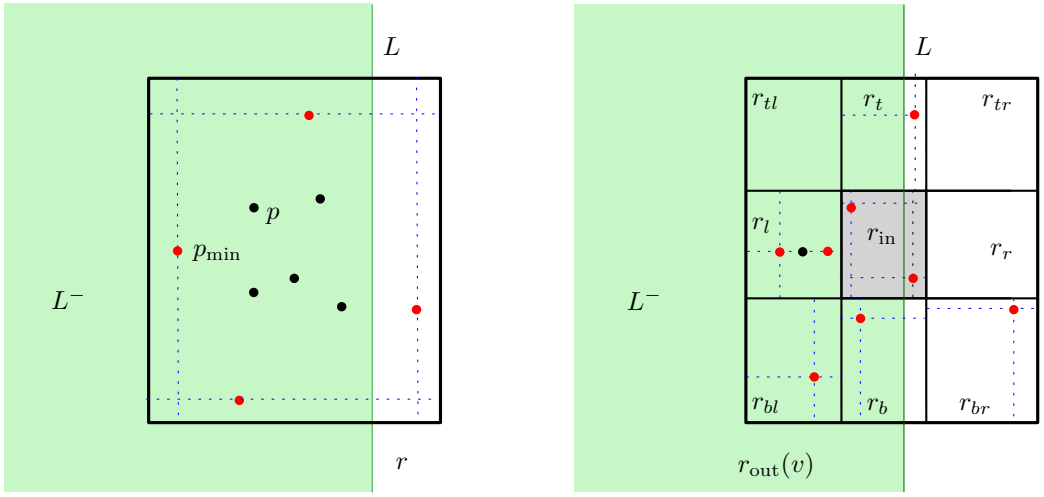

Figure 4 Illustration for (a) Lemma 4, and (b) Lemma 5.

Proof. Assume w.l.o.g. that L is the vertical line $x = a$ for some $a \in \mathbb{R}$, and let $L^- = \{(x, y) \in \mathbb{R}^2 : x \leq a\}$; see Figure 4(b) for an illustration. We prove the contrapositive. Suppose that $\text{Ext}_v \cap L^- = \emptyset$. Then every point of Ext_v has x-coordinate larger than a. Recall that $C_v = \bigcup_{i=1}^{k} r_i$. The leftmost point of $P \cap C_v$ is the leftmost point of $P \cap r_i$ for some $1 \leq i \leq k$. This point is included in $\text{ext}(r_i)$, and hence in $\text{Ext}_v = \bigcup_{i=1}^{k} \text{ext}(r_i)$. Consequently, Ext_v contains the leftmost point of $P \cap C_v$. Then, all points of $P \cap C_v$ have x-coordinate larger than a. Hence, we have $P \cap C_v \cap L^- = \emptyset$, as required. ◀

▶ **Lemma 6.** *Let $C_v = r_{\text{out}}(v) \setminus r_{\text{in}}(v)$ be a cell, and $S = L_1^- \cap L_2^+$ denote the intersection of two half-planes bounded by two horizontal lines or two vertical lines L_1 and L_2. If $P \cap C_v \cap S \neq \emptyset$ and S contains a corner of $r_{\text{in}}(v)$, then $\text{Ext}_v \cap S \neq \emptyset$.*

Proof. Assume w.l.o.g. that L_1 and L_2 are the vertical lines $x = a$ and $x = b$, respectively, for some $a, b \in \mathbb{R}$; see Figure 5 for an illustration. Let $L_1^- = \{(x, y) \in \mathbb{R}^2 : x \leq a\}$ be the left half-plane. If L_2^+ is also a left half-plane, then $L_1^- \cap L_2^+$ is a half-plane and Lemma 5 completes the proof. So we may assume that $L_2^+ = \{(x, y) \in \mathbb{R}^2 : x \geq b\}$ is the right half-plane, and $b < a$ (since $S = L_1^- \cap L_2^+ \neq \emptyset$).

Assume that S contains a corner $c = (c_1, c_2)$ of $r_{\text{in}}(v)$. Recall that C_v is subdivided along the lines spanned by the sides of $r_{\text{in}}(v)$. One of the subdivision lines is $x = c_1$, where $b < c_1 < a$ (since $c \in S$). Consequently, every subrectangle of C_v intersects at most one of L_1 and L_2.

Since $P \cap C_v \cap S \neq \emptyset$, there exists a point $p \in P \cap C_v \cap S$. Assume that $p \in r_x$, where r_x is one of the subrectangles of C_v. Since $\text{ext}(r_x) \cap S \subseteq \text{Ext}_v \cap S$ and by Lemma 4, we have $\text{ext}(r_x) \cap S \neq \emptyset$, thus we have $\text{Ext}_v \cap S \neq \emptyset$. Hence, the lemma follows. ◀

▶ **Lemma 7.** *Let $C_v = r_{\text{out}}(v) \setminus r_{\text{in}}(v)$ be a cell, and let $S = L_1^- \cap L_2^+$ be the intersection of two half-planes bounded by two perpendicular axis-parallel lines L_1 and L_2. If $P \cap C_v \cap S \neq \emptyset$ and $L_1 \cap L_2 \in r_{\text{in}}(v)$, then $\text{Ext}_v \cap S \neq \emptyset$.*

