SEPARABILITY OF POINT SETS BY k-LEVEL LINEAR CLASSIFICATION TREES*

ESTHER M. ARKIN

Department of Applied Mathematics and Statistics, Stony Brook University
Stony Brook, New York 11794, USA
esther.arkin@stonybrook.edu

DELIA GARIJO[†] and ALBERTO MÁRQUEZ[‡]

Departamento de Matemática Aplicada I, Universidad de Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
[†]*dgarijo@us.es*
[‡]*almar@us.es*

JOSEPH S. B. MITCHELL

Department of Applied Mathematics and Statistics, Stony Brook University
Stony Brook, New York 11794, USA
joseph.mitchell@stonybrook.edu

CARLOS SEARA

Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya
Jordi Girona 1, 08034 Barcelona, Spain
carlos.seara@upc.edu

Received 6 April 2011
Revised 9 January 2012
Communicated by Godfried Toussaint

ABSTRACT

Let R and B be sets of red and blue points in the plane in general position. We study the problem of computing a k-level binary space partition (BSP) tree to classify/separate R and B, such that the tree defines a linear decision at each internal node and each leaf of the tree corresponds to a (convex) cell of the partition that contains only red or only blue points. Specifically, we show that a 2-level tree can be computed, if one exists, in

*A preliminary version of this work appeared in the *Abstracts of the 26th European Workshop on Computational Geometry*, Dortmund (Germany), 2010, pp. 41–44. E. Arkin and J. Mitchell are partially supported by the National Science Foundation (CCF-0729019, CCF-1018388). D. Garijo and A. Márquez are partially supported by project MTM2008-05866-C03-01. C. Seara is partially supported by projects MTM2009-07242, Gen. Cat. DGR2009GR1040, and the ESF EUROCORES programme EuroGIGA — ComPoSe IP04 — MICINN Project EUI-EURC-2011-4306.

time $O(n^2)$. We show that a minimum-level ($3 \leq k \leq \log n$) tree can be computed in time $n^{O(\log n)}$. In the special case of axis-parallel partitions, we show that 2-level and 3-level trees can be computed in time $O(n)$, while a minimum-level tree can be computed in time $O(n^5)$.

Keywords: Red-blue separation; binary space partitions; classification; decision trees; machine learning.

1. Introduction

Consider a set of n points in the plane in general position. Each point is either "red" or "blue". Let R denote the set of red points and let B denote the set of blue points. We study the separability of R and B by a *k-level binary space partition tree*. Specifically, a binary space partition tree T is a rooted tree; each node of T corresponds to a (convex, polygonal) region of the plane, with each nonleaf node having an associated partition line, which partitions its corresponding region into the two regions corresponding to its children. The root of T is associated with the entire plane; the root node is at *level* (or *depth*) 0. The children of the root node are at level 1; in general, nodes at *level* i are connected to the root by a (unique) path in T of length i (i.e., having i edges). A k-level tree binary space partition tree T has nodes at levels $\{0, 1, \ldots, k\}$. The regions associated with the leaves of T form a partition of the plane into convex polygons.

We say that R and B are *separated* by a k-level binary space partition tree, T, if each region associated with the leaves of T is *monochromatic* (i.e., contains only points of R or only points of B). The separating k-level tree T corresponds to a recursive partitioning of the plane into disjoint convex regions using (up to) $2^k - 1$ separating straight cuts. Such a tree T of height k (i.e., with k levels) can be used as a classification tree for red/blue points; we can classify, in time $O(k)$, a new point as "red" or "blue" based on the color associated with the cell (corresponding to a leaf in the tree) in which it is located. See Figure 1.

Related work. Separability of point sets is fundamental to classification, clustering, and machine learning. The separating k-level tree generalizes simple separability criteria that have been previously studied. The most basic separability criteria for R and B is that of linear separability, which corresponds to a separating 1-level

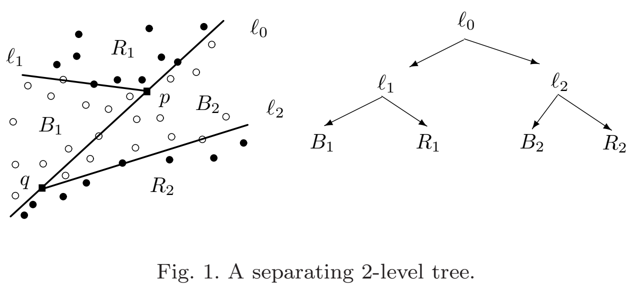


Fig. 1. A separating 2-level tree.

tree: There exists a line separating R and B. Linear separability can be decided in linear time.[13] For sets R and B that are not linearly separable, generalizations include the following separability criteria: A strip (two parallel lines, partitioning the plane into three regions), a wedge (two rays with common origin, partitioning the plane into two regions), a double wedge (two intersecting lines), or three parallel lines. All of these criteria can be decided, and corresponding partitions computed, in optimal $\Theta(n \log n)$ time.[1,2,11,12] (Note that if R and B are strip separable, then they are also wedge separable.) Strip, wedge, double-wedge, or three parallel lines separability criteria are special cases of separability by a 2-level tree.

