On the Parameterized Complexity of Red-Blue Points Separation*†

Édouard Bonnet[1], Panos Giannopoulos[2], and Michael Lampis[3]

1 **Middlesex University, Department of Computer Science, London, UK**
edouard.bonnet@dauphine.fr
2 **Middlesex University, Department of Computer Science, London, UK**
p.giannopoulos@mdx.ac.uk
3 **Université Paris-Dauphine, PSL Research University, CNRS, LAMSADE, Paris, France**
michail.lampis@dauphine.fr

Abstract

We study the following geometric separation problem: Given a set $\mathcal{R}$ of red points and a set $\mathcal{B}$ of blue points in the plane, find a minimum-size set of lines that separate $\mathcal{R}$ from $\mathcal{B}$. We show that, in its full generality, parameterized by the number of lines k in the solution, the problem is unlikely to be solvable significantly faster than the brute-force $n^{O(k)}$-time algorithm, where n is the total number of points. Indeed, we show that an algorithm running in time $f(k)n^{o(k/\log k)}$, for any computable function f, would disprove ETH. Our reduction crucially relies on selecting lines from a set with a large number of different slopes (i.e., this number is not a function of k).

Conjecturing that the problem variant where the lines are required to be axis-parallel is FPT in the number of lines, we show the following preliminary result. Separating $\mathcal{R}$ from $\mathcal{B}$ with a minimum-size set of axis-parallel lines is FPT in the size of either set, and can be solved in time $O^*(9^{|\mathcal{B}|})$ (assuming that $\mathcal{B}$ is the smallest set).

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases red-blue points separation, geometric problem, W[1]-hardness, FPT algorithm, ETH-based lower bound

Digital Object Identifier 10.4230/LIPIcs.IPEC.2017.8

1 Introduction

We study the parameterized complexity of the following Red-Blue Separation problem: Given a set $\mathcal{R}$ of red points and a set $\mathcal{B}$ of blue points in the plane and a positive integer k, find a set of at most k lines that together separate $\mathcal{R}$ from $\mathcal{B}$ (or report that such a set does not exist). Separation here means that each cell in the arrangement induced by the lines in the solution is either monochromatic, i.e., contains points of one color only, or empty. Equivalently, $\mathcal{R}$ is separated from $\mathcal{B}$ if every straight-line segment with one endpoint in $\mathcal{R}$ and the other one in $\mathcal{B}$ is intersected by at least one line in the solution. Note here that we opt for *strict* separation that is, no point in $\mathcal{R} \cup \mathcal{B}$ is on a separating line. Let $n := |\mathcal{R} \cup \mathcal{B}|$.

The variant where the separating lines sought must be axis-parallel will be simply referred to as Axis-Parallel Red-Blue Separation.

* Research partially supported by EPSRC grant EP/N029143/1.
† A full version of the paper is available at https://arxiv.org/abs/1710.00637.

Apart from being interesting in its own right, Red-Blue Separation is also directly motivated by the problem of univariate discretization of continuous variables in the context of machine learning [4, 8]. For example, its two-dimensional version models problem instances with decision tables of two real-valued attributes and a binary decision function. The lines to be found represent cut points determining a partition of the values into intervals and one opts for a minimum-size set of cuts that is consistent with the given decision table. The problem is also known as *minimum linear classification*; see [9] for an application in signal processing. For the case where $k = 1$ and $k = 2$, Red-Blue Separation is solvable in $O(n)$ and $O(n \log n)$ time respectively [7]. When k is part of the input, it is known to be NP-hard [10] and APX-hard [2] even for the axis-parallel variant. The latter also admits a 2-approximation [2].

Results. We first show that Red-Blue Separation is W[1]-hard in the solution size k and that it cannot be solved in $f(k)n^{o(k/\log k)}$ time (for any computable function f) unless ETH fails. Our reduction is from Structured 2-Track Hitting Set, see Section 2, which has been recently used for showing hardness for another classical geometric optimization problem [1]. Then, in Section 3, we show that Axis-Parallel Red-Blue Separation is FPT in the size of either of $\mathcal{R}$ and $\mathcal{B}$. Our algorithm is simple and is based on reducing the problem to $9^{|\mathcal{B}|+2}$ instances of 2-SAT (assuming, w.l.o.g., that $\mathcal{B}$ is the smallest set).

Related work. The following monochromatic points separation problem has also been studied: Given a set of points in the plane, find a smallest set of lines that separates every point from every other point in the set (i.e., each cell in the induced arrangement must contain at most one point). It has been shown to be NP-hard [5], APX-hard [2] and, in the axis-parallel case, to admit a 2-approximation [2]. Very recently, the problem has been also shown to admit an $\mathrm{OPT} \log \mathrm{OPT}$-approximation [6]. Note here that it is trivially FPT in the number of lines, as the number of cells in the arrangement of k lines is at most $\Theta(k^2)$. For results on several other related separation problems, see [7, 3].

2 Parameterized hardness for arbitrary slopes

We show that Red-Blue Separation is unlikely to be FPT with respect to the number of lines k and establish that, unless the ETH fails, the $n^{O(k)}$-time brute-force algorithm is almost optimal. We reduce from Structured 2-Track Hitting Set [1], see below.

For positive integers x, y, let $[x]$ be the set of integers between 1 and x, and $[x, y]$ the set of integers between x and y. For a totally ordered (finite) set X, an *X-interval* is any subset of X of consecutive elements. In the 2-Track Hitting Set problem, the input consists of an integer k, two totally ordered ground sets A and B of the same cardinality, and two sets $\mathcal{S}_A$ of A-intervals and $\mathcal{S}_B$ of B-intervals. The elements of A and B are in one-to-one correspondence $\phi : A \to B$ and each pair $(a, \phi(a))$ is called a *2-element*. The goal is to decide if there is a set S of k 2-elements such that the first projection of S is a hitting set of $\mathcal{S}_A$, and the second projection of S is a hitting set of $\mathcal{S}_B$. We will refer to the interval systems $(A, \mathcal{S}_A)$ and $(B, \mathcal{S}_B)$ as track A and track B.