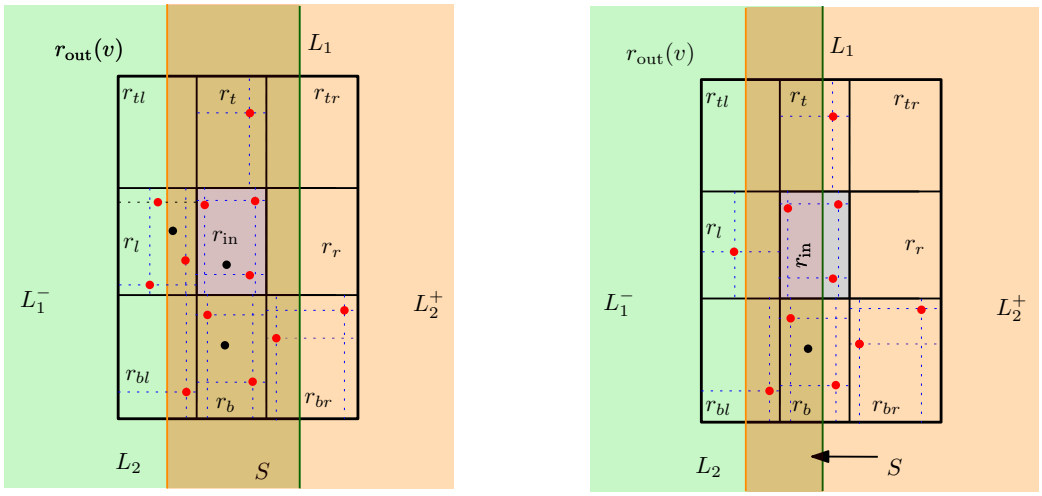

Figure 5 Illustration for Lemma 6.

Proof. Assume w.l.o.g. that L_1 is the vertical line $x = a$, and L_2 is the horizontal line $y = b$ for some $a, b \in \mathbb{R}$; see Figure 6 for an illustration. Let $L_1^- = \{(x, y) \in \mathbb{R}^2 : x \leq a\}$ be the left half-plane and $L_2^+ = \{(x, y) \in \mathbb{R}^2 : y \geq b\}$ be the top half-plane; see Figure 6. Since $L_1 \cap L_2 = (a, b) \in r_{\text{in}}(v)$, then L_1 intersects the subrectangles of C_v above and below r_{in}(v); and L_2 intersects the subrectangles of C_v to the left and right of $r_{\text{in}}(v)$. In particular, none of the subrectangles in C_v intersects both L_1 and L_2.

Since $P \cap C_v \cap S \neq \emptyset$, there exists a point $p \in P \cap C_v \cap S$. Assume that $p \in r_x$, where r_x is one of the subrectangles of C_v. Since r_x intersects at most one of L_1 and L_2, then $r_x \cap S$ equals $r_x \cap L_1^-$ or $r_x \cap L_2^+$. In both cases, by Lemma 4, we have $\text{ext}(r_x) \cap S \neq \emptyset$. Now $\text{ext}(r_x) \subseteq \text{Ext}_v$ implies that $\text{Ext}_v \cap S \neq \emptyset$, as required. ◀

4 Online Algorithm and Competitive Analysis

In this section, we first prove Theorem 8. In Section 4.1, we present our online hitting set algorithm for axis-aligned rectangles of aspect ratio ϱ, for constant $\varrho \geq 1$. In Section 4.2, we analyze its competitive ratio, and then use this algorithm as a subroutine to prove Theorem 1.

▶ **Theorem 8.** *For every $\varrho \geq 1$, there is an $O(\varrho \log n)$-competitive deterministic algorithm for the online hitting set problem for any set of n points in the plane and a sequence of axis-aligned rectangles of aspect ratio at most ϱ.*

4.1 Online Algorithm

We now describe our online algorithm. Given a set P of n points in the plane, we compute a BBD tree T for P, and the set H_v of extremal points $\text{Ext}_v \subset P$ for all nodes $v \in V(T)$. As the adversary presents a sequence of axis-aligned rectangles $\{S_1, \ldots, S_m\}$ of aspect ratio at most ϱ, we maintain the following data structures:

A hitting set $H_i \subset P$ for the first i rectangles $\{S_j : j \leq i\}$, which is initialized to $H_0 = \emptyset$. A set $A_i \subset V(T)$ of ***active*** nodes of T with the following property: If a node $v \in V(T)$ is active, then all ancestors of v are also active. Initially, all nodes are ***inactive*** (i.e., $A_0 = \emptyset$);

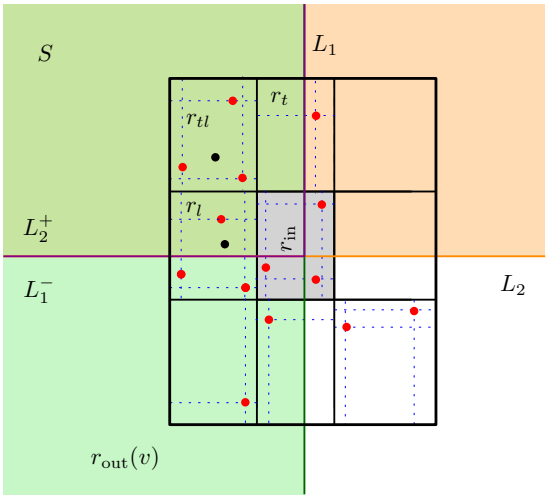

Figure 6 Illustration for Lemma 7.

inactive nodes can become active, and any active node remains active for the remainder of the algorithm. Furthermore, we maintain the property that $\bigcup_{v \in A_i} \text{Ext}_v \subset H_i$, i.e., H_i contains the extremal points of all active nodes (however, H_i may contain additional points as well).