Separability by multiple parallel lines is a special case of separability by a k-level tree; in particular, $m = 2^k - 1$ parallel lines can be a associated with a (height-balanced) k-level tree. The minimum number of parallel lines needed to separate R and B can be computed in $O(n^2 \log n)$ time.[2] If R and B are the vertices of a regular n-gon, $\lfloor n/2 \rfloor$ is a tight upper bound for the number of parallel lines, and, given the minimum number of separating lines, their common orientation can be computed in $O(n \log n)$ time.[3]

Other separability criteria have also been studied. Given any disjoint point sets, R and B, there always exists a separating polygonal chain, which can be computed in $O(n \log n)$ time. Computing a minimum-link separating polygonal chain that turns alternatively left and right by a constant angle $\alpha \geq \pi/2$ can be done in $O(n \log n)$ time.[11] Separability by m parallel lines is a special case of separability by a monotone m-link polygonal chain. The problem of determining a minimum-link separating polygonal chain of R and B is NP-complete.[9] Edelsbrunner and Preparata[8] solved, in time $O(n \log n)$, the special case of computing a minimum-edge convex polygon separating R and B (if a convex separator exists); their time bound was shown to be optimal in Arkin *et al.*[1]

Our motivation is to consider natural generalizations of previously studied separation and classification problems and, in particular, to consider classifiers that are very fast at query time. The speed of classification of a point with respect to a k-level classification tree is proportional to k; thus, we are motivated to determine classification trees having the minimum number of levels.

Outline of the paper. We initiate the study of separability by k-level trees by considering first the special case of $k = 2$, separability by a 2-level tree. Section 2 is devoted to a special case of 2-level separability, that of separability by a *zigzag*, which corresponds to 2-level tree partitioning such that monochromatic cells of the same color are adjacent (Figure 2). In Section 3 we study the general version of 2-level tree separability, including the generalizations to three or four distinct colors of point sets (instead of just two, red and blue). In Section 4 we consider k-level tree separability and possible configurations of points with $O(\log n)$-level trees. Section 5 is devoted to separability by k-level trees whose partitioning cuts are axis-parallel. (Such trees and partitions are closely related to kd-tree data structures, which are useful for various types of range queries; see de Berg *et al.*,[4] chapter 5.)

2. Zigzag Separability

In this section we consider the zigzag separability problem: Determine whether the sets R and B are separable by a *zigzag* $Z = (\ell_1, s, \ell_2)$, that is a simple, nonconvex 3-link polygonal chain formed by two rays ℓ_1, ℓ_2 and a segment s joining the origins of the rays (Figure 2). Let ℓ_s be the line containing the segment s, and let ℓ'_1 (ℓ'_2) be the line containing the ray ℓ_1 (ℓ_2). Let $CH(X)$ denote the convex hull of a point set X. We can assume that the simpler known special cases of separability have already been tested; specifically, we assume that R and B are not separable by a line, strip, wedge, or convex polygonal chain, each of which can be decided in $O(n \log n)$ time. Thus, under this condition, the following lemma is straightforward.

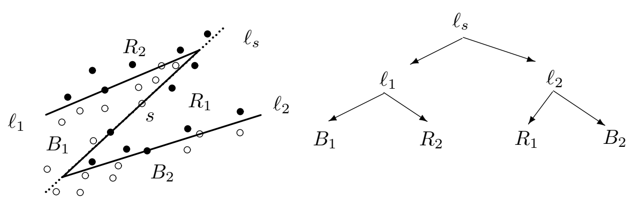


Fig. 2. A separating zigzag.

Lemma 1. *Let R and B be zigzag separable but not separable by a convex polygon. Then, $CH(R)$ contains at least one blue point, and $CH(B)$ contains at least one red point.*

There are three types of zigzags depending on the values of the angles α and β formed by ℓ_s and ℓ_1, and by ℓ_s and ℓ_2, respectively (Figure 3). A separating zigzag $Z = (\ell_1, s, \ell_2)$ defines four wedges that partition R into R_1 and R_2, and B into B_1 and B_2, all four subsets are non-empty, since R and B are not wedge separable.

Since separating zigzags are not necessarily unique, we make the choice specific by considering two optimal separating zigzags: Either a zigzag maximizing $\min\{\alpha, \beta\}$, called *the most convex separating zigzag* (approximating linear separability), or a zigzag that minimizes $\max\{\alpha, \beta\}$ (approximating separability by three parallel lines).