Structured 2-Track Hitting Set (S2-THS for short) is the same problem with color classes over the 2-elements and a restriction on the one-to-one mapping ϕ. Given two integers k and t, A is partitioned into $(C_1, C_2, \ldots, C_k)$ where $C_j = \{a_1^j, a_2^j, \ldots, a_t^j\}$ for each $j \in [k]$. A is ordered: $a_1^1, a_2^1, \ldots, a_t^1, a_1^2, a_2^2, \ldots, a_t^2, \ldots, a_1^k, a_2^k, \ldots, a_t^k$. We define $C'_j := \phi(C_j)$ and $b_i^j := \phi(a_i^j)$ for all $i \in [t]$ and $j \in [k]$. We now impose that ϕ is such that, for each

$j \in [k]$, the set C'_j is a B-interval. That is, B is ordered: $C'_{\sigma(1)}, C'_{\sigma(2)}, \ldots, C'_{\sigma(k)}$ for some permutation on $[k]$, $\sigma \in \mathfrak{S}_k$. For each $j \in [k]$, the order of the elements within C'_j can be described by a permutation $\sigma_j \in \mathfrak{S}_t$ such that the ordering of C'_j is: $b^j_{\sigma_j(1)}, b^j_{\sigma_j(2)}, \ldots, b^j_{\sigma_j(t)}$. In what follows, it will be convenient to see an instance of S2-THS as a tuple $\mathcal{I} = (k \in \mathbb{N}, t \in \mathbb{N}, \sigma \in \mathfrak{S}_k, \sigma_1 \in \mathfrak{S}_t, \ldots, \sigma_k \in \mathfrak{S}_t, \mathcal{S}_A, \mathcal{S}_B)$, where $\mathcal{S}_A$ is a set of A-intervals and $\mathcal{S}_B$ is a set of B-intervals. We denote by $[a^j_i, a^{j'}_{i'}]$ (resp. $[b^j_i, b^{j'}_{i'}]$) all the elements $a \in A$ (resp. $b \in B$) such that $a^j_i \leq_A a \leq_A a^{j'}_{i'}$ (resp. $b^j_i \leq_B b \leq_B b^{j'}_{i'}$).

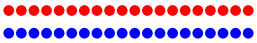
If one deconstructs S2-THS, one finds intervals, a permutation of the color classes σ, and k permutations σ_j's of the elements within the classes. Intervals, thanks to their geometric nature, can be realized by two red points which have to be separated from a diagonal of blue points (see Figure 2), while permutation σ, being on k elements, can be designed straightforwardly without blowing-up the size of the solution (see Figure 3). For these gadgets, we would like to force the chosen lines to be axis-parallel. We obtain this by surrounding them with *long alleys* made off long red paths parallel and next to long blue paths (see Figure 1). The main challenge is to get the permutations σ_j's on t elements. To attain this, we match a selected line L_i (corresponding to an element of index $i \in [t]$) to a specific angle α_i, which *leads* to the intended position of the element of index $\sigma_j(l) = i$, for some $l \in [t]$ (see Figure 4). Note that the depicted gadget actually links the element of index i to elements equal to or smaller than the element indexed at $\sigma_j(l)$. By combining two of these gadgets we can easily obtain only the intended position (see Figure 5).

▶ **Theorem 1.** Red-Blue Separation *is $W[1]$-hard w.r.t. the number of lines k, and unless ETH fails, cannot be solved in time $f(k)n^{o(k/\log k)}$ for any computable function f.*

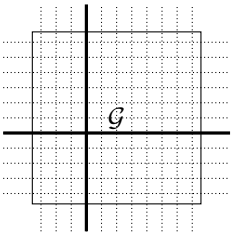
Proof. We reduce from S2-THS, which is $W[1]$-hard and has the above lower bound under ETH [1]. Let $\mathcal{I} = (k \in \mathbb{N}, t \in \mathbb{N}, \sigma \in \mathfrak{S}_k, \sigma_1 \in \mathfrak{S}_t, \ldots, \sigma_k \in \mathfrak{S}_t, \mathcal{S}_A, \mathcal{S}_B)$ be an instance of S2-THS. We build an instance $\mathcal{J} = (\mathcal{R}, \mathcal{B}, 6k + 14)$ of Red-Blue Separation such that $\mathcal{I}$ is a YES-instance for S2-THS if and only if $\mathcal{R}$ and $\mathcal{B}$ can be separated with $6k + 14$ lines.

The points in $\mathcal{R}$ and $\mathcal{B}$ will have rational coordinates. More precisely, most points will be pinned to a z-by-z grid where z is polynomial in the size of $\mathcal{I}$. The rest will have rational coordinates with nominator and denominator polynomial in z. Let Γ be the z-by-z grid corresponding to the set of points with coordinates in $[z] \times [z]$. We call *horizontally* (resp. vertically) *consecutive points* a set of points of Γ with coordinates $(a, y), (a+1, y), \ldots (b-1, y), (b, y)$ for $a, b, y \in [z]$ and $a < b$ (resp. $(x, a), (x, a+1), \ldots (x, b-1), (x, b)$ for $a, b, x \in [z]$ and $a < b$). We denote those points by $\mathcal{C}(a \to b, y)$ (resp. $\mathcal{C}(x, a \to b)$).

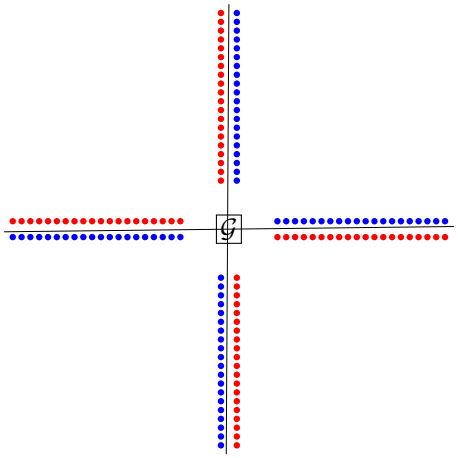
Long alley gadgets. In the gadgets encoding the intervals (see next paragraph), we will need to restrict the selected separating lines to be almost horizontal or almost vertical. To enforce that, we use the *long alley* gadgets. A *horizontal long alley* gadget is made of ℓ horizontally consecutive red points $\mathcal{C}(a \to a + \ell - 1, y)$ and ℓ horizontally consecutive blue points $\mathcal{C}(a \to a + \ell - 1, y')$ with $a, a + \ell - 1, y \neq y' \in [z]$ (see Figure 1a). A *vertical long alley* is defined analogously. Long alleys are called so because $\ell \gg |y - y'|$ thus, separating the red points from the blue points of a horizontal (resp. vertical) long alley with a budget of only one line, requires the line to be almost horizontal (resp. vertical). The use of the long alleys will be the following. Let $\mathcal{G}$ be a gadget for which we wish the separating lines to be almost horizontal or vertical. Say, $\mathcal{G}$ occupies a g-by-g subgrid of Γ (with $g \ll z$). We place four long alley gadgets to the left, top, right, and bottom of $\mathcal{G}$: horizontal ones to the left and right, vertical ones to the top and bottom (as depicted in Figure 1c). The left horizontal (resp. bottom vertical) long alley starts at the x-coordinate (resp. y coordinate)



(a) A horizontal long alley. Separating this subset of points with one line requires the line to be almost horizontal.