When a rectangle S_i arrives, initialize $H_i := H_{i-1}$ and $A_i := A_{i-1}$. If $S_i \cap H_i \neq \emptyset$, then no further changes are needed in our data structures. Suppose that $S_i \cap H_i = \emptyset$.

1. Consider the four corners of S_i.
   - **a.** For each corner $a \in C_{\text{root}}$, find the highest inactive node $v \in V(T)$ such that $a \in C_v$: activate v and its sibling v' (set $A_i := A_i \cup \{v, v'\}$) and add their extremal points to H_i (set $H_i := H_i \cup \text{Ext}_v \cup \text{Ext}_{v'}$).
   - **b.** For each corner $a \notin C_{\text{root}}$, if the root of T is not already active, then activate the root and add its extremal points to H_i.
2. For every node $u \in V(T)$ where S_i crosses C_u,
   - **a.** If u is already active but its children are not, then activate both children of u and add their extremal points to H_i.
   - **b.** If u is inactive, then find the highest inactive ancestor v of u (possibly $v = u$): activate v and its sibling v' and add their extremal points to H_i.
3. If $S_i \cap H_i = \emptyset$ still holds, then add an arbitrary point in $S_i \cap P$ to H_i.

4.2 Competitive Analysis

The algorithm guarantees that H_i is a hitting set for the first i objects $\{S_1, \ldots, S_i\}$. It is also clear that for each new object S_i, we add $O(\varrho)$ new points to H_i: Since S_i has four corners and crosses $O(\varrho)$ cells of the BBD tree by Lemma 3.

Let $\mathsf{OPT} \subset P$ be an offline optimum, i.e., a minimum hitting set for $\mathcal{C} = \{S_1, \ldots, S_m\}$. For each point $p \in \mathsf{OPT}$, let $\mathcal{C}_p = \{S_j \in \mathcal{C} : p \in S_j\}$ be the set of objects hit by p. It is sufficient to show that for every $p \in \mathsf{OPT}$, the algorithm adds new points to the hitting set in $O(\log n)$ steps to hit objects in $\mathcal{C}_p$ (cf. Corollary 10). Since $O(\varrho)$ points are added to the hitting set in each such step, then the algorithm adds $O(\varrho \log n)$ points in response to the objects in $\mathcal{C}_p$, and hence $O(|\mathsf{OPT}| \cdot \varrho \log n)$ points in response to all objects in $\mathcal{C}$.

▶ **Lemma 9.** *If $S_i \in \mathcal{C}_p$ and $S_i \cap H_{i-1} = \emptyset$, then in step i, the algorithm activates a cell of the BBD tree containing p.*

Proof. The algorithm activates the root in step $i = 1$. In the remainder of the proof, we may assume that $i > 1$ and the root is already active. Before the arrival of S_i, we have $p \notin H_{i-1}$, so the leaf node of the BBD tree that contains p is inactive. Let v be the lowest active node in the BBD tree such that C_v contains p. Let u and w be the two children of v such that $p \in C_u$ (hence $p \notin C_w$).

We need to show that the algorithm activates u in step i. Suppose, for the sake of contradiction, that u is inactive at the beginning of step i. Then its sibling w is also inactive at that time (since our algorithm always activates two siblings).

The algorithm activates the highest inactive nodes that contain any of the four corners of S_i (and their siblings), as well as the highest inactive nodes corresponding to every cell crossed by S_i (and their siblings). Therefore, we may assume that neither C_u nor C_w contains any corner of S_i, and neither of them is crossed by S_i. Since $C_v = C_u \cup C_w$, then C_v does not contain any corner of S_i, either. If S_i crosses C_v, then at the beginning of step i, node v is active but its children u and w are inactive, and so the algorithm would activate both u and w in step i. For this reason, we may also assume that S_i does not cross C_v.

We examine all possible positions of C_v and C_u relative to S_i. Recall that $C_v = r_{\text{out}}(v) \setminus r_{\text{in}}(v)$, where $r_{\text{in}}(v)$ may be empty.