Lemma 2. *Let $Z = (\ell_1, s, \ell_2)$ be the most convex separating zigzag for R and B. Then each of the two rays, and the segment of Z pass through two points of different colors. Moreover, either ℓ'_1 is an inner common tangent line of $CH(R_2)$ and $CH(B)$, or ℓ'_2 is an inner common tangent line of $CH(B_2)$ and $CH(R)$.*

Proof. The key idea is to *stretch* the separating zigzag until each part of the structure touches two points of different colors. Moreover, for each of the types of zigzag in Figure 3, either ℓ'_2 intersects ℓ_1 or ℓ'_1 intersects ℓ_2. In the first case ℓ'_1 is an

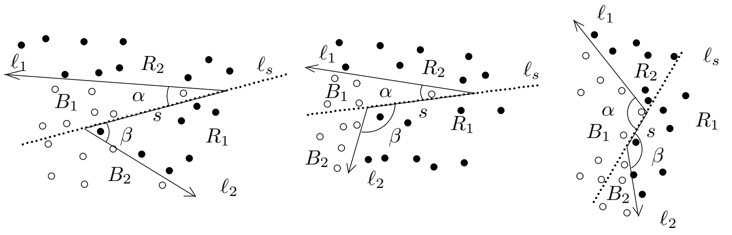


Fig. 3. (a) $0 < \alpha, \beta < \pi/2$, (b) $0 < \alpha < \pi/2$, $\pi/2 \leq \beta < \pi$, and (c) $\pi/2 \leq \alpha, \beta < \pi$.

inner common tangent line of $CH(R_2)$ and $CH(B)$, and in the second case ℓ'_2 is an inner common tangent line of $CH(B_2)$ and $CH(R)$. Notice that both statements hold if ℓ'_1 and ℓ'_2 are parallel. □

Let $I_{X,Y}$ be the number of intersections between pairs of edges of the convex hulls of two point sets X and Y.

Lemma 3. *Let R and B be zigzag separable. Then $I_{R,B} \in \{0, 2, 4, 6\}$.*

Proof. Because the convex hulls are closed Jordan curves, $I_{R,B}$ is even. If $CH(R)$ and $CH(B)$ are nested polygons, $I_{R,B} = 0$ (Figure 4(a)). Assume that $I_{R,B} \geq 2$. By Lemma 2, either $CH(B)$ intersects $CH(R_1)$ but not $CH(R_2)$, or $CH(R)$ intersects $CH(B_1)$ but not $CH(B_2)$. Moreover, $CH(R_1)$ and $CH(B)$ are wedge separable; thus, $I_{R_1,B} \leq 4$. Analogously, $I_{B_1,R} \leq 4$ (Figures 4 and 5). Since $CH(R) = CH(R_1 \cup R_2)$, there are two bridge-edges between $CH(R_1)$ and $CH(R_2)$. An analogous statement holds for $CH(B_1)$ and $CH(B_2)$. Hence, $I_{R,B} \leq 4 + 2 = 6$, corresponding to the at most six alternations of colors in $CH(B \cup R)$ (Figure 5). □

Let R_I (B_I) be the subset of red (blue) interior points of $CH(B)$ ($CH(R)$). By Lemma 1, $|R_I| \geq 1$ and $|B_I| \geq 1$. If $I_{R,B} = 6$, let R'_1, R'_2, and R'_3 (B'_1, B'_2, and B'_3) be the three disjoint subsets of red points (blue points) that are not contained in $CH(B)$ ($CH(R)$) defined according to the 6 intersections of the edges of $CH(B)$ and $CH(R)$. These eight subsets and their respective convex hulls can be computed in $O(n \log n)$ time (Figure 5).

Lemma 4. *Let $Z = (\ell_1, s, \ell_2)$ be the most convex separating zigzag of R and B. Then ℓ_s is a supporting line of some of the following eight convex polygons: $CH(R_I)$, $CH(B_I)$, $CH(R'_1)$, $CH(R'_2)$, $CH(R'_3)$, $CH(B'_1)$, $CH(B'_2)$, and $CH(B'_3)$.*

Proof. We first prove that either R_I is separable from B by the wedge (ℓ_s, ℓ_2), or B_I is separable from R by the wedge (ℓ_1, ℓ_s), and both cases do not always occur (Figures 4(a) and 4(c)). By Lemma 2, if R_2 and B are line separable, then R_1 is

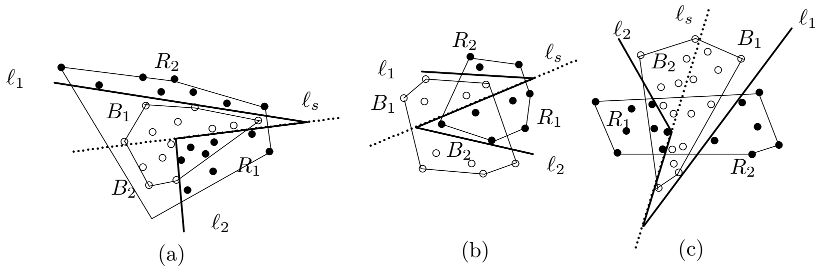


Fig. 4. (a) $I_{R,B} = 0$, (b) $I_{R,B} = 2$, and (c) $I_{R,B} = 4$.