(b) Zoom in gadget $\mathcal{G}$. The horizontal (resp. vertical) lines are *entering* the gadget to the left (resp. at the top) and *exiting* it to the right (resp. at the bottom) with almost the same y-coordinates (resp. x-coordinates). Possible lines are thin dotted while an actual choice of two lines is shown in bold.



(c) We put four *long alleys* to the left, top, right, and bottom of gadget $\mathcal{G}$ where we want the selected lines to be almost axis parallel.

Figure 1 The long alley gadget and its use in combination with another gadget.

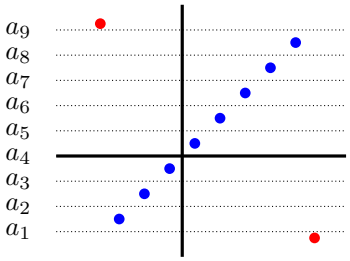
of 1, whereas the right horizontal (resp. top vertical) long alley ends at the x-coordinate (resp. y coordinate) of z; see Figure 5, where the long alleys are depicted by thin rectangles.

Note that we will not surround each and every gadget of the construction by four long alleys. At some places, it will indeed be crucial that the lines can have arbitrary slopes.

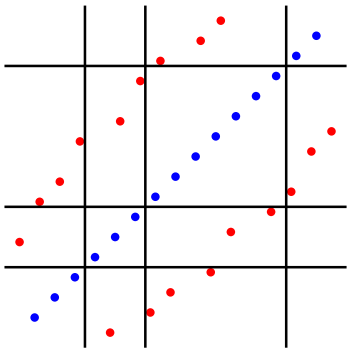
Interval gadgets and encoding track A. The elements of A are represented by a diagonal of $kt-1$ blue points. More precisely, we add the points $(x_0^A, y_0^A), (x_0^A+4, y_0^A+4), (x_0^A+8, y_0^A+8), \ldots, (x_0^A+4kt-8, y_0^A+4kt-8)$ to $\mathcal{B}$ for some offset $x_0^A, y_0^A \in [z]$ that we will specify later. We think those points as going from the first (x_0^A, y_0^A) to the last $(x_0^A+4kt-8, y_0^A+4kt-8)$. An almost horizontal (resp. vertical) line just below (resp. just to the left of) the s-th blue point of this diagonal translates as selecting the s-th element of A in the order fixed by $\leq_A$. The almost horizontal (resp. vertical) line just above (resp. just to the right of) the last blue point corresponds to selecting the kt-th, i.e., last, element of A.

For each interval $[a_i^j, a_{i'}^{j'}]$ in $\mathcal{S}_A$ (for some $i, i' \in [k]$, $j, j' \in [t]$), that is, the interval between the $s := ((j-1)t+i)$-th and the $s' := ((j'-1)t+i')$-th elements of A, we add two red points: one at $(x_0^A+4s-7, y_0^A+4s'-5)$ and one at $(x_0^A+4s'-5, y_0^A+4s-7)$ (see Figure 2a for one interval gadget and Figure 2b for track A). Let $R([a_i^j, a_{i'}^{j'}])$ be this pair of red points. Informally, one red point has its projection along the x-axis just to the left of the s-th blue point and its projection along the y-axis just above the s'-th blue point; the other one has its projection along the x-axis just to the right of the s'-th blue point and its projection along the y-axis just below the s-th blue point. For technical reasons, we add, for every $j \in [k]$, the pair $R([a_1^j, a_t^j])$ encoding the interval formed by all the elements of the j-th color class of A. Adding these intervals to $\mathcal{S}_A$ does not constrain the problem more.

We surround this encoding of track A, which we denote by $\mathcal{G}(A)$, with $4k$ long alleys, whose width is $4t-4$, from x-coordinates $x_0^A+4(j-1)t-2$ to $x_0^A+4jt-6$ for vertical alleys (from y-coordinates $y_0^A+4(j-1)t-2$ to $y_0^A+4jt-6$ for horizontal alleys). We



(a) The interval gadget corresponding to $[a_1, a_9] = \{a_1, \ldots, a_9\}$. In thin dotted, the mapping between elements and potential lines. In bold, the choice of the lines corresponding to picking a_4. If one wants to separate these points with two lines, one almost horizontal and one almost vertical, the choice of the former imposes the latter.



(b) The interval gadgets put together. A representation of one track. Separating these points with the fewest axis-parallel lines requires taking the horizontal and vertical lines associated to a minimum hitting set.

Figure 2 To the left, one interval. To the right, several put together to form one track.

alternate red-blue[1] alleys and blue-red alleys for two contiguous alleys so that there is no need to separate one from the other. We start with a red-blue alley for the left horizontal and top vertical groups of alleys, and with a blue-red alley for the right horizontal and bottom vertical. This last detail is not in any way crucial but permits the construction to be defined uniquely and consistently with the choices of Figure 1c. This, together with the description of long alleys in the previous paragraph, fully defines the $4k$ long alleys (see Figure 5).

