On strip separability of bichromatic point sets

Nicolau Oliver*[1] and Carlos Seara†[1]

[1]Universidad Politécnica de Catalunya

Abstract

In this paper we extend and improve algorithms for the separability of red and blue points in the plane using four parallel lines. We also prove sufficient conditions to meet this separability criteria.

1 Introduction

Separability problems of a bichromatic point set in $\mathbb{R}^2$ by a set of parallel lines were studied in Arkin et al. [1, 2], Hurtado et al. [4], and Seara [9]. The number of these lines is denoted by k. It is well known that two object sets are line separable ($k = 1$) if and only if their convex hulls do not intersect. The decision problem of linear separability for two disjoint sets of points, segments, polygons, or circles can be solved in linear time, see Megiddo [6], and O'Rourke et al. [8].

Let B be a set of blue points and R a disjoint set of red points $B \cap R = \emptyset$, both in the plane, $|B| = |R| = n$, and let $CH(B)$ and $CH(R)$ be their convex hulls. The minimum number k of parallel lines separating R and B into monochromatic strips can be computed in $O(n^2 \log n)$ time and $O(n^2)$ space. If $k \leq 4$, there exists an algorithm that solves the problem in $O(n \log n)$ time and $O(n)$ space under a constraint on the point sets. We ask for determining the minimum k for separating R and B, or whether there exists some direction such that R and B can be separated with k lines.

Open problem 1 *Can it be decided if B and R are separable by four parallel lines in $O(n \log n)$ time and $O(n)$ space if $CH(B)$ does not contain any red point?*

The algorithm in [9] has a constraint over the points. We try to solve the problem without the constraint. If a general algorithm for $k = 4$ is found, maybe we can generalize it for any k. Another open problem in [9]:

Open problem 2 *Can it be decided in $O(kn \log n)$ time if B and R are separable with at most $k \geq 5$ parallel lines?*

Let "/" be denote one of the separating parallel lines. We refer to $red/blue/red/blue\ldots$ as the notation for the subsets $R_1/B_1/R_2/B_2/\ldots$, and the separators as $s_1, s_2, s_3, s_4, \ldots$. Assuming k is minimal for any direction, only two possible orderings of the subsets are possible: $red/blue/red/blue/\ldots$, or $blue/red/blue/red/\ldots$. From now on consider only $red/blue/red/blue/\ldots$ separability. A caliper rotates clockwise from 0 to π around a convex polygon, and the separators are denoted as $s_i - s_j$. Calipers are always parallel and rotate in sync.

Observation 3 *All points outside the polygon enter and leave the rotating caliper once, and all points inside the polygon are always inside the caliper.*

We say that a point p inside a caliper is "alive" and is otherwise "dead". The supporting lines of p with respect to the polygon give an interval of directions in which p is alive; one such line corresponds to the entrance slope of p (its "birth") and the other to the departure slope of p (its "death"). The clockwise slope of the supporting lines with respect to the x-axis are computed to be inside the $[0, \pi]$ interval.

2 Separability using four lines

For the $k = 2, 3$ strip separability, Arkin et al. [2] shows $O(n \log n)$ time optimal algorithms. From now on we only consider relevant directions that need at least $k = 4$ lines for separability. The algorithm starts by constructing a caliper that rotates around $CH(B)$. This caliper will constitute the separators s_1 and s_4, see Figure 1. Then, it computes the support lines of the red points with respect to $CH(B)$ and builds the sorted list of birth/death events in $O(n \log n)$ time. The slopes of the support lines belong to $[0, \pi]$. With this event list the caliper can be rotated over $CH(B)$, yielding the sets R_1,R_2,R_3 for all directions. See Figure 1. Next, the algorithm separates the blue points into B_1 and B_2. Take a red point g inside $CH(B)$ as a guard.