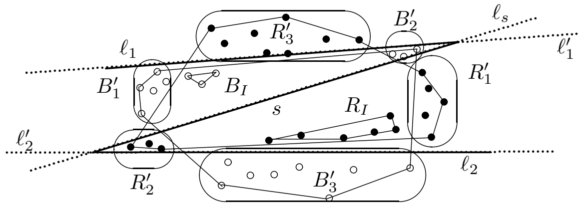


Fig. 5. Subsets of red and blue points for $I_{R,B} = 6$.

separable from B by the wedge (ℓ_s, ℓ_2), and so $R_I \subseteq R_1$ is separable from B by the same wedge. By analogous reasoning, if B_2 and R are line separable, then B_I and R are wedge separable.

If $I_{R,B} \in \{0, 2, 4\}$, then ℓ_s is a supporting line of $CH(R_I)$, because, otherwise, there are not red points inside $CH(B)$ and then B is wedge separable from R, since Z is the most convex separating zigzag. Analogously, if B_2 and R are line separable and $I_{R,B} \in \{0, 2, 4\}$, then ℓ_s is a supporting line of $CH(B_I)$.

Assume that $I_{R,B} = 6$ and recall the second statement of Lemma 2. Let firstly assume that R_2 is line separable from B, and ℓ_s is not a supporting line of $CH(R_I)$. One of the subsets R'_1, R'_2, R'_3 has to be R_2 (say, $R'_3 = R_2$) because, by convexity of $CH(B)$, ℓ'_1 does not separate two of these subsets from $CH(B)$. Thus, R'_1 and R'_2 are contained in R_1 and, since ℓ_s is not a supporting line of $CH(R_I)$, then ℓ_s is a supporting line of either $CH(R'_1)$ or $CH(R'_2)$ (Figure 5). We can proceed analogously, if we assume that B_2 and R are line separable, $I_{R,B} = 6$, and ℓ_s is not a supporting line of $CH(B_I)$. $\square$

Lemma 4 provides the key tool to design the following $O(n \log n)$ time algorithm for computing a separating zigzag $Z = (\ell_1, s, \ell_2)$ for R and B (if it exists). The algorithm looks for ℓ_s and checks the linear separability of $CH(R_2)$ and $CH(B_1)$ by ℓ'_1 and the linear separability of $CH(R_1)$ and $CH(B_2)$ by ℓ'_2. There are a linear number of candidates ℓ_s that are supporting lines of the eight convex polygons above.

Zigzag-algorithm
Input: Point sets R (red) and B (blue)
Output: A separating zigzag $Z = (\ell_1, s, \ell_2)$, or report that none exists

(1) Compute $CH(R)$, $CH(B)$, R_I, B_I, $CH(R_I)$, $CH(B_I)$, and $I_{B,R}$. Check whether $I_{R,B} \in \{0, 2, 4, 6\}$, and compute the intersecting edges of $CH(R)$ and $CH(B)$. Check that $CH(R_I)$ or $CH(B_I)$ is monochromatic. For $R_I = \{r_1\}$ and $B_I = \{b_1\}$, do as follows: If $r_1 \in CH(R)$ and $b_1 \in CH(B)$, then R and B are zigzag separable as shows Figure 6(a) and it is easy to see how to compute the separating zigzag. Analogously if $r_1 \in CH(R)$ and b_1 is interior to $CH(B)$ or vice versa (Figure 6(b)). From now on, assume that $|R_I| \geq 2$ or $|B_I| \geq 2$.

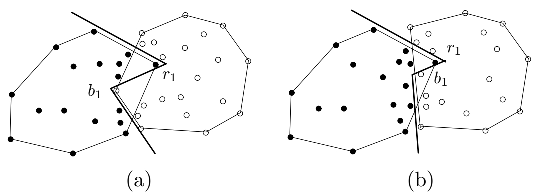


Fig. 6. Zigzag separability with $|R_I| = 1$ and $|B_I| = 1$.

(2) Let P be any of the polygons: $CH(R_I)$, $CH(B_I)$, $CH(R'_1)$, $CH(R'_2)$, $CH(R'_3)$, $CH(B'_1)$, $CH(B'_2)$, or $CH(B'_3)$, with their interior points. Do the following:
   (a) Sort the points in $(R \cup B) - P$ by a counterclockwise rotational sweep over P with an oriented supporting line ℓ_s according to Lemma 4.
   (b) Do a second rotational sweep over P. Each time ℓ_s encounters a red or blue point of $(R \cup B) - P$, maintain and update the convex hulls $CH(R_2)$, $CH(B_1)$ ($CH(R_1)$, $CH(B_2)$) of the red and blue points on the left (right) side of ℓ_s in $O(\log n)$ time.[14] In $O(\log n)$ time, check the linear separability between $CH(R_2)$ and $CH(B_1)$, and between $CH(R_1)$ and $CH(B_2)$, and compute their respective inner common tangent lines (Figure 7). In the affirmative case, a separating zigzag is found.

Analysis of the algorithm. Each step can be done in $O(n \log n)$ time. In step 2, a rotational sweep is done over eight different convex polygons, spending $O(n \log n)$ time on each.