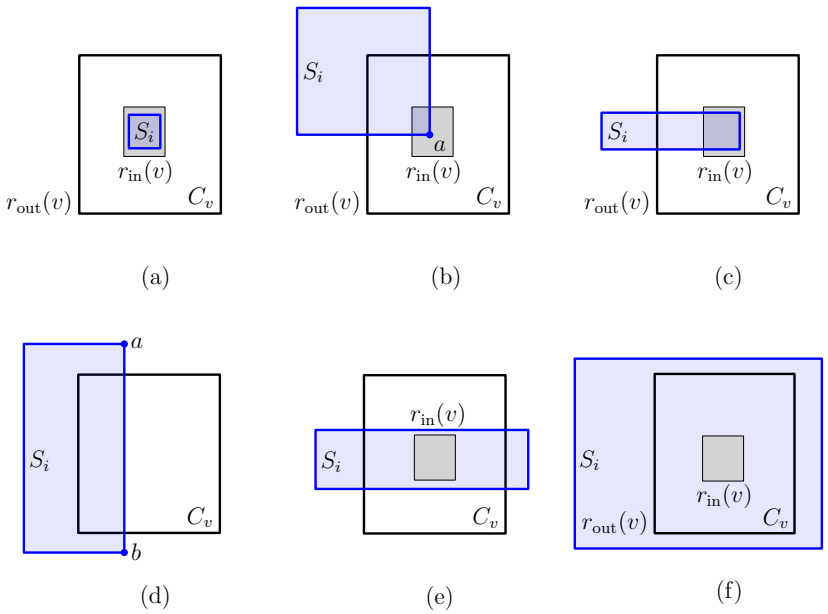

Figure 7 Illustration of (a) Case 1; (b) Case 2a; (c) Case 2b; (d) Case 3a; (e) Case 3b; and (f) Case 4.

Case 1: $S_i \subset r_{\text{out}}(v)$; see Figure 7(a). In this case, all four corners of S_i are in $r_{\text{out}}(v)$. If all four corners of S_i are in $r_{\text{in}}(v)$, then $p \in S_i \subset r_{\text{in}}(v)$, which contradicts the assumption that $p \in C_v$. Therefore, a corner of S_i lies in $r_{\text{out}}(v) \setminus r_{\text{in}}(v) = C_v$. For this corner of S_i, the highest inactive node is u or w. Consequently, the algorithm activates both siblings u and w. In particular, u is activated.

Case 2: $S_i \not\subset r_{\text{out}}(v)$ but $r_{\text{out}}(v)$ contains some corner of S_i; see Figure 7(b-c). Since C_v does not contain any corners of S_i, then all corners of S_i in $r_{\text{out}}(v)$ are in $r_{\text{in}}(v)$. Since $S_i \not\subset r_{\text{out}}(v)$, then $r_{\text{in}}(v)$ contains either one or two corners of S_i. We examine each case separately:

Case 2a: $r_{\text{in}}(v)$ contains precisely one corner of S_i. Assume w.l.o.g. that $r_{\text{in}}(v)$ contains the lower-right corner of S_i (as in Figure 7(b)). Denote by a the lower-right corner of S_i. Then S_i also contains precisely one corner of $r_{\text{in}}(v)$, namely its upper-left corner. Since the remaining three corners of S_i are outside of both $r_{\text{in}}(v)$ and C_v, they are outside of $r_{\text{out}}(v)$. Consequently, S_i also contains precisely one corner of $r_{\text{out}}(v)$, namely the upper-left corner of $r_{\text{out}}(v)$. Since $p \in S_i \cap C_v$, then Lemma 7 yields $\text{Ext}_v \cap S_i \neq \emptyset$. However, C_v was activated in a previous step. This implies that $\text{Ext}_v \subset H_{i-1}$, hence $S_i \cap H_{i-1} \neq \emptyset$: a contradiction.

Case 2b: $r_{\text{in}}(v)$ contains exactly two corners of S_i. Recall that an axis-aligned rectangle S_i crosses a cell C_v, for $v \in V(T)$ if $C_v \cap S_i \neq \emptyset$ but C_v does not contain any vertex of S_i and S_i does not contain any vertex of C_v. Consequently, S_i crosses C_v: a contradiction.

Case 3: $r_{\text{out}}(v)$ does not contain any corner of S_i, but it intersects some edges of S_i; see Figure 7(d-e). Let ab be an edge of S_i that intersects C_v, where a and b are corners of S_i. Since $r_{\text{out}}(v)$ contains neither a nor b, then both a and b are outside of $r_{\text{out}}(v)$, and so the two edges of S_i orthogonal to ab are also outside of $r_{\text{out}}(v)$. Consequently, $r_{\text{out}}(v)$ intersects one edge of S_i or two parallel edges of S_i.

Case 3a: $r_{\text{out}}(v)$ intersects precisely one edge of S_i. Assume w.l.o.g. that $r_{\text{out}}(v)$ intersects the right edge ab of S_i; see Figure 7(d). Let L be the vertical line spanned by ab, and let L^- be the left half-plane determined by L. Note that $p \in S_i \cap C_v$ implies $p \in L^-$, so $P \cap C_v \cap L^- \neq \emptyset$. Lemma 5 yields $\text{Ext}_v \cap S_i \neq \emptyset$. However, C_v was activated in a previous step. This implies that $\text{Ext}_v \subset H_{i-1}$, hence $S_i \cap H_{i-1} \neq \emptyset$: a contradiction.

Case 3b: $r_{\text{out}}(v)$ intersects two parallel edges of S_i. Assume w.l.o.g. that $r_{\text{out}}(v)$ intersects the left and right edges of S_i. If S_i does not contain any corners of $r_{\text{in}}(v)$, then S_i does not contain any vertex of C_v, and so S_i crosses C_v: a contradiction.

So we may assume that S_i contains some corners of $r_{\text{in}}(v)$. Lemma 6 yields $\text{Ext}_v \cap S_i \neq \emptyset$. However, C_v was activated in a previous step. This implies that $\text{Ext}_v \subset H_{i-1}$, hence $S_i \cap H_{i-1} \neq \emptyset$: a contradiction.