Observation 4 *(i) The caliper around $CH(B)$ is never empty of red points. (ii) Red points inside $CH(B)$, including g, belong to R_2. (iii) A line through g separates B_1 from B_2. See Figure 1.*

*Email: nicolau.oliver@estudiantat.upc.edu.

†Email: carlos.seara@upc.edu. Supported by PID2019-104129GB-I00/ MCIN/ AEI/ 10.13039/501100011033.

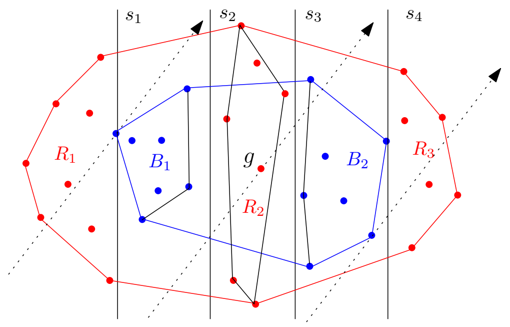

Figure 1: Separability by 4 lines. A red point g inside $CH(B)$ is a *guard* that separates B_1 and B_2 for any direction. A jump event classifying B_1 and B_2.

Jump events describe how the blue points belong to B_1 or B_2 as the caliper rotates. The list of jump events follows from Observation 4, from sorting the angles of the lines through g and each blue point in $O(n \log n)$ time. This list with $O(n)$ angles is the bi-partition list of the guard g. By merging the jump events with the birth and death events we can keep track of all the subsets $R_1/B_1/R_2/B_2/R_3$ as we rotate the caliper. See Figure 1.

In the interval between consecutive events $[e_i, e_{i+1}]$, the subsets $R_1/B_1/R_2/B_2/R_3$ don't change. Neither do their dynamic convex hulls, abbreviated DCH. This interval is separable if $DCH(B_1)$ with $DCH(R_2)$ and $DCH(R_2)$ with $DCH(B_2)$ are separable.

Thus, we compute the supporting lines between the adjacent pairs of dynamic convex hulls (B_1/R_2, R_2/B_2). Intersecting these intervals Θ_1, Θ_2, Θ_3, and Θ_4 with $[e_i, e_{i+1}]$. Repeat for each consecutive pair of events, and merge the results by calculating the union of intervals.

Computing these dynamic convex hulls takes $O(n \log n)$ time, and updating them takes $O(\log n)$ time according to Brodal and Jacob [3]. So the algorithm has $O(n \log n)$ time complexity.

2.1 New algorithm for four line separability

Assume that there are no red points (guards) inside $CH(B)$. Let m guards $G =< g_1, \ldots, g_m >$ be a sequence of guards sorted by birth angle. As above for g, for all $g_i \in G$ compute the sorted bi-partition list in total $O(mn \log n)$ time. This guarantees that we can use the bi-partition lists for the entire rotation.

To separate the blue points into B_1 and B_2, we need at least one guard g_i at any direction inside the caliper, and we use the current guard bi-partition list to do it. When a guard dies, another takes its place.

Observation 5 *There always exists the set $G \subseteq R$, such that for all relevant directions there is at least one guard $g_i \in G$ inside the caliper around $CH(B)$.*

2.2 Minimizing the guard set

The guards have a birth and death event associated, forming the living angle interval. Before an alive guard dies, the next must already be alive. The living angle interval is referred to as the angle that is "guarded/covered" by that guard, see Figure 2; and those intervals must overlap totally covering $[0, \pi]$. Thus, the problem of minimizing guards is equivalent to that of minimizing the sets to cover $[0, \pi]$.

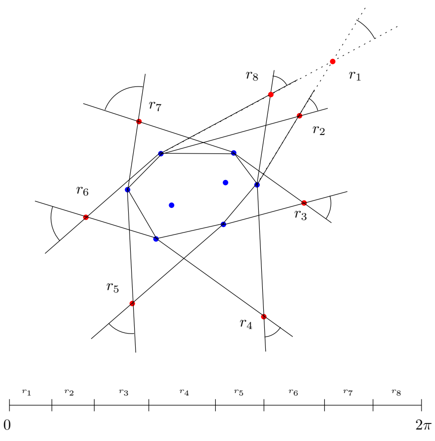

Figure 2: A set of 8 red points covering $[0, 2\pi]$.