To prove the $\Omega(n \log n)$ time lower bound for deciding the zigzag separability, we reduce the strip separability problem[1] to the zigzag separability problem. The reader is referred to Arkin *et al.*[1] for the construction of the reduction. We place red and blue points on two concentric circles with appropriate radii. A modification from the construction in Arkin *et al.*[1] is needed: We place blue points around the smaller, unit-radius circle and red points around a larger circle of radius $d > 1$, and two additional red points, r_1 and r_2, as in Figure 8. (Specifically, radius d is

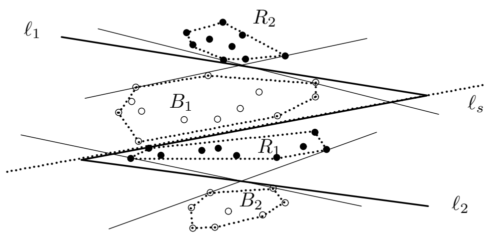


Fig. 7. Supporting lines between monochromatic convex hulls.

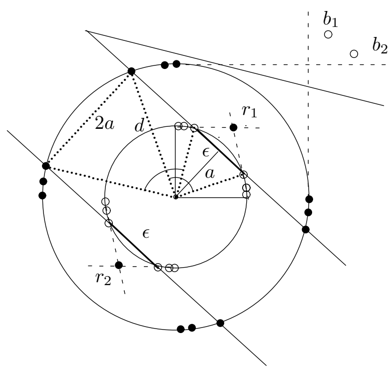


Fig. 8. Construction for the lower bound for zigzag separability.

selected so that a gap of size ϵ between two consecutive blue points on the unit circle determines a line, ℓ, through these two points, and a line, ℓ', through the symmetric pair of blue points, such that ℓ and ℓ' pass through the corresponding red points on the circle of radius d (Figure 8). Letting a denote the distance from the origin to ℓ or ℓ', an appropriate choice of d is $d = 2a/\epsilon = \frac{\sqrt{4-\epsilon^2}}{\epsilon}$.) Additionally, we place two blue points, b_1 and b_2, far enough away from the larger circle, at positions indicated in Figure 8. Now it is clear that there exists a separating zigzag of the sets of red and blue points if and only if the same sets of red and blue points without b_1 and b_2 are strip separable. The last statement is reduced to determining whether there exist two consecutive blue points in the first quadrant of the smallest circle, such that their Euclidean distance is greater than a given $\epsilon > 0$, specified in the input of the problem.

Theorem 1. *Computing a separating zigzag for R and B requires $\Theta(n \log n)$ time.*

Remark. An $O(n^3 \log n)$ time algorithm for determining the separability of R and B by a *monotone* (with respect to some direction) ($k \leq 7$)-polygonal chain is as follows: A mid-segment of the polygonal chain is defined by a line ℓ going through two points. Then, we apply an $O(n \log n)$ time algorithm for the line, wedge or zigzag separability of the point subsets on both sides of ℓ.

3. Separability by a 2-Level Tree

We turn now to the problem of computing a separating 2-level tree $T = (\ell_1, \ell_0, \ell_2)$ for R and B, where ℓ_0, ℓ_1, and ℓ_2 are the oriented line, the ray on the left side of ℓ_0, and the ray on the right side of ℓ_0, respectively (recall Figure 1). Let ℓ'_1 (ℓ'_2) be the line containing ℓ_1 (ℓ_2). Denote by $m(\ell)$ the slope of ℓ. Let p (q) be the intersection point of ℓ_0 and ℓ_1 (ℓ_2). T splits the plane into four convex regions. Recall that R and B are separated by a 2-level tree if there exists a partition of $R \cup B$ into four monochromatic subsets and a 2-level tree, T, whose partition of the plane respects the partition of $R \cup B$.

Criteria. The following criteria provide a systematic classification of possible separating 2-level trees: (1) $m(\ell_0) > 0$, $m(\ell_0) < 0$, or ℓ_0 is horizontal or vertical. (2) Relative position of p and q along ℓ_0: $p \preceq q$ or $q \preceq p$. (3) Slopes of ℓ_1 and ℓ_2 with respect to ℓ_0. (4) Different color assignments to the convex regions.

Classification. We do case analysis according to the following classification criteria: (1) The slope of ℓ_0: We only consider the $m(\ell_0) \geq 0$ case; the case in which $m(\ell_0) < 0$ can be analyzed by rotating the configuration by 90 degrees and applying the corresponding $m(\ell_0) > 0$ case. The first row of Figure 9 illustrates all possible cases for $m(\ell_0) \geq 0$ according to the different relative positions of the rays ℓ_1 and ℓ_2. (2) The relative position of p and q: We only study the case $q \preceq p$. By applying symmetry with respect to a vertical line, followed by a 90-degree rotation, we obtain the case $p \preceq q$; this is seen by comparing the second row with the first row in Figure 9. (3) If two regions that are consecutive (in the order in which the circle at infinity meets the regions) have the same color, the configuration corresponds to one of the following special cases: Linear, zigzag ($p \neq q$), or wedge separability ($p = q$), each of which can be solved in $\Theta(n \log n)$ time.[1,11,12] Thus, we assume that the colors alternate, ℓ_0 has nonnegative slope, and $q \preceq p$.