Case 4: $r_{\text{out}}(v)$ lies in the interior of S_i; see Figure 7(f). Since $p \in P \cap C_v$, then $P \cap C_v \neq \emptyset$, and so $\text{Ext}_v \neq \emptyset$. Since C_v is active, then $\text{Ext}_v \subset H_{i-1}$, and $C_v \subset r_{\text{out}}(v) \subset S_i$ implies that $\text{Ext}_v \subset S_i$. Consequently, $H_{i-1} \cap S_i \neq \emptyset$: a contradiction. ◀

▶ **Corollary 10.** *For every $p \in OPT$, the algorithm adds new points to the hitting set in $O(\log n)$ steps in which rectangles in $\mathcal{C}_p$ arrive.*

Proof. Let $\widehat{\mathcal{C}}_p \subseteq \mathcal{C}_p$ be the set of rectangles $S_i \in \mathcal{C}_p$ for which the algorithm adds new hitting points to H_i (that is, any rectangle $S_i \in \mathcal{C}_p \setminus \widehat{\mathcal{C}}_p$ contains p, but does not require new hitting points as $S_i \cap H_{i-1} \neq \emptyset$). By Lemma 9, the algorithm activates a cell containing p to hit each rectangle in $\widehat{\mathcal{C}}_p$. In the BBD tree, the cells containing p correspond to a descending path from root to leaf. Since the depth of the BBD tree for n points is $O(\log n)$, there are $O(\log n)$ such cells, which implies that $|\widehat{\mathcal{C}}_p| \leq O(\log n)$, as required. ◀

This completes the proof of Theorem 8. Now we have all the tools to prove Theorem 1.

Proof of Theorem 1. Theorem 8 implies the claim if $1 \leq \varrho \leq 2$. Assume that $\varrho > 2$. Partition the incoming rectangles into $O(1 + \log \varrho)$ classes: let $\mathcal{R}_0$ be the set of rectangles of aspect ratio in $[1, 2]$, and for $j = 1, \ldots, \lceil \log_2 \varrho \rceil$, let $\mathcal{R}_j$ be the set of rectangles of aspect ratio in $(2^j, 2^{j+1}]$. Within each class, we further separate rectangles whose width is at least their height from those whose height exceeds their width, yielding $2(1 + \lceil \log_2 \varrho \rceil)$ classes in total. Applying the linear transformation $(x, y) \mapsto (x/2^j, y)$ (respectively, $(x, y) \mapsto (x, y/2^j)$) to both P and the rectangles in the j-th class maps each rectangle to one with aspect ratio in $[1, 2]$. Since the scaling is applied to both P and the rectangles, any hitting set for the original instance corresponds to a hitting set of the same size in the scaled instance. Furthermore, the size of an offline optimum for each class is at most $|\mathrm{OPT}|$, the size of the offline hitting set of all rectangles. Running a separate instance of our online algorithm with $\varrho = 2$ on each of the $O(\log \varrho)$ classes yields a hitting set of size $O(|\mathrm{OPT}| \cdot \log n)$ per class. Taking the union of the resulting hitting sets over all classes is a hitting set of size $O(|\mathrm{OPT}| \cdot (1 + \log \varrho) \log n)$. ◀

4.3 Runtime Analysis

It is not difficult to implement our online algorithm (Section 4.1) for n points in the plane with $O(n \log^2 n)$ preprocessing time and $O(\log n)$ update time. It is known that the BBD tree can be computed in $O(n \log n)$ time [6]. Recall that the BBD tree has depth $O(\log n)$, and the cells on each level are interior-disjoint. Consequently, the extremal points on each level can be computed using four sweep-line algorithms (one for each axis-aligned direction) in $O(n \log n)$ time; and the extremal points of all cells can be computed in $O(n \log^2 n)$ time.

We need several data structures to support our online algorithm. First, we need an orthogonal range searching data structure for P that reports a point in $P \cap S_i$ for a query rectangle S_i. The classical data structure by Chazelle [15] (see also [7, Chap. 5]) already achieves $O(n \log n)$ preprocessing and $O(\log n)$ query time. Second, we need a semi-dynamic orthogonal range emptiness data structure for H_i that supports insertions and emptiness queries $H_i \cap S_i = \emptyset$. The semi-dynamic data structure by Mehlhorn and Näher [27, Theorem 8] supports $O(\log n)$ insertion and query time. (We note that $o(\log n)$ amortized update and query time is possible in the word RAM model [12, 13, 28], i.e., in the rank space of the point set P, which distorts the aspect ratios of orthogonal ranges). Finally, we also augment the BBD tree with binary variables to indicate which cells are active.

When a rectangle S_i arrives, we can test whether $H_i \cap S_i = \emptyset$ in $O(\log n)$ time using the semi-dynamic range emptiness data structure for H_i. If S_i requires new hitting points, we use the BBD tree to find the highest inactive cells containing the four corners of S_i, as well as the highest inactive cells that cross S_i, in $O(\log n)$ time. We can then activate these cells and their siblings, and add $O(1)$ extremal points to the hitting set in $O(1)$ time. If S_i does not contain any of the new extremal points added H_i, we can find a suitable hitting point in $P \cap S_i$ in $O(\log n)$ time using the orthogonal range reporting data structure for P. Finally, updating the semi-dynamic range emptiness data structure for H_i takes $O(\log n)$ time.

In the proof of Theorem 1, we run $O(1 + \log \varrho)$ parallel instances of this online algorithm, each acting on distinct classes of rectangles. So the preprocessing time increases to $O(n \log^2 n(1 + \log \varrho))$ but the update time remains $O(\log n)$.

