Bichromatic Classifications of Points Using Strips

Jaegun Lee
Department of Computer Science and Engineering, Pohang University of Science and Technology, Republic of Korea

Chaeyoon Chung
Department of Computer Science and Engineering, Pohang University of Science and Technology, Republic of Korea

Hee-Kap Ahn
Department of Computer Science and Engineering, Graduate School of Artificial Intelligence, Pohang University of Science and Technology, Republic of Korea

Abstract

Given a set of n points in the plane, each colored either blue or red, we study the problem of finding a strip that separates the blue points from the red points. Specifically, we consider the following two variants: (1) locating a strip that contains no red points while maximizing the number of blue points within the strip, and (2) locating a strip that contains all blue points while minimizing the number of red points within the strip. For variant (1), we present an $O(n^2)$-time algorithm, improving upon the previously best $O(n^2 \log n)$-time result. We also show that this running time is optimal under the standard 3SUM conjecture. We also give an output-sensitive algorithm with running time $O(k_{\mathsf{opt}} n \log n)$ that returns a strip, where k_{opt} is the number of blue points not contained within the strip in an optimal solution. We extend our results to the case of up to t parallel strips, obtaining an $O(n^2 \log n)$-time algorithm. For variant (2), an optimal $\Theta(n \log n)$-time algorithm is known for $t = 1$. We show 3SUM-hardness for $t = 2$ and give an $O(n^2)$-time algorithm. For any $t \geqslant 3$, we present an $O(n^2 \log n)$-time algorithm.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Bichromatic Classification, Separation, Strip, Duality

Digital Object Identifier 10.4230/LIPIcs.SWAT.2026.29

Funding This work was partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2023-00219980), and the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2019-II191906, Artificial Intelligence Graduate School Program (POSTECH)) and (No. 2017-0-00905, Software Star Lab (Optimal Data Structure and Algorithmic Applications in Dynamic Geometric Environment)).

1 Introduction

Given a set of points in the plane, each colored either blue or red, we aim to identify a simple geometric region that separates the two colors as well as possible. In this paper, we focus on the case where the separation region is a *strip*, defined as a closed region bounded by two parallel lines of arbitrary orientation.

This model represents a fundamental classification task in which blue points represent positive instances, while red points represent negative instances. There is a fair amount of work for achieving a *perfect* bichromatic separation, wherein the two colors are distinctly separated without any errors. This is done using simple geometric shapes such as a line (or a collection of lines), a strip, a wedge, and a circle [20, 21, 27].

In many real-world datasets, however, it is often not possible to achieve a perfect separation by a low-complexity shape due to factors such as noise, overlapping distributions, or outliers. In such cases, one typically optimizes a trade-off between two types of classification errors:

false positives, where negative instances are incorrectly classified as positive, and *false negatives*, where positive instances are incorrectly classified as negative. The problems addressed in this paper correspond to two particularly important settings along this trade-off. In applications where admitting a negative instance is unacceptable, the objective is to enforce zero false positives and maximize the number of correctly classified positive instances. Conversely, in applications where missing a positive instance is unacceptable, the objective is to enforce zero false negatives and minimize the number of admitted negative instances.

We first study the following problem, which corresponds to the zero false positives setting.

▶ **Problem 1** (MaxBlue)**.** Given a set $P = B \cup R$ of n points in the plane, find a strip that contains the maximum number of points from set B while excluding any points from set R.

The MaxBlue problem enforces a strict feasibility constraint on red points. A complementary perspective is to require the separating region contains no red point while maximizing the number of covered blue points. This addresses settings where omitting a positive instance is unacceptable, leading to complete coverage of B and minimizing red points. This results in the MinRed variant, which corresponds to the zero false negatives setting.

▶ **Problem 2** (MinRed)**.** Given a set $P = B \cup R$ of n points in the plane, find a strip that contains all points of B while minimizing the number of points of R contained in the strip.

Motivation for using strips. The strip model is a natural and powerful geometric representation of separation. Unlike a single separating line, which represents a zero-width decision boundary, a strip provides a *thickened* boundary that captures a tolerance margin around a linear separator. This model represents situations in which the target set must lie within a bounded corridor that remains free of extraneous or adversarial points. Such a representation is essential when the target data corresponds to a physical object, a high-confidence cluster, or a region that must be certified safe within a specified geometric width [19]. Optimizing the placement of such a strip allows one to quantify the structural purity of a dataset and determine how much perturbation can be tolerated before separation becomes impossible. In this sense, strip separability serves as a robust geometric analogue of margin-based classification.

Applications. This geometric formulation arises naturally in several application domains requiring high-precision spatial reasoning. In metrology and industrial design, strip containment models tolerance-zone analysis: a manufactured component must lie within an allowable corridor while avoiding forbidden regions occupied by other parts [33]. In robotics and motion planning, strip separation captures narrow-passage navigation problems, where a robot must follow a collision-free linear trajectory that accounts for its physical width while avoiding obstacles. In pattern recognition and machine learning, strip separation corresponds to margin-based classification, where maximizing the width of a separating strip improves robustness and generalization [32]. Similarly, in geographic information systems and urban planning, strips model buffer zones that must fully contain protected regions while remaining disjoint from hazardous or restricted areas. These examples illustrate that strips provide a minimal yet expressive geometric primitive for modeling safety margins, tolerances, and robustness constraints.

Generalization to Multiple Strips. Furthermore, we consider natural generalizations of MaxBlue and MinRed, in which up to t parallel strips are allowed for the separation, for a fixed integer t. Extending from a single strip to multiple parallel strips is both natural and essential for modeling structured data in which target points occur in several roughly parallel

groups separated by gaps or forbidden regions. Such patterns arise in applications like crop-row detection [22], multi-lane traffic modeling [18], and circuit layout verification [23]. A single strip cannot capture these interleaved configurations, whereas multiple strips allow the model to represent periodicity and spacing, yielding a more expressive separability that maximizes covered target points while strictly excluding forbidden ones. This formulation is a discrete analogue of maximum-margin separation for nonconvex distributions, related to classical geometric separability and covering problems [26, 31].

▶ **Problem 3** (t-MaxBlue)**.** Given a set $P = B \cup R$ of n points in the plane and an integer t, find t parallel strips whose union contains the maximum number of points from set B while excluding any points from set R.

▶ **Problem 4** (t-MinRed)**.** Given a set $P = B \cup R$ of n points in the plane and an integer t, find t parallel strips whose union contains all points of B while minimizing the number of points from set R contained in the union.

See Figure 1 for an illustration of each problem.

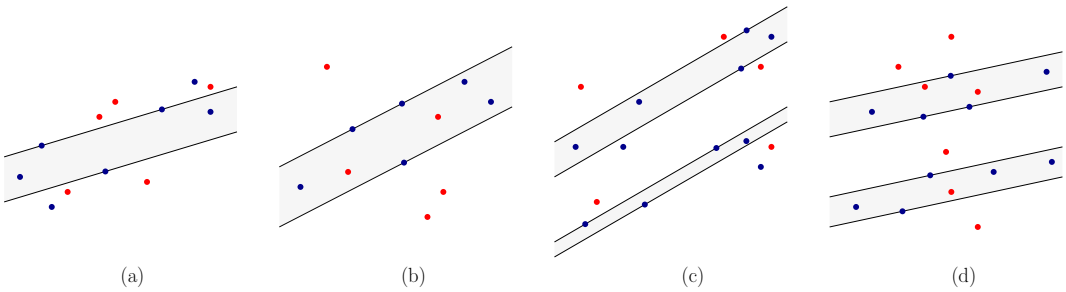

Figure 1 Optimal separation strips for (a) MaxBlue, (b) MinRed, (c) 2-MaxBlue, (d) 2-MinRed.

1.1 Related Work

In the classical approach to geometric bichromatic separation, the objective is to achieve perfect separation between the two classes. This involves finding a simple geometric shape that separates the two colors without any errors. The simplest case is perfect separation by a line, which can be decided in $O(n)$ time using linear programming [20]. Perfect separation by a circle can be decided in $O(n)$ time [27]. Hurtado et al. [21] gave $O(n \log n)$-time algorithms for perfect separation by regions bounded by two lines, including a strip, wedge, and double wedge. Moreover, several of these problems have $\Omega(n \log n)$ lower bounds [3].