The general intention is that in order to separate those two red points from the blue diagonal with a budget of two almost axis-parallel lines, one should take two lines (one almost horizontal and one almost vertical) corresponding to the selection of the same element of A which hits the corresponding interval. In particular, taking two almost horizontal lines (resp. two almost vertical lines) is made impossible due to those vertical (resp. horizontal) long alleys. More precisely, the intended pairs of lines separating the red points $R([a_i^j, a_{i'}^{j'}])$ from the blue diagonal are of the form $x = x_0^A + 4\hat{s} - 6, y = y_0^A + 4\hat{s} - 6$ for $\hat{s} \in [s, s']$. Furthermore, the $4k$ long alleys force a pair of (almost) horizontal and vertical lines corresponding to one element per color class to be taken.

For any $s \in [tk]$, $i \in [t]$, and $j \in [k]$, such that $s = (j-1)t + i$, let $\mathrm{HL}(s)$ be the horizontal line of equation $y = y_0^A + 4s - 6$ and $\mathrm{VL}(s)$ the vertical line of equation $x = x_0^A + 4s - 6$. They *correspond* to selecting a_i^j, the i-th element in the j-th color class of A. The goal of the remaining gadgets is to ensure that when the lines $\mathrm{HL}(s)$ and $\mathrm{VL}(s)$ (with $s = (j-1)t + i$) are chosen, additional lines corresponding to selecting element b_i^j of B have to be expressly selected. We define $\mathrm{HL} := \{\mathrm{HL}(s) \mid s \in [tk]\}$ and $\mathrm{VL} := \{\mathrm{VL}(s) \mid s \in [tk]\}$.

Encoding inter-class permutation σ. To encode the permutation σ of the k color classes of $\mathcal{I}$, we allocate a square subgrid of the same dimension as the space used for the encoding of track A, roughly $4tk$-by-$4tk$, and we place it to the right of A right as depicted in Figure 5. This square subgrid is naturally and regularly split into k^2 smaller square subgrids of equal dimension (roughly $4t$-by-$4t$). This decomposition can be seen as the k color classes of

[1] i.e., for horizontal (resp. vertical) alleys, the red points are above (resp. to the left of) the blue points.

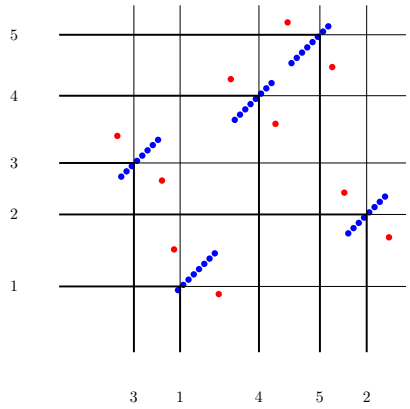


Figure 3 Encoding permutation $\sigma = 31452$. The choices within the five color classes are transferred from almost horizontal lines to almost vertical ones. This way, we obtain the desired reordering of the color classes.

$\mathcal{I}$, or equivalently, the k-by-k crossing[2] obtained by drawing horizontal lines between two contiguous horizontal long alleys and vertical lines between two contiguous vertical long alleys. We only put points in exactly one smaller square subgrid per column and per row. Let $\sigma := \sigma(1)\sigma(2)\dots\sigma(k)$ and $\mathrm{Cell}(a,b)$ be the smaller square subgrid in the a-th row and b-th column of the k-by-k crossing. For each $j \in [k]$, we put in $\mathrm{Cell}(j,\sigma(j))$ a diagonal of $t-1$ blue points and two red points corresponding to the full interval $[a_1^j, a_t^j]$ (see Figure 3). We denote by $\mathcal{G}(\sigma)$ those sets of red and blue points in the encoding of σ. We surround $\mathcal{G}(\sigma)$ by $2k$ vertical long alleys similar to the $2k$ long alleys surrounding $\mathcal{G}(A)$. Notice that $\mathcal{G}(\sigma)$ and $\mathcal{G}(A)$ share the same $2k$ surrounding horizontal long alleys.

The way the gadget $\mathcal{G}(\sigma)$ works is quite intuitive. Given k choices of horizontal lines originating from a separation in $\mathcal{G}(A)$ and a budget of k extra lines for the separation within $\mathcal{G}(\sigma)$, the only option is to copy with the vertical line the choice of the horizontal line. It results in a vertical propagation of the initial choices accompanied by the desired reordering of the *color classes*. The vertical line matching the choice of $\mathrm{HL}(s)$ in the corresponding cell of $\mathcal{G}(\sigma)$ is denoted by $\mathrm{VL}'(s)$. Let $\mathrm{VL}' := \{\mathrm{VL}'(s) \mid s \in [tk]\}$. Note that corresponding lines in VL and in VL$'$ have a different order from left to right.

Encoding of the intra-class permutations σ_j's and track B. If the encoding of permutation σ is conceptually simple, the number of intended lines separating red and blue points in $\mathcal{G}(\sigma)$ has to be linear in the number of permuted elements. Since we wish to encode a permutation σ_j (for every $j \in [k]$) on t elements, we cannot use the same mechanism as it would blow-up our parameter dramatically and would not result in an FPT reduction.

For the gadget $\mathcal{G}_{\approx v}(\sigma_j)$ partially encoding the permutation σ_j, we will crucially use the fact that separating lines can have arbitrary slopes. Slightly to the right (at distance at least ℓ) of the vertical line bounding the right end of $\mathcal{G}(\sigma)$ and far in the south direction, we place a gadget $\mathcal{G}(B)$ encoding track B similarly to the encoding of track A up to some symmetry that we will make precise later. We also incline the whole encoding of track B with a small, say 5, degree angle, in a way that its top-left corner is to the right of its bottom-left corner. We round up the real coordinates that this rotation incurs to rationals at distance

[2] we use this term informally to avoid confusion with what we have been calling *grids* so far.