For an easier analysis of the point configurations for the design of algorithms, we expand the four cases for $q \preceq p$ in the first row of Figure 9 (from left to right) into the seven cases in Figure 10 as follows: The first case is just the case (a) of Figure 10; the second case is expanded into the cases (b) and (c) in Figure 10 according to the slope of ℓ_1; the third case is expanded into the cases (d) and (e)

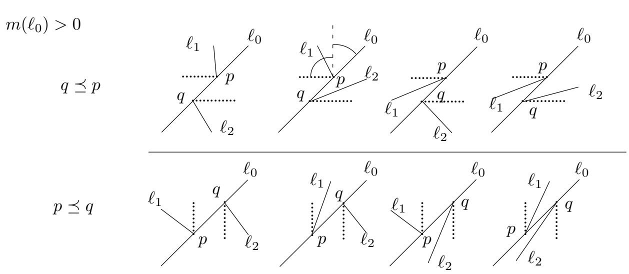


Fig. 9. $m(\ell_0) > 0$.

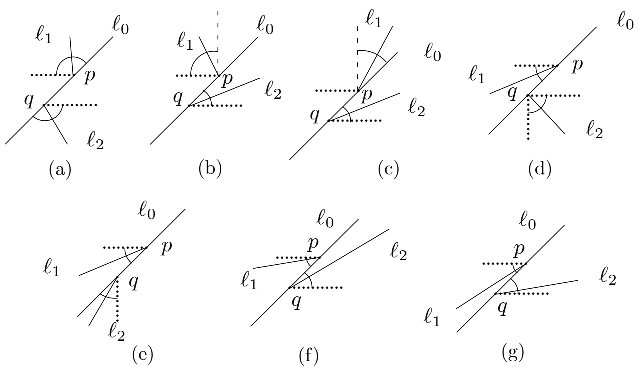


Fig. 10. Configurations for $m(\ell_0) > 0$ and $q \preceq p$.

in Figure 10 according to the slope of ℓ_2; and, finally, the fourth case is expanded into the cases (f) and (g) in Figure 10 according to whether ℓ_1' intersects ℓ_2 (case (f)) or ℓ_2' intersects ℓ_1 (case (g)).

Then, these seven cases in Figure 10 can be reduced by applying symmetries to only four essential cases in Figure 11 as follows: Case (d) is obtained from case (b) by a 180-degree rotation; case (e) is obtained from case (c) by a 180-degree rotation; and case (g), where ℓ_2' intersect ℓ_1, is obtained from case (f), where ℓ_1' intersect ℓ_2, by a 180-degree rotation. Thus, we only consider the four types (1), (2), (3), and (4) of 2-level trees in Figure 11 with a concrete assignment of colors. For types (2), (3), and (4), the line ℓ_1' always intersects ℓ_2.

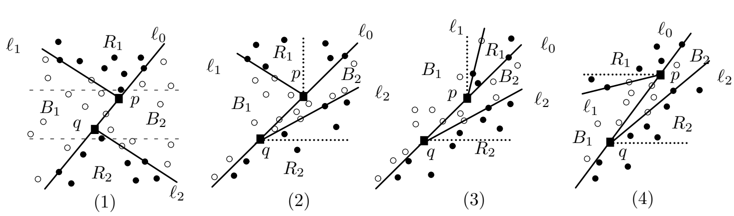


Fig. 11. The 4 types of 2-level trees, up to symmetry.

We design algorithms for the types of 2-level trees $T = (\ell_1, \ell_0, \ell_2)$ illustrated in Figure 11. From now on, we assume that R and B are not separable by a line, wedge, strip, zigzag, or convex polygonal chain. The following lemma is straightforward.

Lemma 5. *If R and B are separable by a 2-level tree, then $I_{R,B} \in \{0, 2, 4, 6\}$.*

The next lemma will allow us to restrict our attention to supporting lines that separate the relevant convex hulls.

Lemma 6. *If R and B are separable by a 2-level tree $T = (\ell_0, \ell_1, \ell_2)$, then it holds that:* (i) *ℓ_0 is a supporting line of $CH(R_1)$ or $CH(R_2)$, and* (ii) *ℓ'_1 (ℓ'_2) is a common supporting line of $CH(R_1)$ and $CH(B_1)$ ($CH(R_2)$ and $CH(B_2)$).*

Proof. Consider any 2-level tree, among the cases shown in Figure 11. Rotate ℓ_0 counterclockwise, about pivot p, until it encounters a point r from $CH(R_1)$ or $CH(R_2)$; say, $r \in CH(R_1)$. Then rotate ℓ_0 counterclockwise about pivot r (or consecutive points in $CH(R_1)$) until it encounters either a point of $CH(R_2)$ or a blue point that would pass into the convex region containing R_1 if we were to continue rotating. This process can modify the partition into B_1 and B_2, but maintains the property of being a separating 2-level tree. Once ℓ_0 is fixed, the lines ℓ'_1 and ℓ'_2 are rotated in a similar manner, with p and q sliding along ℓ_0, until they become supporting lines. □

3.1. *Algorithms*

The overall strategy of the algorithms is as follows: Compute a line that classifies/separates one of the point sets (say, R) into subsets R_1 and R_2, and then use this classification to look for a classification of B into subsets B_1 and B_2 according to a 2-level tree. Below, we present an optimal $O(n \log n)$ time algorithm for 2-level trees of type (1), which is the easy case, since we know that a horizontal line (for which there is a linear number of candidates) will give the classifications of one of the point sets (say, R). Then, in the following subsection, we address trees of types (2), (3), and (4), showing that all such separating 2-level trees can be computed in time $O(n^2)$.