The study of bichromatic separation with errors has been motivated in the context of situations where perfect separation does not exist. In the false negative setting, which minimizes the number of red points within a separation region that contains all blue points, Glazenburg et al. [17] gave $O(n \log n)$-time algorithms for finding an optimal separation by a halfplane, strip, or wedge, and an $O(n^2)$-time algorithm for finding an optimal separation by a double wedge. Bitner et al [6] studied optimal separation by a disk and gave an $O(n^2 \log n)$-time algorithm. The variant using an axis-parallel square or rectangle has also been considered [24].

Fewer results are known for the complementary false positive setting, which maximizes the number of blue points within a separation region that contains no red point. Eckstein et al. [13] showed NP-hardness for separation by a box for blue and red points in d-dimensional space when the dimension d is part of the input. For the planar case ($d = 2$), the problem

can be solved in $O(n \log^3 n)$ time [4]. Glazenburg et al. [17] gave algorithms for optimal separation of blue and red points in the plane using various geometric shapes, including halfplanes, strips, wedges, and double wedges. Their optimal strip separation algorithm, which is exactly MaxBlue, runs in $O(n^2 \log n)$ time.

Few results are known about natural generalizations concerning the union of multiple geometric objects. Seara [28] studied perfect separation using two strips, two parallel strips, and two wedges. Maji and Sadhu [24] studied MinRed using two interior-disjoint axis-parallel rectangles (or squares) and two interior-disjoint convex polygons.

Some works consider alternative objective functions for bichromatic separation. For instance, Everett et al. [15] and Glazenburg et al. [17] minimized the total number of misclassified points (uncovered blue points plus red points within the separation region). Dobkin et al. [12] maximized the discrepancy, the difference between the numbers of blue and red points within the region.

1.2 Our Results

We present exact algorithms for MaxBlue, t-MaxBlue and t-MinRed. We give an $O(n^2)$-time algorithm for MaxBlue, improving upon the previous best $O(n^2 \log n)$-time result by Glazenburg et al. [17]. We show that this running time is optimal under the standard 3SUM conjecture (Section 4).

The time complexity of our algorithm depends primarily on the total number of points n. However, a perfect separation strip can be computed in $\Theta(n \log n)$ time [28] if one exists, revealing a significant computational gap when the number k_{opt} of blue points not contained in an optimal strip for MaxBlue is small. Considering the practical motivations and applications of this problem, the excluded points typically represent outliers or noise within the dataset. In such contexts, k_{opt} is expected to be substantially smaller than n, making an output-sensitive approach highly advantageous. We present an output-sensitive algorithm for MaxBlue with running time $O(nk_{\mathsf{opt}} \log n)$. Consequently, our algorithm is expected to run in near-linear time for such practical inputs, making the output-sensitive approach particularly valuable.

To ensure robustness, we run the output-sensitive algorithm within a time budget of $O(n^2)$; if it does not terminate within this bound, we invoke the $O(n^2)$-time algorithm. This hybrid approach yields an overall complexity of $O(\min\{n^2, nk_{\mathsf{opt}} \log n\})$ for MaxBlue (Section 5), effectively bridging the gap between theoretical worst-case analysis and practical efficiency.

Finally, we consider the natural generalization for the MaxBlue problem and MinRed problem, in which up to t parallel strips are allowed for the separation. We present an $O(n^2 \log n)$-time algorithm for the t-MaxBlue problem, for any value of t (Section 6). For the t-MinRed, we present an $O(n^2 \log n)$-time algorithm for $t \geqslant 3$ (Section 7). When $t = 2$, we show that the problem is 3SUM-hard, and give an $O(n^2)$-time algorithm. A summary of our results and previous works is shown in Table 1.

Table 1 A summary of previous works and our results. † indicates that the problem is 3SUM-hard.

	MaxBlue	t-MaxBlue	t-MinRed
Previous works	$O(n^2 \log n)$ [17]	-	-
Ours	$\boldsymbol{O(n^2)}$† (Sec. 4) $\boldsymbol{O(nk_{\mathsf{opt}} \log n)}$ (Sec. 5)	$\boldsymbol{O(n^2 \log n)}$ (Sec. 6)	$\boldsymbol{O(n^2)}$† for $t = 2$ $\boldsymbol{O(n^2 \log n)}$ for $t \geqslant 3$ (Sec. 7)

2 Preliminaries

For any two points u, w in the plane, we denote the line segment connecting them by uw.

Duality

We employ the standard point-line duality transform [5]. For any point $p = (a, b) \in \mathbb{R}^2$, its dual is the line $p^* : y = ax - b$. Conversely, the dual of a nonvertical line $\ell : y = ax - b$ is the point $\ell^* = (a, b)$. This duality transform is order preserving: p lies on ℓ if and only if ℓ^* lies on p^*, and p lies above ℓ if and only if ℓ^* lies above p^*. For a set of points P, we define $P^* = \{p^* \mid p \in P\}$. A strip bounded by two parallel lines in the primal plane corresponds to the vertical line segment connecting their dual points in the dual plane (see Figure 2(a–b)).

▶ **Observation 5.** *Let σ be a strip bounded by parallel lines ℓ_1 and ℓ_2. A point $p \in \mathbb{R}^2$ lies in σ if and only if its dual line p^* intersects the vertical line segment $\ell_1^*\ell_2^*$ connecting ℓ_1^* and ℓ_2^*.*

Proof. Without loss of generality, assume ℓ_1 lies above ℓ_2. A point p lies in σ if and only if it is on or below ℓ_1 and on or above ℓ_2. Since ℓ_1 and ℓ_2 are parallel, they share the same slope, which implies that their dual points ℓ_1^* and ℓ_2^* have the same x-coordinate; thus, the segment $\ell_1^*\ell_2^*$ is vertical. By the order-preserving property, $p \in \sigma$ is equivalent to ℓ_1^* lying on or below p^* and ℓ_2^* lying on or above p^*, which means p^* intersects the segment $\ell_1^*\ell_2^*$. ◀

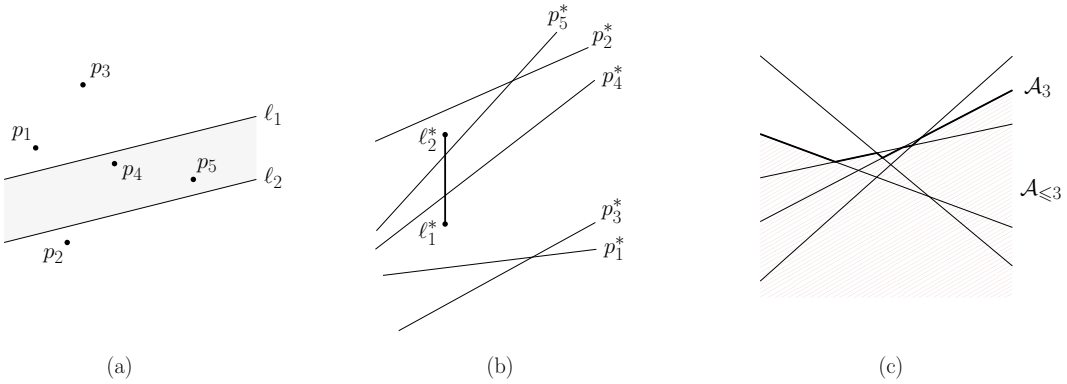

Figure 2 (a) Points $p_1, \ldots, p_5$ and a strip σ bounded by lines ℓ_1 and ℓ_2. (b) For $i = 1, \ldots, 5$, $p_i \in \sigma$ if and only if p_i^* intersects the vertical segment $\ell_1^*\ell_2^*$. (c) For an arrangement of a set L of lines, $\mathcal{A}_3(L)$ is shown in bold. The set of edges contained in the shaded region forms $\mathcal{A}_{\leqslant 3}(L)$.

Arrangement

Let L be a set of n lines in the plane. The arrangement $\mathcal{A}(L)$ is the subdivision of the plane induced by L. The *level* of a point $p \in \mathbb{R}^2$ is the number of lines in L strictly below p. Since the level is constant along any edge of $\mathcal{A}(L)$, the level of an edge is well-defined. The *k-level* of $\mathcal{A}(L)$, denoted by $\mathcal{A}_k(L)$, consists of the edges with level k; this set forms an x-monotone chain [25]. We denote the union of the i-levels for all $0 \leqslant i \leqslant k$ by $\mathcal{A}_{\leqslant k}(L)$ (see Figure 2(c) and [8, 29]).