less than, say, $(kt)^{-10}$. We denote by $\hat{v}$ the distance along the y-axis between $\mathcal{G}(\sigma)$ and $\mathcal{G}(B)$. Eventually $\hat{v}$ will be chosen much larger than $\Theta(kt)$, which is the size of $\mathcal{G}(A)$, $\mathcal{G}(B)$, $\mathcal{G}(\sigma)$. Below $\mathcal{G}(\sigma)$ at a distance $2\hat{v}$ along the y-axis, we place gadgets $\mathcal{G}_{\approx_v}(\sigma_j)$'s; from left to right, we place $\mathcal{G}_{\approx_v}(\sigma_{\sigma(1)})$, $\mathcal{G}_{\approx_v}(\sigma_{\sigma(2)})$, $\ldots$, $\mathcal{G}_{\approx_v}(\sigma_{\sigma(k)})$ such that for every $i \in [k]$, $G_{\approx_v}(\sigma_{\sigma(i)})$ falls below the i-th column of the k-by-k crossing of $\mathcal{G}(\sigma)$. Gadgets $\mathcal{G}_{\approx_v}(\sigma_j)$'s are represented by small round shapes in Figure 5. Notwithstanding what is drawn on the overall picture, the $\mathcal{G}_{\approx_v}(\sigma_j)$'s can be all placed at the same y-coordinates. Let $y_1 := y_0 - 2\hat{v}$ (the exact value of y_1 is not crucial). Also, we represent track B slanted by a 45 degree angle, instead of the actual 5 degree angle, to be able to fit everything on one page and convey the main ideas of the construction. In general, for the figure to be readable, the true proportions are not respected. The size of every gadget is much smaller than the distance between two different groups of gadgets, so that every line *entering* a gadget traverses it in an axis-parallel fashion.

Gadget $\mathcal{G}_{\approx_v}(\sigma_j)$ is built as follows. For each $i \in [t]$ and $j \in [k]$, we draw a fictitious points p_i^j corresponding to the intersection of a close to vertical line corresponding to picking element b_i^j in gadget $\mathcal{G}(B)$ with the bottom end of $\mathcal{G}(B)$. From left to right, the p_i^j's have the same order as the b_i^j's in $(B, \leq_B)$. For every $s = (j-1)t + i$ (with $j \in [k]$ and $i \in [t]$), let q_i^j be the point of y-coordinate y_1 on the line $\mathrm{VL}'(s)$. We define the line $\mathrm{SL}(s)$ as going through p_i^j and q_i^j, and set $\mathrm{SL} := \{\mathrm{SL}(s) \mid s \in [tk]\}$. We add two blue points just to the left and just to the right of q_i^j at distance $\epsilon := z^{-10}$. We add two blue points on line $\mathrm{SL}(s)$, one to the left of q_i^j and one to the right of q_i^j. Finally, we place two red points for each $\mathcal{G}_{\approx_v}(\sigma_j)$ at the bottom-left and top-right of the gadget (see Figure 4). In the figure, the lines in SL form a large angle with the y-axis, while in fact they are quite close to a 5 degree angle and behave like *relatively vertical*[3] lines within $\mathcal{G}(B)$ (since $\mathcal{G}(B)$ is also inclined by 5 degrees).

Assuming that line $\mathrm{VL}'(s = (j-1)t + i)$ has been selected, it can be observed from Figure 4 that separating the red points from the blue points in $\mathcal{G}_{\approx_v}(\sigma_j)$ with a budget of one additional line requires to take a line crossing $\mathrm{VL}'(s)$ at (or very close to) q_i^j and with a higher or equal slope to $\mathrm{SL}(s)$. It is not quite what we wanted. What we achieved so far is only to link the *choice of* a_i^j with the *choice of an element smaller or equal to* b_i^j. We will use a symmetry $\mathcal{G}_{\approx_h}(\sigma_j)$ of gadget $\mathcal{G}_{\approx_v}(\sigma_j)$ to get the other inequality so that choosing some lines corresponding to a_i^j actually forces to take some lines corresponding to b_i^j.

We add a gadget $\mathcal{G}(\mathrm{id})$ below the $\mathcal{G}_{\approx_v}(\sigma_j)$'s. $\mathcal{G}(\mathrm{id})$ is obtained by mimicking $\mathcal{G}(\sigma)$ for the identity permutation. We surround $\mathcal{G}(\mathrm{id})$ by $2k$ new horizontal long alleys. The horizontal line matching the choice of $\mathrm{VL}'(s)$ in $\mathcal{G}(\mathrm{id})$ is denoted by $\mathrm{HL}'(s)$. At a distance $\hat{h} \approx \hat{v}/(cos(5°) \cdot sin(5°))$ to the right of $\mathcal{G}(\mathrm{id})$ we place gadgets $\mathcal{G}_{\approx_h}(\sigma_j)$'s analogously to the $\mathcal{G}_{\approx_v}(\sigma_j)$'s. The fictitious points p'^j_i, p_i^j used for the construction of the lines $\mathrm{SL}'(s)$, $\mathrm{SL}(s)$ are located at the right end of $\mathcal{G}(B)$ and ordered as B when read from top to bottom. The difference in the construction of $\mathcal{G}(B)$ from the B-intervals (compared to $\mathcal{G}(A)$ from the A-intervals) is that the diagonal of blue points go from the top-left corner to the bottom-right corner (instead of bottom-left to top-right). Similarly to our previous definitions, we define $\mathrm{HL}' := \{\mathrm{HL}'(s) \mid s \in [tk]\}$ and $\mathrm{SL}' := \{\mathrm{SL}'(s) \mid s \in [tk]\}$. The choice of $\hat{h}$ makes the lines of SL' form a close to 5 degree angle with the x-axis and so *enter* $\mathcal{G}(B)$ relatively horizontal.

Putting the pieces together. We already hinted at how the different gadgets are combined together. We choose the different typical values so that: $kt \ll \hat{v} < \hat{h} \ll z$. For instance, $\hat{v} := 100((kt)^2 + 1)$ and $z := 100(\hat{h}^5 + 1)$. An important and somewhat hidden consequence of z being much greater than $\hat{v}$ and $\hat{h}$ is that the bulk of the construction (say, all the gadgets

[3] By that, we mean that the lines are close to vertical for axes aligned with the encoding of track B.