3 Sufficient conditions for four line separability

As m could be linear in n, this begs the question of finding conditions over R and B that guarantee that m is constant. There exists a family of configurations with a constant number of guards.

Condition 1 *If there exist guards $a, b, c \in R$ such that all the sides of the triangle $\widehat{abc}$ cross $CH(B)$, then $G = \{a, b, c\}$ is a guard set that covers the entire rotation of the caliper.*

Thus, all configurations that satisfy the Condition 1 have an optimal set of guards of constant size. The guard set $G = \{a, b, c\}$ had a close relationship with the triangle it formed. So, we extend this geometrical analysis to other guard sets: Trace the polygonal line given by the sequence, closing it by adding an edge from the last to the first guard.

Condition 2 *$< g_1, \ldots, g_m >$ is a guard sequence and all the edges of the closed polygonal line traced by the sequence of guards cross $CH(B)$, and m is odd.*

Lemma 6 *If $G =< g_1, \ldots, g_m >$ satisfies Condition 2, then G covers the entire rotation of the caliper.*

This family of closed polygonal lines can be considered for a constant value of m, as $m = 3$ yields the triangle.

The polygons that result from Condition 2 can be from convex to self intersecting. It includes the family of star polygons, with a special interest in those of the form $\{m/\lfloor \frac{m}{2} \rfloor\}$. For an example, see the pentagram $\{5/2\}$ for $m = 5$ and the heptagram $\{7/3\}$ in Figure 3.

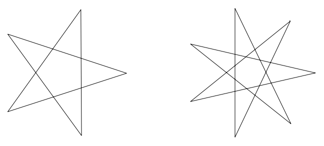
Figure 3: A pentagram $\{5/2\}$ and a heptagram $\{7/3\}$.

Corollary 7 *If $G =< g_1, \ldots, g_m >$ traces an odd star polygon, whose segments all intersect $CH(B)$, then G covers the entire rotation of the caliper.*

The star polygons show that the new algorithm does indeed allow for solving much more general configurations. Nevertheless, Lemma 6 is a sufficient but not necessary condition for an input R and B to have a constant size set of guards. See Figure 4.

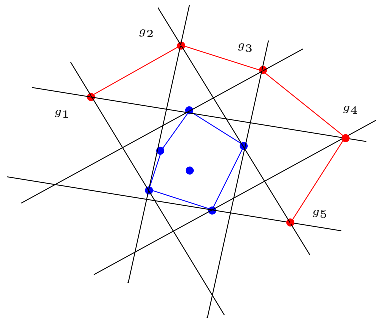

Figure 4: Guards covering the entire rotation, but not tracing a star polygon. Equivalent guards $g_1 \equiv g_5$.

If two guards have the same living angle, they would receive identical use by the algorithm. Thus, two guards are said to be equivalent if the interval they cover is identical, i.e. the support lines of both guards are pairwise parallel. For each red point exterior to $CH(B)$ there is precisely only one other point that satisfies this equivalence: find the support lines of the original point, trace the two parallel support lines tangent to $CH(B)$ on the antipodal points, its intersection point is the equivalent guard. See Figure 4. This equivalence relation reflects that it is possible to swap a guard with its equivalent one, without it affecting the execution of the algorithm.

Observation 8 *If $G =< g_1, \ldots, g_m >$ covers the entire rotation with m odd, there exists equivalent guards $G' =< g'_1, \ldots, g'_m >$ that trace a star polygon.*

4 Finding the optimal guards

As pointed out before, the new algorithm complexity depends on the set G of guards. So it is central to find a constant size G. The guards can be understood as the interval of the caliper rotation they cover, it is the only relevant attribute for the algorithm. This suggests representing the guards as intervals in the unit circle, and each guard being their covering interval. See Figure 5.