Let $f_k(L)$ denote the complexity of $\mathcal{A}_k(L)$. It is known that $f_k(L) = O(nk^{1/3})$ [11] and $f_k(L) = n2^{\Omega(\sqrt{\log k})}$ [30]. $\mathcal{A}_k(L)$ can be computed in $O((n + f_k(L)) \log n)$ time [7] or in expected $O(n \log n + nk^{1/3})$ time [8]. In contrast, the complexity of $\mathcal{A}_{\leqslant k}(L)$ is $\Theta(nk)$ [2, 10, 14]. Everett et al. [15] computed $\mathcal{A}_{\leqslant k}(L)$ in $O(n \log n + kn)$ time. Furthermore, Chan [9] proposed an expected $O(n \log n + k^2 \log n)$-time algorithm to minimize a linear objective function over $\mathcal{A}_{\leqslant k}(L)$.

3 Problem Formulation in the Dual Plane

We reformulate the problems with the dual plane. For an input set $P = B \cup R$ of points in the primal plane, we consider its corresponding dual set $P^* = B^* \cup R^*$. As established in Observation 5, a strip in the primal plane maps to a vertical segment in the dual plane. By extension, a set of t parallel strips in the primal plane corresponds to t vertical segments sharing a common x-coordinate in the dual plane. Thus, each problem can be expressed in terms of dual-plane geometry as follows.

▶ **Problem 6** (MaxBlue*). Given a set $P^* = B^* \cup R^*$ of n lines in the plane, find a vertical segment that intersects the maximum number of lines in B^* while intersecting no lines in R^*.

▶ **Problem 7** (MinRed*). Given a set $P^* = B^* \cup R^*$ of n lines in the plane, find a vertical segment that intersects all lines in B^* while minimizing the number of lines in R^* that intersect it.

▶ **Problem 8** (t-MaxBlue*). Given a set $P^* = B^* \cup R^*$ of n lines in the plane and an integer t, find t vertical segments with a common x-coordinate whose union intersects the maximum number of lines in B^* while intersecting no lines in R^*.

▶ **Problem 9** (t-MinRed*). Given a set $P^* = B^* \cup R^*$ of n lines in the plane and an integer t, find t vertical segments with a common x-coordinate whose union intersects all lines in B^* while minimizing the number of lines in R^* intersected by the union.

4 $O(n^2)$-time algorithm for MaxBlue and 3SUM-hardness

We present an $O(n^2)$-time algorithm for MaxBlue, improving upon the previous best $O(n^2 \log n)$-time algorithm [17]. We also show that this running time is optimal under the standard 3SUM conjecture.

▶ **Theorem 10.** *We can solve MaxBlue and MaxBlue* in $O(n^2)$ time.*

Proof. We can solve MaxBlue by solving MaxBlue*. Let $P^* = B^* \cup R^*$ be a set of n lines in the plane. Our algorithm is rather simple. It computes a vertical segment in each cell C of $\mathcal{A}(R^*)$ that intersects the most lines in B^*. Among all such segments, it reports the one that intersects the most. An optimal vertical segment intersects no lines in R^*, which ensures the algorithm's correctness. However, achieving the optimal running time is challenging.

We describe the algorithm in detail and analyze its time complexity. Consider the arrangements $\mathcal{A}(R^*)$ and $\mathcal{A}(P^*)$. Note that once $\mathcal{A}(P^*)$ is computed, the arrangement $\mathcal{A}(R^*)$ is implicitly available. Let C be a cell of $\mathcal{A}(R^*)$. Imagine we sweep C with a vertical line ℓ from left to right, starting from the leftmost vertex of C. Let c_B be the number of lines in B^* whose intersection with ℓ lies in the interior of C. Observe that $c_B = 0$ at the leftmost vertex of C, and c_B changes only when ℓ hits an intersection point x between a line h in B^* and the boundary of C, excluding the leftmost and rightmost vertices: c_B increases by one if h enters C at x, and c_B decreases by one if h leaves C at x. Thus, we can maintain c_B by traversing the upper and lower boundary chains of C from their leftmost vertex within the arrangement $\mathcal{A}(P^*)$, and determine the position of ℓ at which c_B is maximized. We repeat this procedure for every cell in $\mathcal{A}(R^*)$.

The arrangement $\mathcal{A}(P^*)$ can be computed in $O(n^2)$ time. For each cell C in $\mathcal{A}(R^*)$, the procedure above takes $O(n_C)$ time, where n_C denotes the number of vertices of $\mathcal{A}(P^*)$ lying on the boundary of C. Let $d(v)$ be the degree of a vertex v in $\mathcal{A}(P^*)$, that is, number

of edges incident to v in $\mathcal{A}(P^*)$. Observe that each vertex v of $\mathcal{A}(P^*)$ appears on the boundaries of at most $d(v)$ cells of $\mathcal{A}(R^*)$. Thus, the total time complexity is bounded by $\sum_{C\in\mathcal{A}(R^*)} O(n_C) = \sum_{v\in\mathcal{A}(P^*)} O(d(v)) = O(n^2)$. ◀

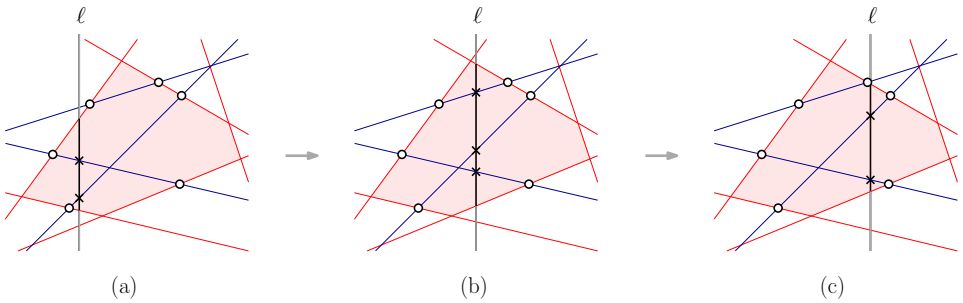

Figure 3 Sweeping a cell in $\mathcal{A}(R^*)$ with a vertical line ℓ. (a) $c_B = 2$, (b) $c_B = 3$, and (c) $c_B = 2$.

We show that MaxBlue is 3SUM-hard by a reduction from the GEOMBASE problem. Thus, there is no strongly sub-quadratic algorithm for MaxBlue unless the 3SUM problem can be solved in $O(n^{2-\varepsilon})$ time for any $\varepsilon > 0$. The GEOMBASE problem is defined as follows: Given n points in the plane with integer coordinates on three horizontal lines $y = 0$, $y = 1$ and $y = 2$, determine whether there exists a non-horizontal line containing three of the points. It is known that the GEOMBASE problem is 3SUM-hard [16].

▶ **Theorem 11.** *MaxBlue and MaxBlue* are 3SUM-hard.*

Proof. We use a reduction from the GEOMBASE problem. Consider n blue points in the plane with integer x-coordinates larger than 0 and at most M on three horizontal lines $y = 0$, $y = 1$ and $y = 2$, for some integer M. For each blue point $p = (\alpha, 1)$, we place two red points $q = (\alpha + 1/(2\sqrt{4+M^2}), 1)$ and $q' = (\alpha - 1/(2\sqrt{4+M^2}), 1)$. We show that no line $\ell : (b-a)y - 2x + 2a = 0$ passing through any two blue points $(a, 0), (b, 2)$ intersects qq' unless it passes through p. If ℓ does not pass through p, $|2\alpha - a - b| \geqslant 1$ since α, a, and b are all integers. Then the distance between p and ℓ is $|2\alpha - a - b|/\sqrt{4+(a-b)^2} \geqslant |2\alpha - a - b|/\sqrt{4+M^2} \geqslant 1/\sqrt{4+M^2} > 1/(2\sqrt{4+M^2})$. See Figure 4(a).

Similarly, for each $i = 0, 2$, we place two red points q, q' on $y = i$ for each blue point (β, i) such that no line passing through any two blue points $(a, j), (b, k)$ for $\{j, k\} = \{0, 1, 2\} \setminus \{i\}$ intersects qq' unless it passes through (β, i). Thus, we have n blue points and $2n$ red points in total. The reduction takes $O(n)$ time.

If a non-horizontal line contains three blue points, there is a strip containing three blue points while excluding any red points. If no such line exists, no strip contains three blue points while excluding any red points. See Figure 4(b). ◀

5 $O(nk_{\text{opt}} \log n)$-Time Algorithm for MaxBlue

We present an $O(nk_{\text{opt}} \log n)$-time algorithm for MaxBlue, where k_{opt} denotes the number of blue points not covered by an optimal strip. In the dual plane, we are given a set $P^* = B^* \cup R^*$ of n lines in the plane, and our goal is to find a vertical segment that intersects the maximum number of lines in B^* while intersecting no lines in R^*. For convenience, we will refer to the lines in B^* as blue lines and the lines in R^* as red lines.