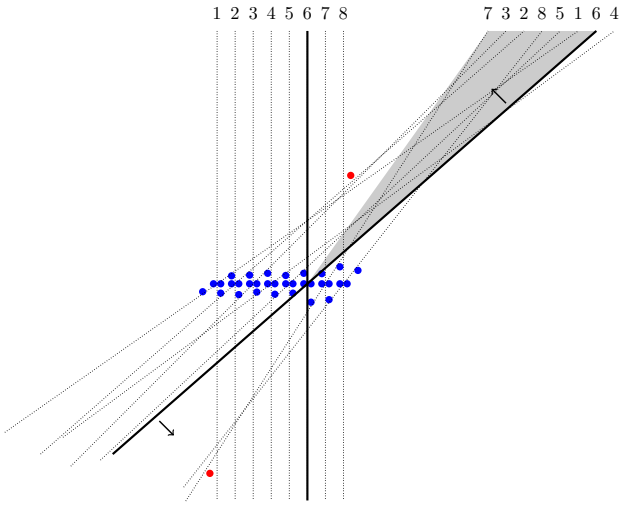


Figure 4 Half-encoding of permutation $\sigma_j = 73285164$ of the j-th color class. Observe that the choice of the, say, sixth almost horizontal candidate line only forces to take the slanted line depicted in bold or a line having the same intersection with the almost horizontal line but a larger slope. For the sake of legibility, the angles between the vertical lines and the slanted lines are exaggerated.

which are not long alleys) occupies a tiny space in the top-left corner of Γ. We set the length ℓ of the long alleys to $100(k^2+1)$. Point (x_0^A, y_0^A) corresponds to the bottom-left corner of the square in bold with a diagonal close to the overall top-left corner.

Slightly outside grid Γ we place 14 pairs of long alleys (7 horizontal and 7 vertical) of width, say, $(kt)^{-10}$ to force the 14 lines in bold in Figure 5. Note that, on the figure, we do not explicitly represent those long alleys but only the lines they force. The purpose of those new long alleys is to separate groups of gadgets from each other. Going clockwise all around the grid Γ, we alternate red-blue and blue-red alleys so that two consecutive long alleys do not need a further separation. The even parity of those alleys make this alternation possible. Each one of the 64 faces that those 14 lines define is called a *super-cell*.

The four lines in bold surrounding $\mathcal{G}(B)$ are close (say, at distance $10t$) to the north, south, west, and east ends of that gadget. On the four super-cells adjacent to the super-cell containing $\mathcal{G}(B)$, shown in gray, we place $4k$ long alleys each of width $4t-4$, analogously to what was done for $\mathcal{G}(A)$, but slanted by a 5 degree angle (as the gadget $\mathcal{G}(B)$). As for track A, these alleys force, relatively to the orientation of $\mathcal{G}(B)$, one *close to horizontal* line and one *close to vertical line* per color class. The long alleys are placed just next to $\mathcal{G}(B)$ and are not crossed by any other candidate lines.

This finishes the construction. We ask for a separation of $\mathcal{R}$ and $\mathcal{B}$ with $6k+14$ lines. The correctness of the reduction is deferred to the long version of the paper. ◀

3 FPT Algorithm Parameterized by Size of Smaller Set

We present a simple FPT algorithm for Axis-Parallel Red-Blue Separation parameterized by $\min\{|\mathcal{R}|, |\mathcal{B}|\}$. In the following, w.l.o.g., we assume that $\mathcal{B}$ is the smaller set.

▶ **Theorem 2.** *An optimal solution of* Axis-Parallel Red-Blue Separation *can be computed in* $O(n \log n + n|\mathcal{B}|9^{|\mathcal{B}|})$ *time.*

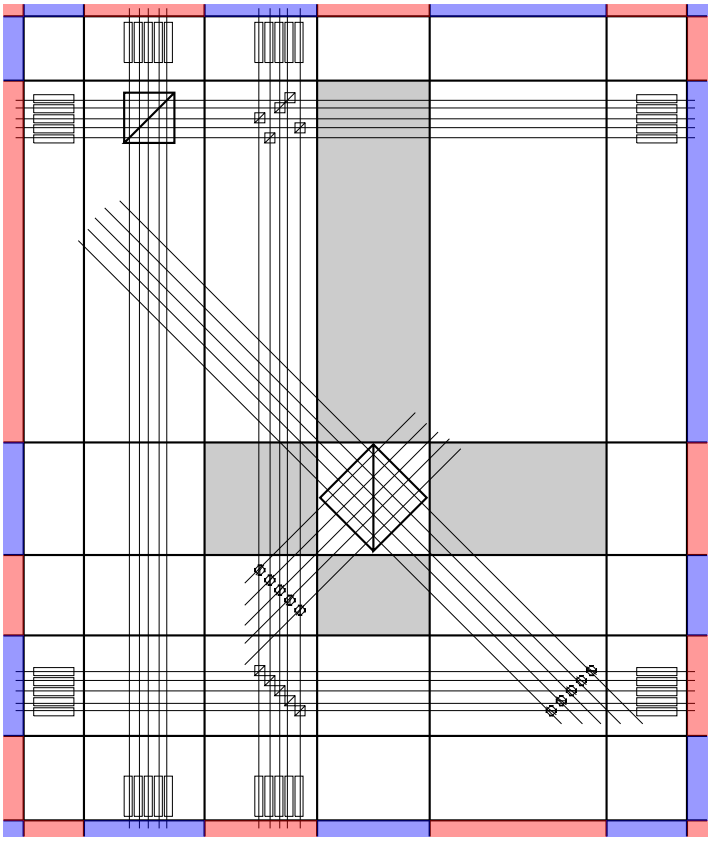
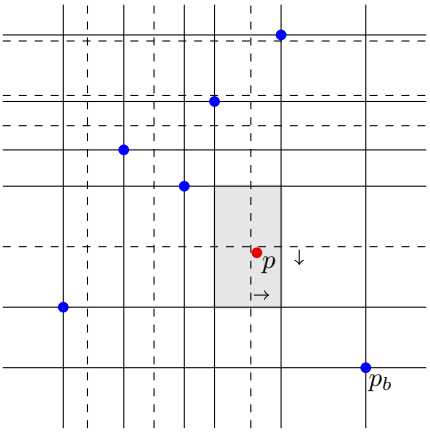


Figure 5 The overall picture. The thin rectangles are long alleys, the bold large squares with a diagonal are the encoding of track A, in the top left corner, and track B, slanted by 45 degrees (for the sake of fitting the whole construction on one page; in reality the encoding of B is only inclined by 5 degrees). The smaller squares with a diagonal are simple interval gadgets and the small round gadgets are half-encodings of the permutations σ_i's. The four super-cells filled with grey contain $4k$ long alleys slanted by 5 degrees. The (super-)cells filled with red and blue match their color, and are monochromatic once the 14 lines imposed by the outermost long alleys have been selected.