Such representation leads to a minimization interval cover problems in $\mathbb{R}$: to find, amongst a set of intervals, the minimum amount such that the union covers an interval $[a, b]$. This is a well-known problem [7] that can be solved via a greedy algorithm in $O(n \log n)$ time with respect to the number of intervals in the set. This indeed can be adapted to find a minimal G. The intersection graph of the intervals has as

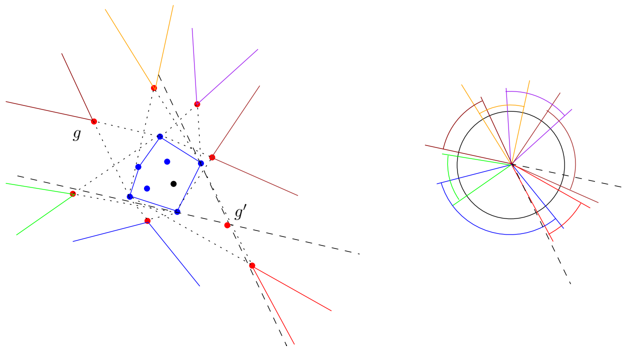

Figure 5: Red guards and their angular intervals.

vertices the intervals, and two vertices are connected if the intersection of the intervals is not empty [5]. See Figure 6. This graph represents pairs of guards that are both alive in some direction. The rotation of the caliper is clockwise, so the edges are arcs and satisfy that the origin guard dies before the destination guard. The direction of the arc captures how the sequence $< g_1, g_2, \ldots, g_m >$ jumps from one guard to the next.

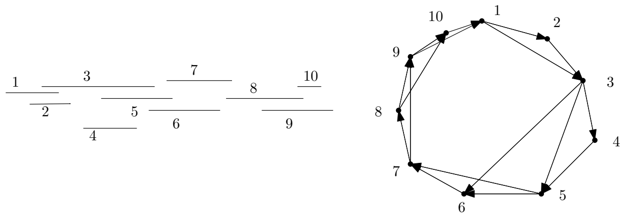

Figure 6: Intersection graph of angular intervals.

Observation 9 *g_i dominates g_j if all directions covered by g_j are also covered by g_i.*

Only the red points that are not dominated by other red points are considered as candidates. This pruning can be done by sorting the guards in $O(n \log n)$ time.

4.1 An algorithm to find optimal guards

Once dominated red points have been pruned, we find the smallest size set of guards. Let G^{opt} be an optimal guard set, and $< g_1, \ldots, g_m >^{opt}$ its sequence. A local optimality can be found among consecutive guards of the sequence. Given $g_i^{opt} \in < g_1, \ldots, g_m >^{opt}$ there might be several guards that cover the angle of death of g_i^{opt}. Let $candidates(g_i)$ be the set of guards that could "succeed" g_i. It follows that $g_{i+1}^{opt} \in candidates(g_i)$. These candidates can be understood more easily as the adjacent vertices of g_i in the intersection graph.

From the set $candidates(g_i)$, a greedy heuristic chooses the guard that dies the last. The candidate selected by this heuristic will be alive at least for all directions covered by any guard in $candidates(g_i^{opt})$, clockwise from g_i^{opt} death. Let g_{i+1}^{greedy} be this greedy choice of candidate.

Observation 10 *g_{i+1}^{greedy} is as optimal as g_{i+1}^{opt}.*

This is the same greedy heuristic used to solve the classical interval cover problem. Given this heuristic, instead of representing all the outgoing edges for each g_i in the graph, just draw the ones chosen by the greedy heuristic. Omitting degenerate cases, all vertices now have at most one outgoing degree. If we assume that R and B are separable using $k = 4$ lines, the whole rotation must be covered, so $\forall g_i \ |candidates(g_i)| \geq 1$. This results in $deg_{out}(g_i) = 1$. This graph will be referred to as the *greedy intersection graph*, $GIG = (V, E)$, where the vertices $V = R_{pruned}$ exclude dominated guards, and it is computed in $O(n \log n)$ time from the intervals.