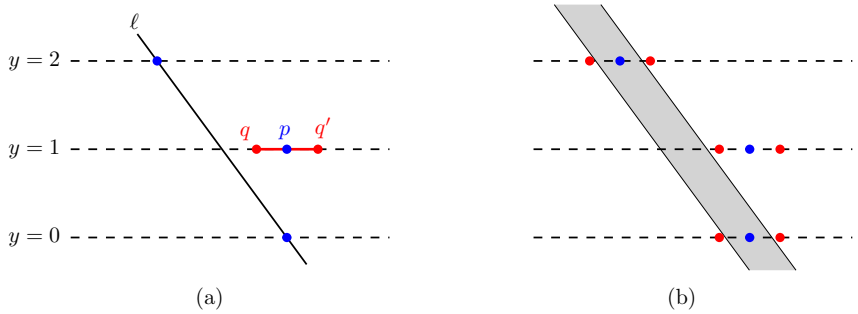

Figure 4 (a) We place two red points q, q' on $y = 1$ for each blue point p on $y = 1$ such that no line passing through any two blue points $(a, 0), (b, 2)$ intersects qq' unless it passes through p. (b) A non-horizontal line contains three blue points if and only if there is a strip containing three blue points while excluding any red points.

We focus on vertical segments whose endpoints lie on blue lines. If an endpoint does not lie on a blue line, we can shorten the segment until each endpoint does, without changing the set of blue lines it intersects.

In Section 5.1, we compute a value M such that $k_{\mathsf{opt}}/2 \leqslant M < 2k_{\mathsf{opt}}$. Then, in Section 5.2, we determine k_{opt} and compute an optimal vertical segment using binary search. We assume $k_{\mathsf{opt}} \neq 0$, as the case $k_{\mathsf{opt}} = 0$ can be detected in $O(n \log n)$ time [21].

5.1 Reducing a Search Space

We find a value M such that $k_{\mathsf{opt}}/2 \leqslant M < 2k_{\mathsf{opt}}$ using exponential search. For each $k = 2^0, 2^1, \ldots$, we check if there exists a vertical segment with exactly k blue lines above it and exactly k blue lines below it, and no red lines intersecting it. Let k' be the smallest value of k for which the exponential search succeeds. We have $k_{\mathsf{opt}} \leqslant 2k'$ because there exists a vertical segment that intersects no red lines and misses exactly $2k'$ blue lines. The search fails for $k'/2$. We show $k_{\mathsf{opt}} > k'/2$. Suppose $k_{\mathsf{opt}} \leqslant k'/2$. Then there exists a vertical segment with k_1 blue lines above and k_2 blue lines below, intersecting no red line, such that $\max\{k_1, k_2\} \leqslant k_{\mathsf{opt}} \leqslant k'/2$. Shortening it gives a vertical segment with $k'/2$ blue lines above it and $k'/2$ blue lines below it, intersecting no red lines. This contradicts that the search fails for $k'/2$. Thus, $k_{\mathsf{opt}} > k'/2$.

To perform this exponential search, we observe that a vertical segment with exactly k blue lines above it and exactly k blue lines below it has one endpoint on $\mathcal{A}_k(B^*)$ and the other on $\mathcal{A}_{|B|-k-1}(B^*)$. We denote such a segment by $\Upsilon_k(x)$, where x is its x-coordinate. Imagine moving $\Upsilon_k(x)$ from left to right while keeping its endpoints on these two x-monotone chains. As we increase x, we maintain the number of red lines intersected by $\Upsilon_k(x)$ and check if there exists a position x' such that $\Upsilon_k(x')$ intersects no red lines.

▶ **Lemma 12.** *For $0 \leqslant i < j < |B|$, suppose that the points in $\mathcal{A}_i(B^*) \cap R^*$ and $\mathcal{A}_j(B^*) \cap R^*$ are given in sorted order of their x-coordinates. We can decide whether there exists a vertical segment intersecting no red lines, with one endpoint on $\mathcal{A}_i(B^*)$ and the other on $\mathcal{A}_j(B^*)$ in $O(|R^*| + |\mathcal{A}_i(B^*) \cap R^*| + |\mathcal{A}_j(B^*) \cap R^*|)$ time.*

Proof. Let L be a value smaller than the smallest x-coordinate among the points in $(\mathcal{A}_i(B^*) \cap R^*) \cup (\mathcal{A}_j(B^*) \cap R^*)$. Any vertical segment lying on or to the left of $x = L$ with one endpoint on $\mathcal{A}_i(B^*)$ and the other on $\mathcal{A}_j(B^*)$ intersects the same set of red lines. See Figure 5.

We sweep the plane with a vertical line ℓ, starting at $x = L$ and moving to the right, and count the number N_r of red lines intersected by the vertical segment whose endpoints are $\mathcal{A}_i(B^*) \cap \ell$ and $\mathcal{A}_j(B^*) \cap \ell$. At $x = L$, we compute N_r in $O(|R^*|)$ time.

As ℓ moves to the right, N_r changes only when ℓ passes through a point in $(\mathcal{A}_i(B^*) \cap R^*) \cup (\mathcal{A}_j(B^*) \cap R^*)$. Therefore, given these intersection points in increasing order of their x-coordinates, we can update N_r while sweeping and determine whether there exists a position at which the segment intersects no red lines in $O(|\mathcal{A}_i(B^*) \cap R^*| + |\mathcal{A}_j(B^*) \cap R^*|)$ time. ◀

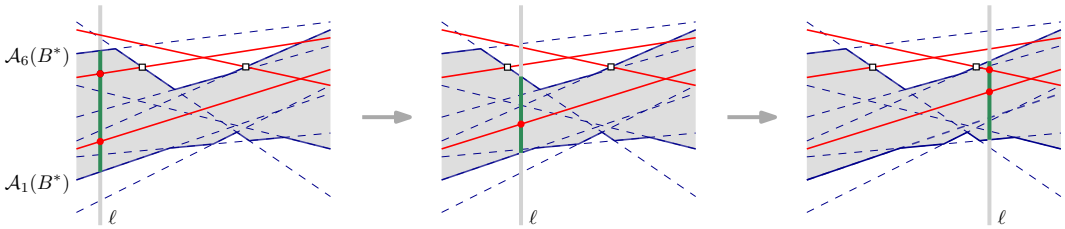

Figure 5 Three snapshots of the algorithm in Lemma 12 for $i = 1$ and $j = 6$.

Let k be an integer with $0 \leqslant k < |B|$. We show that $\mathcal{A}_k(B^*) \cap R^*$ and $\mathcal{A}_{|B|-k-1}(B^*) \cap R^*$ can be computed and sorted in $O(nk \log n)$ time.

▶ **Lemma 13.** *We can compute a collection of $k + 1$ edge-disjoint concave chains whose union is $\mathcal{A}_{\leqslant k}(B^*)$ in $O(nk^{1/3} \log n)$ time.*

Proof. We use a standard algorithm to decompose $\mathcal{A}_{\leqslant k}(B^*)$ into $O(k)$ concave chains as in [1, 9]. Chan [9] showed that this can be done in the same time we construct $\mathcal{A}_k(B^*)$. Thus, the chains can be constructed in $O(nk^{1/3} \log n)$ time [7, 11]. ◀

▶ **Lemma 14.** *$\mathcal{A}_{\leqslant k}(B^*) \cap R^*$ has size $O(nk)$ and can be computed in $O(nk \log n)$ time.*

Proof. We first decompose $\mathcal{A}_{\leqslant k}(B^*)$ into $k + 1$ concave chains in $O(nk^{1/3} \log n)$ time using Lemma 13. By construction, every vertex of each chain lies on $\mathcal{A}_k(B^*)$. Therefore, each chain has complexity $O(nk^{1/3})$. Then, for every pair consisting of a chain and a line in R^*, we compute all intersection points between them. Since each chain is concave, a line intersects it at most twice, and these intersections can be found in $O(\log n)$ time. As there are $O(k)$ chains and $O(n)$ lines, the total number of intersections is $O(nk)$, and all intersections can be computed in $O(nk \log n)$ time. See Figure 6. ◀

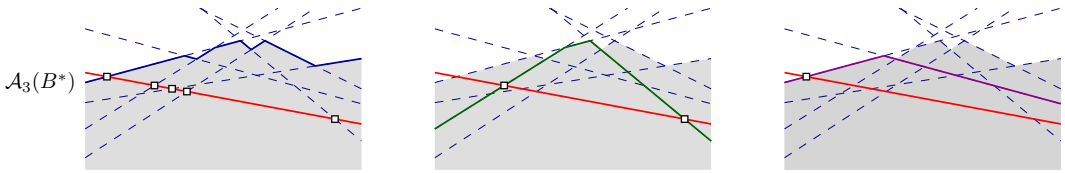

Figure 6 (a) For a set B^* of 7 blue lines, a red line intersects $\mathcal{A}_{\leqslant 3}(B^*)$ in five points. (b-c) After decomposing $\mathcal{A}_{\leqslant 3}(B^*)$ into concave chains, a red line intersects each concave chain in at most two points.