We first give a high-level description of the algorithm. It begins by subdividing the plane into at most $|\mathcal{B}| + 1$ vertical strips, each consisting of the area "between" two horizontally successive blue points, and at most $|\mathcal{B}| + 1$ horizontal strips, each consisting of the area "between" two vertically successive blue points (see Figure 6a). Since each strip can contain only red points in its interior, an optimal solution uses at most two lines inside a single strip (Lemma 5(a)). We can therefore guess (by exhaustive enumeration) the number of lines used in each strip in an optimal solution. This gives a running time of roughly $9^{|\mathcal{B}|}$. A second observation is that if an optimal solution uses two lines in a strip, these can be placed as far away from each other as possible (Lemma 5(b)). To complete the solution we must decide where to place the lines in strips that contain only one line of an optimal solution. We consider every pair of blue and red points whose separation may depend on the exact placement of these lines. The key idea is that the separation of two such points can be expressed as a 2-CNF constraint. If the upcoming formal exposition seems a bit more complicated than this informal idea, it is because we have to deal with points sharing the same x- or y-coordinates.



(a) The cell decomposition (solid lines), a guess of how S intersects it (dashed lines), and an interesting cell (in gray) for a point p_b (bottom-right corner). The red point p cannot be in the south-east quadrant of this cell which translates to the 2-clause $y_p^2 \vee \neg x_p^4$. Indeed, it should be that the horizontal line of S is below it or that the vertical line is to its right.

(b) Two consecutive red points in a horizontal strip $\mathcal{R}_h(i)$. If the corresponding line of S is below p, then it is also below p' which translates to $y_p^i \to y_{p'}^i$.

(c) Two consecutive red points in a vertical strip $\mathcal{R}_v(j)$. If the corresponding line of S is to the left of p, then it is also to the left of p' which translates to $x_p^i \to x_{p'}^i$.

Figure 6 Illustration of the algorithm and the two kinds of clauses of the 2-SAT instance.

We now proceed to a formal description of our algorithm, beginning with some definitions. For a point $p \in \mathbb{R}^2$, let $p(x)$ and $p(y)$ be its x-coordinate and y-coordinate, respectively. Also, let X, Y be the sets of x, y coordinates of the points in $\mathcal{B}$. In order to ease presentation later on, with a slight terminology abuse, we add $-\infty, +\infty$ to both X and Y. Let $X(i), Y(i)$ be the respective i-th elements of these sets in increasing order with $0 \leqslant i$, and let $k = |X| - 2$ and $l = |Y| - 2$; $k \leqslant |B|$ and $l \leqslant |B|$.

▶ **Definition 3.** The vertical strips are defined as $V_i = \{p \in \mathbb{R}^2 \mid X(i) \leqslant p(x) \leqslant X(i+1)\}$ for $i \in [0, k]$.

▶ **Definition 4.** The horizontal strips are defined as $H_i = \{p \in \mathbb{R}^2 \mid Y(i) \leqslant p(y) \leqslant Y(i+1)\}$ for $i \in [0, l]$.

The horizontal and vertical strips defined above essentially partition the plane into open monochromatic (red) or empty regions, while the boundaries of the strips may contain both red and blue points. As a result, we have the following properties of an optimal solution.

▶ **Lemma 5.** *(a) An optimal solution of* Axis-Parallel Red-Blue Separation *contains at most two lines in each horizontal or vertical strip. (b) In the case where a strip has two lines, these lines can be assumed to be placed in a way such that all red points in the interior of the strip lie between them.*

Proof of Theorem 2. We describe an FPT algorithm which first guesses how many lines an optimal solution uses in each strip and then produces a 2-SAT instance of size $O(|\mathcal{B}|n)$ in order to check if its guess is feasible. We assume that we have access to two lists containing the input points sorted lexicographically by their (x, y) and (y, x) coordinates.

Let S be some optimal solution. We first guess how many lines of S are in each horizontal and each vertical strip. Since, by Lemma 5, S contains at most two lines per strip, and there

are $l+1 \leqslant |\mathcal{B}|+1$ horizontal strips and $k+1 \leqslant |\mathcal{B}|+1$ vertical strips, there are at most $3^{|\mathcal{B}|+1}$ possibilities to guess from for each direction thus, $O(9^{|\mathcal{B}|})$ in total.

In what follows, we assume that we have fixed how many lines of S are in each strip. We give an algorithm deciding in polynomial time if such a specification gives a feasible solution. Since a specification fully determines the number of lines of a solution, the algorithm simply goes through all specifications and selects one with minimum cost among all feasible ones.

We produce a 2-SAT instance, which will be satisfiable if and only if a given specification is feasible. We first define the variables: for each horizontal strip H_i that contains exactly one line from S and for each red point $p \in H_i$, we define a variable y^i_p. Its informal meaning is "the line of S in H_i is below point p". When p lies on the upper (lower) boundary of H_i, y^i_p is set to true (false) by default. Similarly, for each vertical strip V_j that contains exactly one line from S and for each red point $p \in V_j$, we define a variable x^j_p. Its meaning is "the line of S in V_j is to the left of p". It is set to true (false) when p lies on the right (left) boundary of V_j. We have constructed $O(n)$ variables (at most four for each point of $\mathcal{R}$).

Next, we construct 2-CNF clauses imposing the informal meaning described. For each strip H_i that contains exactly one horizontal line from S and each pair of red points $p, p' \in H_i$ that are consecutive in lexicographic (y, x) order, we add the clause $(y^i_p \to y^i_{p'})$. We can skip pairs that have a point lying on the upper or lower boundary of H_i as the corresponding variable has been already set to true or false respectively and the clause is satisfied; see the description in the previous paragraph. Similarly, for each strip V_j that contains exactly one vertical line from S and each pair of red points $p, p' \in V_j$ that are consecutive in lexicographic (x, y) order, we add the clause $(x^j_p \to x^j_{p'})$; pairs that have a point lying on the left or right boundary of V_j do not produce any clauses. Given any solution, we can construct from its lines an assignment following the informal meaning described above that satisfies all clauses added so far, while from any satisfying assignment we can find lines according to the informal meaning. We call the $O(n)$ clauses constructed so far the coherence part of our instance.

What remains is to add some further clauses to our instance to ensure also that the solution is feasible, that is, it separates all pairs of red and blue points.