Observation 11 *If R and B are separable by $k = 4$ lines, the vertices of GIG have $deg_{out}(g_i) = 1$ and the number of edges is $\sum_i^n deg_{out}(g_i) = n$.*

First, GIG can't be acyclic, meaning that it is not possible to cover the entire rotation. So, GIG has at least a directed cycle of length 2, and the cycle corresponds to a $< g_1, \ldots, g_m >^{opt}$. In fact, each connected component of the GIG must have one and only one cycle. Any path starting from any vertex eventually ends up in one of those cycles.

Observation 12 *GIG cycles have the same length.*

A way of finding $< g_1, \ldots, g_m >^{opt}$ can be described in terms of executing search algorithms over GIG, see Tarjan [10]: for any vertex in GIG, follow the arcs until detecting a cycle, which is an $O(n \log n)$ time algorithm. Finally, we have an algorithm that finds the smallest set G of guards, even in the worst case that $m = n$.

Thus, in $O(n \log n)$ time we can detect if the input can be solved by the new algorithm in optimal $O(n \log n)$ time.

5 The second open question

The main insight is that the new $k = 4$ algorithm works because it imposes a similar structure to the $k = 2$ algorithm. The new $k = 4$ algorithm rotates an extra caliper around the $CH(R_{\text{rec}})$, where R_{rec} (Red recursive) are the points inside $CH(B)$. If a few blue points lay inside $CH(R_{\text{rec}})$, call them B_{rec}, then the recursive algorithm rotates a third caliper around $CH(B_{\text{rec}})$. While the substructure repeats, the recursive algorithm can nest further calipers. These are used in a very similar manner to how they are used in the new $k = 4$ algorithm.

For each of these recursively defined convex hulls, compute the same events as for the new $k = 4$ algorithm. Each nested hull bi-partitions the hull that contains it. Each caliper determines a birth and death for each point. The cost of all these operations amounts to repeating the computations k times, once per each nested caliper. The events generated are thus $O(kn)$, and the total cost is $O(kn \log n)$.

References

[1] E. M. Arkin, F. Hurtado, J. S. Mitchell, C. Seara, and S. S. Skiena. Some separability problems in the plane. *16th EuroCG, Eilat, Israel, March 13-15*, 2000.

[2] E. M. Arkin, F. Hurtado, J. S. Mitchell, C. Seara, and S. S. Skiena. Some lower bounds on geometric separability problems. *IJCGA*, 16(01):1–26, 2006.

[3] G. Brodal and R. Jacob. Dynamic planar convex hull. In *43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 617–626, 2002.

[4] F. Hurtado, M. Noy, P. A. Ramos, and C. Seara. Separating objects in the plane by wedges and strips. *Discrete Applied Mathematics*, 109(1):109–138, 2001.

[5] J. Kratochvil and J. Matousek. Intersection graphs of segments. *Journal of Combinatorial Theory, Series B*, 62(2):289–315, 1994.

[6] N. Megiddo. Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems. *SIAM J. on Comput.*, 12(4):759–776, 1983.

[7] V. Mäkinen, V. Staneva, A. Tomescu, D. Valenzuela, and S. Wilzbach. Interval scheduling maximizing minimum coverage. *Discrete Applied Mathematics*, 225:130–135, 2017.

[8] J. O'Rourke, S. Rao Kosaraju, and N. Megiddo. Computing circular separability. *Discrete & Computational Geometry*, 1:105–113, 1986.

[9] C. Seara. On geometric separability. *PhD thesis, Universitat Politènica de Catalunya*, 2002.

[10] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. on Comput.*, 1(2):146–160, 1972.