Let $Q = \mathcal{A}_{\leqslant k}(B^*) \cap R^*$, which can be computed in $O(nk \log n)$ time by Lemma 14. Observe that $\mathcal{A}_k(B^*) \cap R^* = \mathcal{A}_k(B^*) \cap Q$. We can compute $\mathcal{A}_k(B^*)$ in $O(nk^{1/3} \log n)$ time [7, 11]. After sorting the points of Q in increasing order of their x-coordinates, we traverse the chain $\mathcal{A}_k(B^*)$ from left to right and, for each point of Q, check whether it lies on $\mathcal{A}_k(B^*)$. Since both sequences are processed in increasing order of their x-coordinates, this can be done in linear time in the size of Q and $\mathcal{A}_k(B^*)$. Consequently, we can identify all points in $\mathcal{A}_k(B^*) \cap R^*$ within the same time bound.

▶ **Corollary 15.** *$\mathcal{A}_k(B^*) \cap R^*$ has size $O(nk)$ and can be computed in $O(nk \log n)$ time.*

Observe that $\mathcal{A}_{|B|-k-1}(B^*) \cap R^*$ has size $O(nk)$ and can be computed in $O(nk \log n)$ time by Corollary 15 and symmetry. Thus, by Lemma 12 and Corollary 15, we can decide whether there exists $\Upsilon_k(x)$ for some x intersecting no red lines in $O(nk \log n)$ time.

Recall that $k_{\mathsf{opt}}/2 \leqslant k' < 2k_{\mathsf{opt}}$, where k' is the smallest value of $k = 2^0, 2^1, \ldots$ for which the exponential search succeeds. Once the search succeeds, we terminate it and set $M = k'$. The total running time is $\sum_{k=2^0,2^1,\ldots,k'} O(nk \log n) = O(nk' \log n)$, which is $O(nk_{\mathsf{opt}} \log n)$.

▶ **Lemma 16.** *We can compute M such that $k_{opt}/2 \leqslant M < 2k_{opt}$ in $O(nk_{opt} \log n)$ time.*

5.2 Finding an Optimal Strip

By Lemma 16, we can compute M such that $k_{\mathsf{opt}} \in [M/2, 2M]$ in $O(nk_{\mathsf{opt}} \log n)$ time. Observe that for an integer $i \in [M/2, 2M]$, there exists a vertical segment that intersects no red lines and misses exactly i blue lines if and only if $i \geqslant k_{\mathsf{opt}}$. Thus, we find k_{opt} using binary search within $[M/2, 2M]$ as follows. At each binary step, let $[s, e]$ be the current range and m its median. Initially, $[s, e] = [M/2, 2M]$. We test if there exists a vertical segment intersecting no red lines and missing exactly m blue lines. If so, update $[s, e]$ to $[s, m]$. Otherwise, update $[s, e]$ to $[m + 1, e]$. Continue the binary search recursively.

Preprocessing

We compute, for every $i = 0, \ldots, 2M$, the sorted list of points in $\mathcal{A}_i(B^*) \cap R^*$ in increasing order of their x-coordinates. We also do this for every $i = |B| - 2M - 1, \ldots, |B| - 1$. We show that this can be done in $O(nM \log n)$ time in total.

▶ **Lemma 17.** *For every $i = 0, \ldots, 2M$, the list of points in $\mathcal{A}_i(B^*) \cap R^*$ sorted in increasing order of their x-coordinates can be computed in $O(nM \log n)$ total time.*

Proof. We first compute the arrangement $\mathcal{A}_{\leqslant 2M}(B^*)$ in $O(n \log n + nM)$ time [15]. This allows us to store each chain $\mathcal{A}_i(B^*)$ explicitly for every $i = 0, \ldots, 2M$ by tracing the arrangement. Note that the complexity of $\mathcal{A}_{\leqslant 2M}(B^*)$ is $O(nM)$.

Then we compute all intersection points $Q = \mathcal{A}_{\leqslant 2M}(B^*) \cap R^*$ and sort them in increasing order of their x-coordinates. By Lemma 14, this can be done in $O(nM \log n)$ time.

We determine which chain $\mathcal{A}_i(B^*)$ contains each point of Q by sweeping the plane with a vertical line ℓ. For each $i = 0, \ldots, 2M$, we maintain the edge of $\mathcal{A}_i(B^*)$ intersected by ℓ, sorted by increasing i. Since the points $\mathcal{A}_i(B^*) \cap \ell$ for $i = 0, \ldots, 2M$ are sorted by their y-coordinates, we can determine which chain $\mathcal{A}_i(B^*)$ contains the point of Q hit by ℓ by binary search in $O(\log M)$ time. Since $|Q| = O(nM)$, it takes $O(nM \log M)$ time in total. ◀

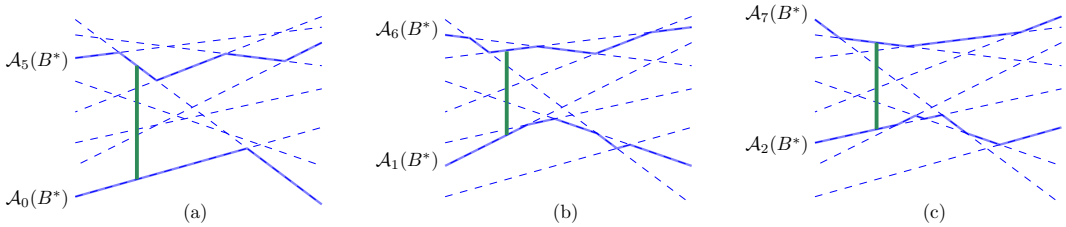

Figure 7 For 8 blue lines, a vertical segment missing two blue lines has endpoints in $\mathcal{A}_i(B^*)$ and $\mathcal{A}_j(B^*)$ for $(i,j) \in \{(0,5),(1,6),(2,7)\}$.

Binary search

At each binary step with range $[s,e]$, we test if there exists a vertical segment missing exactly $m = \lfloor (s+e)/2 \rfloor$ blue lines and intersecting no red lines. If a vertical segment with endpoints in $\mathcal{A}_i(B^*)$ and $\mathcal{A}_j(B^*)$ misses exactly m blue lines, then $0 \leqslant i \leqslant m$ and $j = |B| - m - 1 + i$. See Figure 7. Thus, for every $i = 0, \ldots, m$, we check if there exists a vertical segment with endpoints in $\mathcal{A}_i(B^*)$ and $\mathcal{A}_{|B|-m-1+i}(B^*)$ that intersects no red lines.

By preprocessing, we have the sorted list of points in $\mathcal{A}_i(B^*) \cap R^*$ and $\mathcal{A}_{|B|-m-1+i}(B^*) \cap R^*$ for every $i \in \{0,1,\ldots,m\}$. By Lemma 12, we can test in $O(|R^*| + |\mathcal{A}_i(B^*) \cap R^*| + |\mathcal{A}_{|B|-m-1+i}(B^*) \cap R^*|)$ time whether there exists a vertical segment with endpoints in $\mathcal{A}_i(B^*)$ and $\mathcal{A}_{|B|-m-1+i}(B^*)$ that intersects no red lines. Thus, the total time is

$$\sum_{i=0,\ldots,m} O(|R^*| + |\mathcal{A}_i(B^*) \cap R^*| + |\mathcal{A}_{|B|-m-1+i}(B^*) \cap R^*|),$$

which is $O(nm + |\mathcal{A}_{\leqslant m}(B^*)| + |\mathcal{A}_{\geqslant |B|-m-1}(B^*)|)$, where $\mathcal{A}_{\geqslant |B|-m-1}(B^*)$ denotes the union of i-levels of $\mathcal{A}(B^*)$ for all $i \geqslant |B| - m - 1$. By Lemma 14, $|\mathcal{A}_{\leqslant m}(B^*) \cap R^*| = O(nm)$, and $|\mathcal{A}_{\geqslant |B|-m-1}(B^*) \cap R^*| = O(nm)$ by symmetry. Thus, each binary-search step takes $O(nm) = O(nk_{\mathsf{opt}})$ time.