5 Generalizations to Positive Homothets of Polygons

In Section 4, we presented an $O(\log n)$-competitive algorithm for the online hitting set problem with n points in the plane and axis-aligned squares (of aspect ratio $\varrho = 1$). Axis-aligned squares are homothets of a unit square. Since a suitable linear transformation maps any parallelogram to a unit square, our result extends to positive homothets of any parallelogram.

▶ **Corollary 11.** *For every parallelogram M in the plane, there is an $O(\log n)$-competitive algorithm for the online hitting set problem for any set of n points in the plane and a sequence of positive homothets of M in the plane.*

We further generalize Theorem 1 to positive homothets of polygons with k vertices.

▶ **Lemma 12.** *Every polygon with $k \geq 3$ vertices is the union of at most $5k-12$ parallelograms.*

Proof. Every simple polygon M with $k \geq 3$ vertices admits a triangulation with $k-2$ triangles. In general, a polygon M with $k \geq 3$ vertices and $h \geq 0$ holes admits a triangulation with $k+2h-2$ triangles [29, Lemma 5.2]. We have $h \leq k/3-1$, as every hole as well as the outer boundary has at least 3 vertices. Therefore, M is the union of at most $5k/3-4$ triangles. Every triangle T is decomposed by its three medians into four congruent subtriangles: one containing the center of T, and three incident to each of the corners of T. The union of the central subtriangle and a corner subtriangle is a parallelogram; see Figure 8. Thus, T is the union of three parallelograms, and consequently M is the union of at most $3(5k/3-4) = 5k-12$ parallelograms. ◀

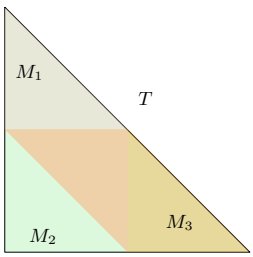

Figure 8 A triangle is the union of three parallelograms.

▶ **Lemma 13.** *Let T be a polygon that can be written as a union of k parallelograms. Then there is an $O(k^2 \log n)$-competitive deterministic algorithm for the online hitting set problem for any set P of n points and a sequence of positive homothets of T in the plane.*

Proof. Assume that T is the union of k parallelograms, i.e., $T = \bigcup_{j=1}^{k} M_j$, where each M_j is a parallelogram. By Corollary 11, there is an $O(\log n)$-competitive deterministic algorithm ALG_j for the online hitting set problem for the same point set P and a sequence of positive homothets of the parallelogram M_j, for every $j \in \{1, \ldots, k\}$.

Online algorithm. We now describe a deterministic online algorithm for the point set P and a sequence $T_1, T_2, \ldots$ of positive homothets of T. For every $i \in \mathbb{N}$, we have $T_i = a_i T + b_i$ for some scaling factor $a_i > 0$ and translation vector $b_i \in \mathbb{R}^2$. Since $T = \bigcup_{j=1}^{k} M_j$, then $T_i = \bigcup_{j=1}^{k} M_{i,j}$, where $M_{i,j} = a_i M_j + b_i$ is a positive homothet of the parallelogram M_j.

We maintain a hitting set $H_i \subset P$ for the first i homothets $\{T_\ell : \ell \leq i\}$. We initialize the algorithm ALG_j for $j = 1, \ldots, k$, that each maintain a hitting set $H_{i,j} \subset P$ for some ***subset*** of the first i parallelograms $\{M_{\ell,j} : \ell \leq i\}$. We also maintain the set $H_i = \bigcup_{j=1}^{k} H_{i,j}$.

When a homothet $T_i = a_i T + b_i$ arrives, we initialize $H_i := H_{i-1}$, and $H_{i,j} := H_{i-1,j}$ for all $j = 1, \ldots, k$. If $T_i \cap H_i \neq \emptyset$, then no further changes are needed. Otherwise, we compute the parallelograms $M_{i,j} = a_i M_j + b_i$ for $j = 1, \ldots, k$. For each $j = 1, \ldots, k$, if $P \cap M_{i,j} \neq \emptyset$, then we feed $M_{i,j}$ to the algorithm ALG_j, which in turn adds new points to $H_{i,j}$. Finally, we update H_i by setting $H_i := \bigcup_{j=1}^{k} H_{i,j}$. This completes the description of the algorithm.

Competitive analysis. The algorithm guarantees that H_i is a hitting set for the first i objects $\{T_1, \ldots, T_i\}$. It is also clear that for each new object T_i, we add $O(k)$ new points to H_i: since each algorithm ALG_j adds $O(1)$ points to $H_{i,j}$.

Let $\mathsf{OPT} \subset P$ be an offline optimum, i.e., a minimum hitting set for $\mathcal{C} = \{T_1, \ldots, T_m\}$. For each point $p \in \mathsf{OPT}$, let $\mathcal{C}_p = \{T_i \in \mathcal{C} : p \in T_i\}$ be the set of objects hit by p. It is sufficient to show that for every $p \in \mathsf{OPT}$, the algorithm adds $O(k^2 \log n)$ points to H_i in response to the arrival of the objects in $\mathcal{C}_p$.