Consider a cell $C_{ij} = H_i \cap V_j$, where $i \in [0, l]$ and $j \in [0, k]$. A red point $p \in C_{ij}$ is called *C_{ij}-separable* for a point $p_b \in \mathcal{B}$, if p can be separated from p_b by a vertical or horizontal line running through the interior of C_{ij}. We will sometimes call p just separable when C_{ij} and p_b are obvious from the context. We say that C_{ij} is *interesting* for a point $p_b \in \mathcal{B}$ if the following conditions hold: (i) C_{ij} contains at least one red point that is C_{ij}-separable for p_b; (ii) at least one of H_i or V_j contains at most one horizontal or one vertical line from S respectively; (iii) if $X(j+1) < p_b(x)$ or $p_b(x) < X(j)$, then there is no vertical line from S in a strip between p_b and V_j; and (iv) if $Y(i+1) < p_b(y)$ or $p_b(y) < Y(i)$, then there is no horizontal line from S in a strip between p_b and H_i. Note that even if C_{ij} is interesting for p_b, it may contain a red point p that is *already separated* from p_b by a line going through C_{ij}: this happens exactly when H_i or V_j contains two horizontal or vertical lines from S respectively and p lies either in the interior of C_{ij} or on its boundary but not on the same side of H_i or V_j as p_b. The motivation for these definitions is that the cells that are interesting for p_b contain exactly the red points that need to be separated from p_b by lines going through the cells and whose positions cannot be predetermined. We therefore have to add some clauses to express these constraints.

For each $p_b \in \mathcal{B}$ and each cell C_{ij} that is interesting for p_b we construct a clause for every red point $p \in C_{ij}$ that is separable and not already separated from p_b. Initially, the clause is empty. If the specification says that there is exactly one line from S in H_i, we add to the clause a literal as follows: if $y(p_b) \geqslant Y(i+1)$, we add $\neg y^i_p$ (meaning that the horizontal line

is above p, and hence separates p from p_b); if $y(p_b) \leqslant Y(i)$, we add y_p^i. Furthermore, if the specification says that there is exactly one line from S in V_j, we add to the clause a literal as follows: if $x(p_b) \geqslant X(i+1)$, we add the literal $\neg x_p^j$; if $x(p_b) \leqslant X(i)$, we add x_p^j. Observe that this process produces clauses of size at most two. It may produce an empty clause, rendering the 2-SAT unsatisfiable, in the case where there is no line of S in H_i or V_j, but this is desirable since in this case no feasible solution matches the specification. Note that we have constructed $O(|\mathcal{B}||\mathcal{R}|)$ clauses in this way (at most four for each pair of a blue with a red point). Hence, the 2-SAT formula we have constructed has $O(n)$ variables and $O(|\mathcal{B}|n)$ clauses. Since 2-SAT can be solved in linear time, we obtain the promised running time.

To complete the proof we rely on the informal correspondence between assignments to the 2-SAT instance and Axis-Parallel Red-Blue Separation solutions. If there exists a solution that agrees with the guessed specification, this solution can easily be translated to an assignment that satisfies the coherence part of the 2-SAT formula. Furthermore, for any blue point p_b and any separable and not already separated red point p in a cell C_{ij} that is interesting for p_b, the solution must be placing at least one line going through C_{ij} in a way that separates p_b from p (this follows from the fact that the cell is interesting). Hence, the corresponding 2-SAT clauses are also satisfied. Conversely, given an assignment to the 2-SAT instance, we construct an Axis-Parallel Red-Blue Separation solution following the informal meaning of the variables. Note that for every blue point p_b, every red point is C_{ij}-separable for p_b for at least one cell C_{ij}. For any cell C_{ij} that is not interesting for p_b and contains at least one separable point, we have that either all red points in the cell are separated from p_b by lines outside the cell or all separable red points in the cell are separated from p_b by the four lines running through the cell. If C_{ij} is interesting for p_b, then all separable (and not already separated) red points in the cell are separated from p_b because of the additional 2-SAT clauses we added in the second part of the construction. ◀

4 Open problems

The most intriguing open problem is settling the complexity of Axis-Parallel Red-Blue Separation w.r.t. the number of lines. We conjecture it to be FPT. Other problems include the complexity of Red-Blue Separation when the lines can have three different slopes and of Axis-Parallel Red-Blue Separation in 3-dimensions.

Acknowledgements. We thank Sergio Cabello and Christian Knauer for fruitful discussions.

References

1 Édouard Bonnet and Tillmann Miltzow. Parameterized hardness of art gallery problems. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPIcs*, pages 19:1–19:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.ESA.2016.19.

2 G. Calinescu, A. Dumitrescu, H.J. Karloff, and P. Wan. Separating points by axis-parallel lines. *Int. J. Comput. Geometry Appl.*, 15(6):575–590, 2005.

3 O. Devillers, F. Hurtado, M. Mora, and C. Seara. Separating several point sets in the plane. In *Proc. of the 13th Canad. Conf. Comput. Geom.*, pages 81–84, 2001.

4 U.M. Fayyad and K.B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc. of 13th Int. Joint Conf. on Artificial Intelligence*, pages 1022–1029, 1993.

5 R. Freimer, J.S.B. Mitchell, and C.D. Piatko. On the complexity of shattering using arrangements. TR 91-1197, Dept. Comput. Sci., Cornell Univ., NY, 1991.

6 S. Har-Peled and M. Jones. On separating points by lines. arXiv:1706.02004v1, 2017.

7 Ferran Hurtado, Mercè Mora, Pedro A. Ramos, and Carlos Seara. Separability by two lines and by nearly straight polygonal chains. *Discrete Applied Mathematics*, 144(1-2):110–122, 2004. doi:10.1016/j.dam.2003.11.014.

8 J. Kujala and T. Elomaa. Improved algorithms for univariate discretization of continuous features. In *Proc. of the 11th PKDD*, volume 4702 of *LNCS*, pages 188–199, 2007.

9 B. Lu, H. Du, X. Jia, Y. Xu, and B. Zhu. On a minimum linear classification problem. *J. of Global Optimization*, 35(1):103–109, 2006.

10 N. Megiddo. On the complexity of polyhedral separability. *Discr. & Comput. Geom*, 3:325–337, 1988.