Since the binary search takes $O(\log M) = O(\log k_{\mathsf{opt}})$ steps, its total time is $O(nk_{\mathsf{opt}} \log k_{\mathsf{opt}})$. Thus, the overall running time is $O(nM \log n + nk_{\mathsf{opt}} \log k_{\mathsf{opt}}) = O(nk_{\mathsf{opt}} \log n)$.

▶ **Theorem 18.** *We can solve* MaxBlue *and* MaxBlue* *in* $O(nk_{opt} \log n)$ *time.*

Our output-sensitive algorithm is asymptotically faster than the $O(n^2)$-time algorithm for $k_{\mathsf{opt}} = o(n/\log n)$. We can guarantee the running time $O\left(\min\{n^2, nk_{\mathsf{opt}} \log n\}\right)$ by terminating the exponential search immediately when k exceeds $n/\log n$ and running the quadratic-time algorithm.

6 Algorithm for t-MaxBlue

We present an $O(n^2 \log n)$-time algorithm for t-MaxBlue. Note that t is part of the input, but it does not affect the asymptotic running time.

▶ **Theorem 19.** *We can solve* t*-MaxBlue and* t*-MaxBlue** *in* $O(n^2 \log n)$ *time.*

Proof. In t-MaxBlue*, we are given a set $P^* = B^* \cup R^*$ of n lines in the plane and an integer t, and find t vertical segments with a common x-coordinate whose union intersects the maximum number of lines in B^* while intersecting no lines in R^*. For convenience, we will refer to the lines in B^* as blue lines and the lines in R^* as red lines.

We first compute the arrangement $\mathcal{A}(R^* \cup B^*)$ in $O(n^2)$ time. Then, we sweep the plane with a vertical line ℓ from left to right. For any fixed position, ℓ is subdivided by $\ell \cap R^*$ into $|R| + 1$ segments (possibly open or degenerate to a point). By shrinking those segments, we consider the maximal vertical segments whose endpoints lie on $\ell \cap B^*$ and that do not intersect any lines of R^*. See Figure 8. Let $s_1, s_2, \ldots, s_{|R|+1}$ denote these segments, ordered by decreasing y-coordinate, and let n_i denote the number of blue lines intersected by s_i.

During the sweep, we maintain two lists, L_1 and L_2. List L_1 contains pairs (s_i, n_i), sorted by decreasing y-coordinate of the segments. List L_2 contains the sorted numbers n_i. Any combinatorial changes to the vertical segments in ℓ occur only when ℓ hits a vertex of $\mathcal{A}(R^* \cup B^*)$. Any changes to the numbers occur only when ℓ hits an intersection of a blue line and a red line, which is a vertex of $\mathcal{A}(R^* \cup B^*)$. At such an intersection, only two consecutive numbers n_i and n_{i+1} change, each by at most one. In Figure 8, L_1 has $(s_2, 3)$ and $(s_3, 2)$ at ℓ_1, but they change to $(s_2, 2)$ and $(s_3, 3)$ at ℓ_2 due to the intersection u. After ℓ_4, s_2 intersects no blue lines, and L_1 has $(s_2, 0)$. Let *an event* refer to a moment when ℓ hits a vertex of $\mathcal{A}(R^* \cup B^*)$.

By maintaining a balanced binary search tree that stores $R^* \cup B^*$ in sorted order of their intersections with ℓ, as in a standard plane sweep algorithm, we can compute all events sorted in their x-coordinates in $O(n^2 \log n)$ total time. At an event with a blue line and a red line defining the event, we can update both L_1 and L_2 in $O(\log n)$ time. Thus, the overall running time is $O(n^2 \log n)$.

To find an optimal solution, we maintain an additional value, the sum of the largest t values in L_2. The sum can be updated in $O(\log n)$ time per event by maintaining L_2. Let $N \subseteq \{n_1, \ldots, n_{|R|+1}\}$ be the set of the t largest values in L_2, and let $S \subseteq \{s_1, \ldots, s_{|R|+1}\}$ be the corresponding segments at a position of ℓ where the sum is maximized. The vertical segments in S intersect $\sum_{m \in N} m$ blue lines but no red lines, and S is an optimal solution to t-MaxBlue*. By the dual transform, the dual of the vertical segments in S are t parallel strips that contain $\sum_{m \in N} m$ blue points but no red points. Thus, the set of t parallel strips, dual to S, is an optimal solution to t-MaxBlue. ◀

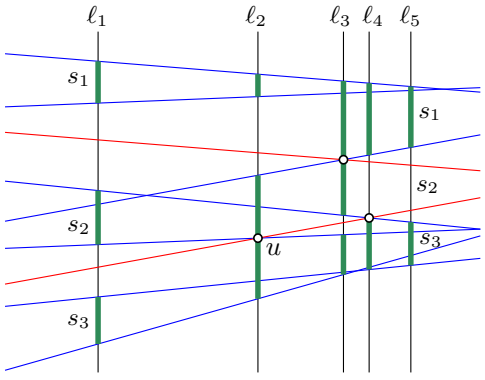

Figure 8 ℓ moves from ℓ_1 to ℓ_5. Any changes to L_2 occur only when ℓ hits an intersection of a blue line and a red line, marked by a small disk.

7 Algorithm and 3SUM-hardness for t-MinRed

We first show that t-MinRed is 3SUM-hard even when $t = 2$. Then we show that t-MinRed can be solved in $O(n^2)$ time for $t = 2$ and in $O(n^2 \log n)$ time for $t \geqslant 3$.

▶ **Lemma 20.** *2-MinRed is 3SUM-hard.*

Proof. We use a reduction from the GEOMBASE problem. Consider a set of n red points on the horizontal lines $y = 0$, $y = 1$, and $y = 2$, each with an integer x-coordinate satisfying $0 < x \leqslant M$ for some integer M. For each $i = 0, 1, 2$, and for each red point $p = (\beta, i)$, we place two blue points q, q' on $y = i$ as in Theorem 11 in total $O(n)$ time such that no line passing through any two red points (a, j), (b, k) for $\{j, k\} = \{0, 1, 2\} \setminus \{i\}$ intersects qq' unless it passes through p. We have n red points and $2n$ blue points in total. The reduction takes $O(n)$ time.

Suppose that there exists a non-horizontal line ℓ containing three red points. Then there are two strips parallel to ℓ such that their union contains all blue points while containing exactly $n - 3$ red points. If no non-horizontal line contains three red points, no two parallel strips can cover all blue points while excluding three or more red points. In particular, every pair of parallel strips whose union contains all blue points must contain at least $n - 2$ red points. ◀

In the dual plane, we are given a set $P^* = B^* \cup R^*$ of n lines in the plane and an integer t. Our goal is to find t vertical segments with a common x-coordinate whose union intersects all lines in B^* while minimizing the number of lines in R^* intersected by the union.

▶ **Theorem 21.** *We can solve 2-MinRed in $O(n^2)$ time.*

Proof. For a cell C of $\mathcal{A}(B^*)$, let $\mathsf{H_U}(C)$ and $\mathsf{H_L}(C)$ denote the upper hull and the lower hull of C, respectively. Let s_1 and s_2 be an optimal pair of vertical segments where s_1 lies above s_2. Since $s_1 \cup s_2$ intersects all lines in B^*, s_1 has its upper endpoint on $\mathcal{A}_{|B|-1}(B^*)$, s_2 has its lower endpoint on $\mathcal{A}_0(B^*)$, and both s_1 and s_2 are incident to a common cell in $\mathcal{A}(B^*)$. See Figure 9(a).