Let $\mathcal{C}'_p$ be a set of objects $T_i \in \mathcal{C}_p$ such that our algorithm adds new points to the hitting set in step i. Since our algorithm adds $O(k)$ points to the hitting set in each step, it is enough to show that $|\mathcal{C}'_p| \leq O(k \log n)$. We further partition $\mathcal{C}'_p$ based on which parallelograms are hit by point p. For $j = 1, \ldots, k$, let $\mathcal{C}'_{p,j} = \{T_i \in \mathcal{C}'_p : p \in M_{i,j}$, and $p \notin M_{i,j'}$ for $j' < j\}$; and $\mathcal{M}'_{p,j} = \{M_{i,j} : T_i \in \mathcal{C}'_{p,j}\}$. By definition, $\mathcal{C}'_p = \bigcup_{j=1}^{k} \mathcal{C}'_{p,j}$. Note that $\mathcal{M}'_{p,i}$ is a set of homothets of the parallelogram M_j that contain the point $p \in P$. By assumption, algorithm ALG_j is $O(\log n)$-competitive, and therefore adds $O(\log n)$ points to the hitting set in response to parallelograms in $\mathcal{M}'_{p,j}$. Recall that we feed a parallelogram $M_{i,j} \in \mathcal{M}'_{p,j}$ to ALG_j only if object T_i has not been hit by H_{i-1}, and hence ALG_j adds at least one new point to the hitting set in step i. Consequently, we have $|\mathcal{M}'_{p,j}| \leq O(\log n)$. Overall, we obtain $|\mathcal{C}'_p| = \sum_{j=1}^{k} |\mathcal{C}'_{p,j}| = \sum_{j=1}^{k} |\mathcal{M}'_{p,j}| \leq O(k \log n)$, as required. Hence, the theorem follows. ◀

The combination of Lemmas 12 and 13 together immediately implies Theorem 2.

6 Conclusions

Our main result (Theorem 1) is an $O(\log n)$-competitive algorithm for the online hitting set problem for n points in the plane, and a sequence of axis-aligned squares (or axis-aligned rectangles of bounded aspect ratio). This is the first online hitting set algorithm that is $O(\log n)$-competitive for geometric objects of arbitrary sizes in the plane. Our result further generalizes to positive homothets of any simple k-gon for $k \geq 3$, and achieves an $O(k^2 \log n)$-competitive algorithm. Even though a disk can be approximated by a regular k-gon as $k \to \infty$, our generalized result does not imply any competitive algorithm for disks. Very recently, Bhore, Gupta and Kumar [9] presented a randomized online hitting set algorithm with $O(\log n)$ expected competitive ratio for n points and a sequence of pseudo-disks in the plane, including disks of arbitrary radii, or positive homothets of a convex polygon (with arbitrarily many vertices). It remains an open problem whether deterministic $O(\log n)$-competitive algorithms are possible for these classes of geometric objects.

Does our result generalize to higher dimensions? Is there an $O(\log n)$-competitive online hitting set algorithm for axis-aligned cubes of arbitrary sizes in $\mathbb{R}^d$ for a constant dimension d? Is there one for $d = 3$? While BBD trees generalize to $\mathbb{R}^d$, for any constant dimension $d \in \mathbb{N}$, our online algorithm does not generalize to (hyper-)cubes in d-space. The reason is that the extremal points (cf. Section 3.2) do not necessarily capture the intersection $C_v \cap R$ of a cell C_v of the BBD tree and an axis-aligned box. In the plane, we have shown that C_v contains a vertex of R, or $C_v \cap R$ is a crossing intersection, or $C_v \cap R$ behaves as an axis-aligned halfplane with respect to the sub-rectangles of C_v (Section 3.2). However, in $\mathbb{R}^d$, $d \geq 3$, it is possible that R contains exactly one edge of the cell $C_v = r_{\text{out}}(v)$ and yet $C_v \cap R$ does not contain any of the extremal points of C_v. It remains open whether $O(\log n)$-competitive algorithms are possible for geometric objects of arbitrary sizes in dimensions $d \geq 3$.

References

1 Pankaj K. Agarwal, Kyle Fox, Kamesh Munagala, and Abhinandan Nath. Parallel algorithms for constructing range and nearest-neighbor searching data structures. In *Proc. 35th ACM Symposium on Principles of Database Systems, (PODS)*, pages 429–440. ACM, 2016. doi:10.1145/2902251.2902303.

2 Shanli Alefkhani, Nima Khodaveisi, and Mathieu Mari. Online hitting set of d-dimensional fat objects. In *Proc. 21st International Workshop on Approximation and Online Algorithms, (WAOA)*, volume 14297 of *LNCS*, pages 134–144. Springer, 2023. doi:10.1007/978-3-031-49815-2_10.

3 Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. *SIAM J. Comput.*, 39(2):361–370, 2009. doi:10.1137/060661946.

4 Sunil Arya, Theocharis Malamatos, and David M. Mount. Space-time tradeoffs for approximate nearest neighbor searching. *J. ACM*, 57(1):1:1–1:54, 2009. doi:10.1145/1613676.1613677.

5 Sunil Arya and David M. Mount. Approximate range searching. *Comput. Geom.*, 17(3-4):135–152, 2000. doi:10.1016/S0925-7721(00)00022-5.

6 Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, 1998. doi:10.1145/293347.293348.

7 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008. doi:10.1007/978-3-540-77974-2.