3.1.1. *Type* (1)

If there exists a 2-level tree of type (1), then there exists a horizontal line between two consecutive (in y-coordinate) red points, r_i and r_{i+1} that classifies correctly R into R_1 and R_2, according to the classification by a separating 2-level tree. (Recall, by the general position assumption, there are no two points (red or blue) with the same y-coordinate.) Let h_i be the horizontal line through r_i. Let R_i^u be the set of red points that are on or above h_i; let R_i^d be the red points that are on or below h_i.

Blue points are classified as being "left" or "right" according to whether they lie left or right of ℓ_0 (which is, of course, unknown to us at the beginning of the algorithm). Additionally, blue points are classified as being above h_i, below h_{i+1}, or in the middle between h_i and h_{i+1}. Thus, the blue points are partitioned into the subsets B_i^{lu} and B_i^{ru} above h_i, the subsets B_{i+1}^{ld} and B_{i+1}^{rd} below h_{i+1}, and the subsets B_i^{lm} and B_i^{rm} of the blue points, B_i^m, that lie between h_i and h_{i+1}. (Here, superscripts indicate "up" (u), "down" (d), "left" (l), "right" (r), and "middle" (m).) Let $a_i = |B_i^{lm}| + |B_i^{rm}|$; then, $a_1 + \cdots + a_{n-1} \leq n$. Refer to Figure 12.

The algorithm can be viewed as a sweep line algorithm, sweeping a horizontal line from top to bottom, processing the data at the events when the line passes

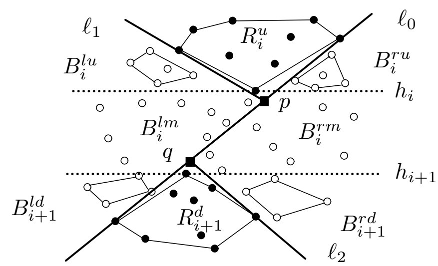


Fig. 12. Subsets in a 2-level tree of type (1).

through a red point; at the event that the line passes through red point r_i, we consider the partition of the plane by the two horizontal lines h_i and h_{i+1}, passing through r_i and r_{i+1}, respectively. Initially (at iteration $i = 1$), all blue points above h_1 are classified left/right according to being left/right of the vertical line through r_1. Then, as i increases (and h_i sweeps downward), blue points that enter the region above h_i are reclassified according to being left/right of the vertical line through r_i. (Once a blue point is swept over and is above h_i, its classification as left/right is never changed.) A similar sweep process, with a horizontal line from bottom to top, allows us to classify blue points below each horizontal line h_i as being left/right of the vertical line through r_i. By the specification of the type (1) case, we know that, if a 2-level tree of type (1) exists with a corresponding horizontal line h_i supporting R_1, then each blue point of B_i^{lu} (resp., B_i^{ru}) must lie left of (resp., right of) every vertical line through a red point at or above the blue point, and, symmetrically, each blue point of B_{i+1}^{ld} (resp., B_{i+1}^{rd}) must lie left of (resp., right of) every vertical line through a red point at or below the blue point.

Type (1)-algorithm

Input: Point sets R (red) and B (blue)

Output: A separating 2-level tree $T = (\ell_1, \ell_0, \ell_2)$ of type (1), or report that none exists

(1) In $O(n \log n)$ time compute the sequences $(r_1, r_2, \ldots, r_n)$ and $(b_1, b_2, \ldots, b_n)$ of the red and blue points, respectively, sorted by decreasing y-coordinate.
(2) For $i = 1$ to $n-1$, consider horizontal partitioning lines h_i, and do the following:
   (a) Maintain the sets R_i^u, R_{i+1}^d, B_i^{lu}, B_i^{ru}, B_{i+1}^{ld}, B_{i+1}^{rd}, B_i^m, and their convex hulls. This takes overall time $O(n \log n)$: With each increment of i, we update two red points and a_i blue points; thus, overall we have maintained seven convex hulls, through a sequence of $O(n)$ updates (insertions and deletions), with each update done in $O(\log n)$ time.[5] (Recall from the discussion above, $O(n \log n)$-time sweeps from top to bottom and from bottom to top suffice to classify blue points as left/right, thereby allowing us to distinguish points of B_i^{lu} from points of B_i^{ru} and points of B_{i+1}^{ld} from points of B_{i+1}^{rd}.)