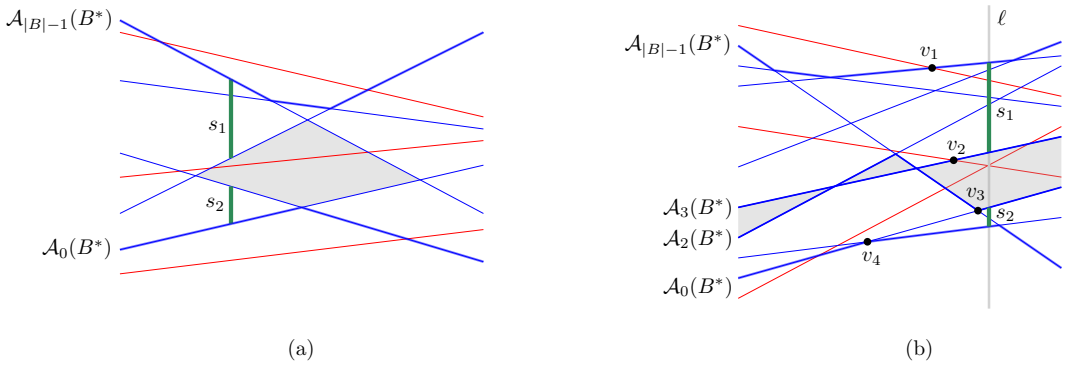

Figure 9 (a) s_1 has its upper endpoint on $\mathcal{A}_{|B|-1}(B^*)$, s_2 has its lower endpoint on $\mathcal{A}_0(B^*)$, and both s_1 and s_2 are incident to a common cell in $\mathcal{A}(B^*)$. (b) The gray region represents the cells in $\mathcal{C}_2$. The number of red lines intersected by $s_1 \cup s_2$ is $n(v_2) - n(v_1) + n(v_4) - n(v_3) = 1 - 0 + 3 - 3 = 1$.

Therefore, for each cell C in $\mathcal{A}(B^*)$, except the cell below $\mathcal{A}_0(B^*)$ and the cell above $\mathcal{A}_{|B|-1}(B^*)$, we find two vertical segments s_1 and s_2 such that their endpoints are the intersections between a vertical line and $\{\mathcal{A}_{|B|-1}(B^*), \mathsf{H_U}(C), \mathsf{H_L}(C), \mathcal{A}_0(B^*)\}$, and the number of lines in R^* intersected by $s_1 \cup s_2$ is minimized.

To do this efficiently, we first compute $\mathcal{A}(B^* \cup R^*)$ in $O(n^2)$ time, which also gives $\mathcal{A}(B^*)$. For a vertex v in $\mathcal{A}(B^* \cup R^*)$, let $n(v)$ denote the number of lines in R^* lying above v. We can compute $n(v)$ for every vertex v in $\mathcal{A}(B^* \cup R^*)$ in $O(n^2)$ time by traversing the arrangement $\mathcal{A}(B^* \cup R^*)$.

For a cell C in $\mathcal{A}(B^*)$, all points in $\mathsf{H_L}(C)$ have the same level in $\mathcal{A}(B^*)$, and all points in $\mathsf{H_U}(C)$ have the same level in $\mathcal{A}(B^*)$, if they are defined. Their levels differ by one in $\mathcal{A}(B^*)$. Let $\mathcal{C}_i$ denote the set of cells in $\mathcal{A}(B^*)$ whose lower hull lies on $\mathcal{A}_i(B^*)$ for $i = 0, \ldots, |B| - 2$. Then, $\mathcal{C}_i$ is the set of cells in $\mathcal{A}(B^*)$ whose upper hull lies on $\mathcal{A}_{i+1}(B^*)$. Note that $\bigcup_{i=0}^{|B|-2} \mathcal{C}_i$ is the set of all cells in $\mathcal{A}(B^*)$, except the cell below $\mathcal{A}_0(B^*)$ and the cell above $\mathcal{A}_{|B|-1}(B^*)$.

Consider $\mathcal{C}_i$ for a fixed $0 \leqslant i \leqslant |B| - 2$. We find two optimal vertical segments lying on a vertical line ℓ, restricted to the cells in $\mathcal{C}_i$ as follows. Imagine we sweep the plane with a vertical line ℓ from left to right. At a fixed position of ℓ, let s_1 and s_2 be the vertical segments lying on ℓ whose endpoints are the intersections of ℓ and each of the four chains in $\{\mathcal{A}_{|B|-1}(B^*), \mathcal{A}_{i+1}(B^*), \mathcal{A}_i(B^*), \mathcal{A}_0(B^*)\}$. See Figure 9(b).

Let $v_1 \in \mathcal{A}_{|B|-1}(B^*)$, $v_2 \in \mathcal{A}_{i+1}(B^*)$, $v_3 \in \mathcal{A}_i(B^*)$, and $v_4 \in \mathcal{A}_0(B^*)$ be the rightmost vertices in $\mathcal{A}(B^* \cup R^*)$ among the ones on or left to ℓ. Then, the number of lines in R^* intersected by $s_1 \cup s_2$ is $n(v_2) - n(v_1) + n(v_4) - n(v_3)$. Thus, while we slide ℓ, the number changes only when ℓ encounters a vertex of $\mathcal{A}(B^* \cup R^*)$ contained in the four chains.

Since $n(v)$ is computed for every vertex v in $\mathcal{A}(B^* \cup R^*)$ in the preprocessing and the total number of vertices in the four chains is $O(n)$, this procedure can be done in $O(n)$ time by traversing the four chains in the arrangement $\mathcal{A}(B^* \cup R^*)$. We repeat this for every $0 \leqslant i \leqslant |B| - 2$, which takes $O(n^2)$ time in total. ◀

▶ **Theorem 22.** *We can solve t-MinRed in $O(n^2 \log n)$ time for $t \geqslant 3$.*

Proof. We give a reduction from t-MinRed to $(t+1)$-MaxBlue in $O(n)$ time, which completes the proof by Theorem 19. Let $P = B \cup R$ be a set of n points in the plane. Assume that there is an optimal solution for t-MinRed whose strips have a positive slope. The other case can be handled symmetrically.

We first compute an axis-parallel square Q that contains P in its interior. Let v_a and v_b be the upper-left and lower-right vertices of Q, respectively. Consider two translates Q_a and Q_b of Q, one with its lower-right corner at v_a and one with its upper-left corner at v_b. Let $v_1, \ldots, v_4$ and $v_5, \ldots, v_8$ denote the corners of Q_a and Q_b, respectively. For each v_i, we place n points on it. Let V be the set of $8n$ points placed on $v_1, \ldots, v_8$. Observe that the convex hull of V contains the convex hull of $B \cup R$. See Figure 10.

Let S_{opt} be an optimal solution for t-MinRed whose strips have positive slope and contain no point in V. Let $1 \leqslant k_{\mathsf{opt}} < |R|$ denote the number of points in R covered by S_{opt}. Let $\overline{S_{\mathsf{opt}}}$ be a set of $t+1$ strips that are parallel and disjoint to the strips of S_{opt} and whose union covers all points in $(P \cup V) \setminus S_{\mathsf{opt}}$. Since S_{opt} covers all points in B, $\overline{S_{\mathsf{opt}}}$ covers no points in B, and it covers $8n + |R| - k_{\mathsf{opt}}$ points in total, $8n$ from V and $|R| - k_{\mathsf{opt}}$ from R.

We now construct an instance of $(t+1)$-MaxBlue with two point sets $E = V \cup R$ and $F = B$. Here, an optimal solution consists of $t+1$ parallel strips whose union contains the maximum number of points of E while containing no points of F. Let S be an optimal solution for the $(t+1)$-MaxBlue instance. Let $\overline{S}$ be a set of t strips such that each strip is the region between two consecutive strips of S.

We show that $\overline{S}$ is an optimal solution for t-MinRed for the instance. We first show three observations on S: (1) every point in V is covered by S, (2) every point in $F = B$ is covered by $\overline{S}$, and (3) every point in R is covered by either S or $\overline{S}$.

For (1), suppose there is a point of V not covered by S. Then S covers at most $7n$ points of V, and hence at most $7n + |R|$ points of $E = V \cup R$. This contradicts the optimality of S, because $\overline{S_{\mathsf{opt}}}$ covers $8n + |R| - k_{\mathsf{opt}} > 7n + |R|$ points of E while covering no point of $F = B$.

For (2), suppose there is a point $p \in F = B$ not covered by $\overline{S}$. By definition, p is not covered by S, so p lies outside $S \cup \overline{S}$. This implies that some vertex in V is not covered by S, contradicting (1). See Figure 10(b).

For (3), suppose there is a point $q \in R$ covered by neither S nor $\overline{S}$. Then we could enlarge an outermost strip of S so that it contains q without including any point of $F = B$, contradicting the optimality of S.