8 Mark de Berg, Haggai David, Matthew J. Katz, Mark H. Overmars, A. Frank van der Stappen, and Jules Vleugels. Guarding scenes against invasive hypercubes. *Comput. Geom.*, 26(2):99–117, 2003. doi:10.1016/S0925-7721(03)00016-6.

9 Sujoy Bhore, Anupam Gupta, and Amit Kumar. Improved Online Hitting Set Algorithms for Structured and Geometric Set Systems. In *42nd International Symposium on Computational Geometry (SoCG 2026)*, volume 367 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2026. doi:10.4230/LIPIcs.SoCG.2026.14.

10 Srecko Brlek, Hugo Tremblay, Jérôme Tremblay, and Romaine Weber. Efficient computation of the outer hull of a discrete path. In *Proc. 18th IAPR International Conference on Discrete Geometry for Computer Imagery (DGCI)*, volume 8668 of *LNCS*, pages 122–133. Springer, 2014. doi:10.1007/978-3-319-09955-2_11.

11 Timothy M. Chan. Approximate nearest neighbor queries revisited. *Discret. Comput. Geom.*, 20(3):359–373, 1998. doi:10.1007/PL00009390.

12 Timothy M. Chan, Kasper Green Larsen, and Mihai Pătraşcu. Orthogonal range searching on the ram, revisited. In *Proc. 27th Symposium on Computational Geometry*, pages 1–10. ACM Press, 2011. doi:10.1145/1998196.1998198.

13 Timothy M. Chan and Konstantinos Tsakalidis. Dynamic orthogonal range searching on the RAM, revisited. *J. Comput. Geom.*, 9(2):45–66, 2018. doi:10.20382/JOCG.V9I2A5.

14 Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. *SIAM J. Comput.*, 33(6):1417–1440, 2004. doi:10.1137/S0097539702418498.

15 Bernard Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM J. Computing*, 17(3):427–462, 1988. doi:10.1137/0217026.

16 Danny Z. Chen and Haitao Wang. Computing L_1 shortest paths among polygonal obstacles in the plane. *Algorithmica*, 81(6):2430–2483, 2019. doi:10.1007/S00453-018-00540-X.

17 Minati De, Saksham Jain, Sarat Varma Kallepalli, and Satyam Singh. Online geometric covering and piercing. *Algorithmica*, 86(9):2739–2765, 2024. doi:10.1007/S00453-024-01244-1.

18 Minati De, Ratnadip Mandal, and Satyam Singh. New lower bound and algorithms for online geometric hitting set problem. *arXiv*, 2024. doi:10.48550/arXiv.2409.11166.

19 Minati De and Satyam Singh. Online hitting of unit balls and hypercubes in $\mathbb{R}^d$ using points from $\mathbb{Z}^d$. *Theor. Comput. Sci.*, 992:114452, 2024. doi:10.1016/J.TCS.2024.114452.

20 Minati De, Satyam Singh, and Csaba D. Tóth. Online hitting sets for disks of bounded radii. In *Proc. 33rd European Symposium on Algorithms, (ESA)*, volume 351 of *LIPIcs*, pages 50:1–50:16. Schloss Dagstuhl, 2025. doi:10.4230/LIPIcs.ESA.2025.50.

21 Adrian Dumitrescu, Anirban Ghosh, and Csaba D. Tóth. Online unit covering in Euclidean space. *Theor. Comput. Sci.*, 809:218–230, 2020. doi:10.1016/J.TCS.2019.12.010.

22 Adrian Dumitrescu and Csaba D. Tóth. Online unit clustering and unit covering in higher dimensions. *Algorithmica*, 84(5):1213–1231, 2022. doi:10.1007/S00453-021-00916-6.

23 Guy Even and Shakhar Smorodinsky. Hitting sets online and unique-max coloring. *Discret. Appl. Math.*, 178:71–82, 2014. doi:10.1016/J.DAM.2014.06.019.

24 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

25 Ziyue Huang and Ke Yi. Approximate range counting under differential privacy. In *Proc. 37th International Symposium on Computational Geometry (SoCG)*, volume 189 of *LIPIcs*, pages 45:1–45:14. Schloss Dagstuhl, 2021. doi:10.4230/LIPIcs.SoCG.2021.45.

26 Arindam Khan, Aditya Lonkar, Saladi Rahul, Aditya Subramanian, and Andreas Wiese. Online and dynamic algorithms for geometric set cover and hitting set. In *Proc. 39th Symposium on Computational Geometry (SoCG)*, volume 258 of *LIPIcs*, pages 46:1–46:17. Schloss Dagstuhl, 2023. See also https://arxiv.org/abs/2303.09524. doi:10.4230/LIPIcs.SOCG.2023.46.

27 Kurt Mehlhorn and Stefan Näher. Dynamic fractional cascading. *Algorithmica*, 5(2):215–241, 1990. doi:10.1007/BF01840386.

28 Christian Worm Mortensen. Fully dynamic orthogonal range reporting on RAM. *SIAM J. Comput.*, 35(6):1494–1525, 2006. doi:10.1137/S0097539703436722.

29 Joseph O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987. URL: https://www.science.smith.edu/~jorourke/books/ArtGalleryTheorems/.

30 Haitao Wang. A new algorithm for Euclidean shortest paths in the plane. In *Proc. 53rd ACM SIGACT Symposium on Theory of Computing, (STOC)*, pages 975–988. ACM, 2021. doi:10.1145/3406325.3451037.