By (2), $\overline{S}$ covers all points of $F = B$. Let k be the number of points of R covered by $\overline{S}$. Suppose $\overline{S}$ is not optimal for t-MinRed, that is, $k > k_{\mathsf{opt}}$. By (3), S covers $|R| - k$ points of R, and by (1), it covers all $8n$ points of V. Thus S covers $8n + |R| - k$ points of $E = V \cup R$, which is less than $8n + |R| - k_{\mathsf{opt}}$, contradicting the optimality of S_{opt}. ◀

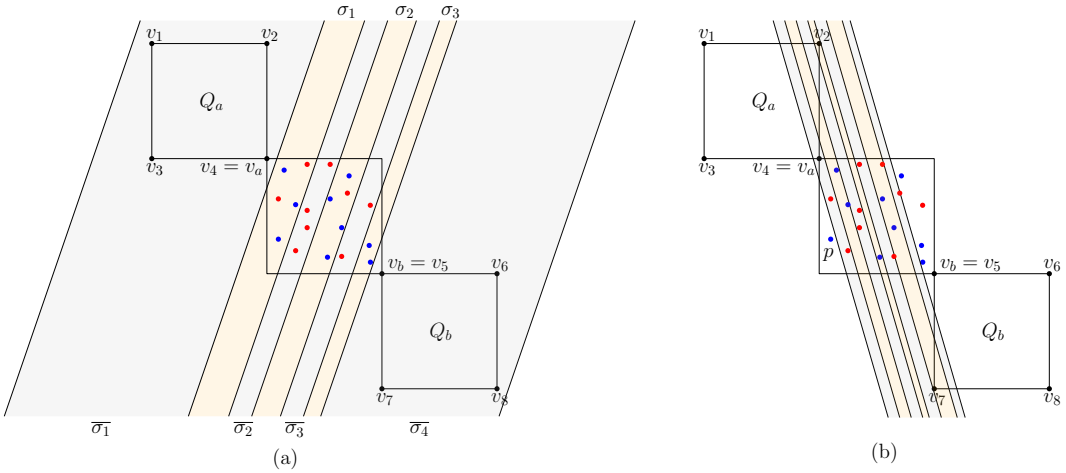

Figure 10 (a) $S_{\mathsf{opt}} = \{\sigma_1, \sigma_2, \sigma_3\}$ and $\overline{S_{\mathsf{opt}}} = \{\overline{\sigma_1}, \overline{\sigma_2}, \overline{\sigma_3}, \overline{\sigma_4}\}$ (b) S consists of the gray strips, and $\overline{S}$ consists of the yellow strips. If $S \cup \overline{S}$ does not cover p, there is a vertex in V not covered by S.

References

1. Pankaj K. Agarwal, Boris Aronov, Timothy M. Chan, and Micha Sharir. On levels in arrangements of lines, segments, planes, and triangles. *Discrete & Computational Geometry*, 19(3):315–331, 1998. doi:10.1007/PL00009348.
2. Noga Alon and E Györi. The number of small semispaces of a finite set of points in the plane. *Journal of Combinatorial Theory, Series A*, 41(1):154–157, 1986. doi:10.1016/0097-3165(86)90122-6.
3. Esther M. Arkin, Ferran Hurtado, Joseph S. B. Mitchell, Carlos Seara, and Steven S. Skiena. Some lower bounds on geometric separability problems. *International Journal of Computational Geometry & Applications*, 16(01):1–26, 2006. doi:10.1142/S0218195906001902.
4. Jonathan Backer and J. Mark Keil. The mono-and bichromatic empty rectangle and square problems in all dimensions. In *Proceedings of the 9th Latin American Symposium on Theoretical Informatics (LATIN 2010)*, pages 14–25. Springer, 2010. doi:10.1007/978-3-642-12200-2_3.
5. Mark de Berg, Otfried Cheong, Marc Van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 2008.
6. Steven Bitner, Yam Cheung, and Ovidiu Daescu. Minimum separating circle for bichromatic points in the plane. In *Proceedings of the 7th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD 2010)*, pages 50–55. IEEE, 2010.
7. Gerth Stølting Brodal and Riko Jacob. Dynamic planar convex hull. In *Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS 2002)*, pages 617–626. IEEE Computer Society, 2002. doi:10.1109/SFCS.2002.1181985.
8. Timothy M. Chan. Remarks on k-level algorithms in the plane. Manuscript, 1999.

9 Timothy M. Chan. Low-dimensional linear programming with violations. *SIAM Journal on Computing*, 34(4):879–893, 2005. doi:10.1137/S0097539703439404.

10 Kenneth L. Clarkson and Peter W. Shor. Applications of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4(5):387–421, 1989. doi:10.1007/BF02187740.

11 T. K. Dey. Improved bounds for planar k -sets and related problems. *Discrete & Computational Geometry*, 19(3):373–382, March 1998. doi:10.1007/PL00009354.

12 David P. Dobkin, Dimitrios Gunopulos, and Wolfgang Maass. Computing the maximum bichromatic discrepancy, with applications to computer graphics and machine learning. *Journal of Computer and System Sciences*, 52(3):453–470, 1996. doi:10.1006/JCSS.1996.0034.

13 Jonathan Eckstein, Peter L. Hammer, Ying Liu, Mikhail Nediak, and Bruno Simeone. The maximum box problem and its application to data analysis. *Computational Optimization and Applications*, 23(3):285–298, 2002. doi:10.1023/A:1020546910706.

14 Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10. Springer Science & Business Media, 1987. doi:10.1007/978-3-642-61568-9.

15 Hazel Everett, Jean-Marc Robert, and Marc Van Kreveld. An optimal algorithm for computing ($\leqslant K$)-levels, with applications. *International Journal of Computational Geometry & Applications*, 6(03):247–261, 1996.

16 Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry*, 5(3):165–185, 1995. doi:10.1016/0925-7721(95)00022-2.

17 Erwin Glazenburg, Thijs van der Horst, Tom Peters, Bettina Speckmann, and Frank Staals. Robust bichromatic classification using two lines. In *Proceedings of the 35th International Symposium on Algorithms and Computation (ISAAC 2024)*, volume 322, pages 33:1–33:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ISAAC.2024.33.

18 Dirk Helbing. Modeling multi-lane traffic flow with queuing effects. *Physica A: Statistical Mechanics and its Applications*, 242(1-2):175–194, 1997.

19 Michael E Houle and Godfried T Toussaint. Computing the width of a set. In *Proceedings of the first annual symposium on Computational geometry*, pages 1–7, 1985.

20 Michael F. Houle. Algorithms for weak and wide separation of sets. *Discrete Applied Mathematics*, 45(2):139–159, 1993. doi:10.1016/0166-218X(93)90057-U.

21 Ferran Hurtado, Marc Noy, Pedro A. Ramos, and Carlos Seara. Separating objects in the plane by wedges and strips. *Discrete Applied Mathematics*, 109(1-2):109–138, 2001. doi:10.1016/S0166-218X(00)00230-4.

22 Md Nazmuzzaman Khan, Adibuzzaman Rahi, Veera P Rajendran, Mohammad Al Hasan, and Sohel Anwar. Real-time crop row detection using computer vision-application in agricultural robots. *Frontiers in Artificial Intelligence*, 7:1435686, 2024. doi:10.3389/FRAI.2024.1435686.

23 Adam Kimura, Jon Scholl, James Schaffranek, Matthew Sutter, Andrew Elliott, Mike Strizich, and Glen David Via. A decomposition workflow for integrated circuit verification and validation. *Journal of Hardware and Systems Security*, 4(1):34–43, 2020. doi:10.1007/S41635-019-00086-6.

24 Sukanya Maji and Sanjib Sadhu. Geometric objects covering all red points and minimum blue points. *Turkish Journal of Engineering*, 9(1):47–55, 2025.

25 Jiri Matousek. *Lectures on Discrete Geometry*, volume 212. Springer Science & Business Media, 2013.

26 Nimrod Megiddo. On the complexity of polyhedral separability. *Discrete & Computational Geometry*, 3(4):325–337, 1988. doi:10.1007/BF02187916.

27 Joseph O'rourke, S. Rao Kosaraju, and Nimrod Megiddo. Computing circular separability. *Discrete & Computational Geometry*, 1(2):105–113, 1986.

28 Carlos Seara. *On geometric separability.* PhD thesis, Univ. Politecnica de Catalunya, 2002.

29 Csaba D. Toth, Joseph O'Rourke, and Jacob E. Goodman. *Handbook of Discrete and Computational Geometry.* CRC press, 2017.

30 Géza Tóth. Point sets with many k-sets. *Discrete & Computational Geometry*, 26(2):187–194, 2001. doi:10.1007/S004540010022.

31 Godfried T Toussaint. Solving geometric problems with the rotating calipers. In *Proc. IEEE Melecon*, volume 83, page A10, 1983.

32 Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

33 Chee-Keng Yap. *Fundamental problems of algorithmic algebra*, volume 49. Oxford University Press Oxford, 2000.