(b) Determine line separability of the following pairs of convex hulls, in time $O(\log n)$ per check:[14] (i) $CH(R_i^u)$ and $CH(B_i^{lu})$, $CH(R_i^u)$ and $CH(B_i^{ru})$; (ii) $CH(R_{i+1}^d)$ and $CH(B_{i+1}^{ld})$, $CH(R_{i+1}^d)$ and $CH(B_{i+1}^{rd})$; and, (iii) $CH(R_i^u) \cup CH(B_i^{lu}) \cup CH(B_{i+1}^{ld})$ and $CH(R_{i+1}^d) \cup CH(B_i^{ru}) \cup CH(B_{i+1}^{rd})$. (Note that these conditions are necessary but not sufficient for the existence of a separating 2-level tree of type (1).)

(c) Determine a line ℓ_0 (if it exists) separating $CH(R_i^u) \cup CH(B_i^{lu}) \cup CH(B_{i+1}^{ld}) \cup CH(B_i^{lm})$ from $CH(R_{i+1}^d) \cup CH(B_i^{ru}) \cup CH(B_{i+1}^{rd}) \cup CH(B_i^{rm})$. To do this efficiently, we appeal to Lemma 6, which tells us that ℓ_0 is a supporting line of $CH(R_1)$ (i.e., the current $CH(R_i^u)$) or of $CH(R_2)$ (i.e., the current $CH(R_{i+1}^d)$). Consider an oriented supporting line, ℓ, of $CH(R_i^u)$. In $O(a_i \log n)$ time do the following. (1) Sort the points of B_i^m according to a rotating sweep of ℓ around $CH(R_i^u)$ (keeping ℓ in contact with $CH(R_i^u)$). (2) Do a rotating sweep with ℓ in contact with $CH(R_i^u)$, maintaining the convex hull, $CH(B_i^{lm})$, of blue points that are left of ℓ and between h_i and h_{i+1}. If, at some stage of this sweep, the line ℓ is found to separate $CH(B_i^{lu})$ and $CH(B_{i+1}^{ld})$ from $CH(R_{i+1}^d)$, $CH(B_i^{ru})$, and $CH(B_{i+1}^{rd})$, then we have identified a candidate ℓ_0 and a corresponding classification of blue points B_i^m into B_i^{lm} and B_i^{rm}.

(d) Determine lines separators ℓ_1 and ℓ_2 (if they exist), as follows. Since line ℓ_1' must separate $CH(R_i^u)$ from $CH(B_i^{lu}) \cup CH(B_i^{lm}) \cup CH(B_{i+1}^{ld})$, we compute the inner common tangent lines, and the intersections of them with ℓ_0, obtaining an interval $[a, b]$ on ℓ_0, such that $p \in [a, b]$. Similarly, line ℓ_2' must separate $CH(R_{i+1}^d)$ from $CH(B_i^{ru}) \cup CH(B_i^{rm}) \cup CH(B_{i+1}^{rd})$, and we compute an interval $[c, d]$ of intersection points $\ell_2' \cap \ell_0$ on ℓ_0, such that $q \in [c, d]$. If there exists a point p in $[a, b]$ that lies at or above (in y-coordinate) a point q in $[c, d]$, then we have discovered separators ℓ_1 and ℓ_2, and we report the separating 2-level tree T given by the separators ℓ_0, ℓ_1, and ℓ_2, with corresponding sets $R_1 = R_i^u$, $R_2 = R_{i+1}^d$, $B_1 = B_i^{lu} \cup B_{i+1}^{ld} \cup B_i^{lm}$, and $B_2 = B_i^{ru} \cup B_{i+1}^{rd} \cup B_i^{rm}$.

Analysis of the algorithm. The overall running time of the algorithm is $O(n \log n)$, since the convex hulls can be maintained in logarithmic time per point swept over, and separability of pairs of convex hulls can be determined in time $O(\log n)$.

3.1.2. *Types* (2), (3) *and* (4)

Let $(b_1, \ldots, b_n)$ be the sequence of blue points sorted by increasing x-coordinate. Let f_1 and f_2 be vertical lines through b_1 and b_n respectively. Let R_1'' (R_2'') be the set of red points of R_1 (R_2) between f_1 and f_2. Since R and B are not wedge or strip separable, then $B_1 \neq \emptyset$, $B_2 \neq \emptyset$, and either $R_1'' \neq \emptyset$ or $R_2'' \neq \emptyset$ (Figure 13). Since the line ℓ_1' always intersects ℓ_2, we assume that there is at least one blue point of B_2 inside the wedge with origin p and defining rays (rightward from p) along ℓ_0 and ℓ_1', since otherwise R and B are zigzag separable.

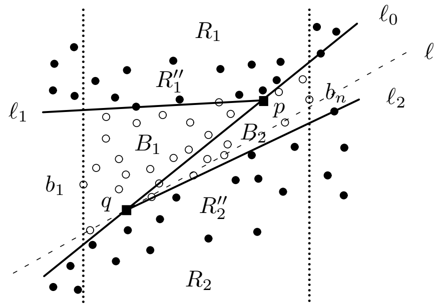


Fig. 13. First and last blue points, b_1 and b_n, and line ℓ separating R_1 and R_2.

The following lemma is immediate:

Lemma 7. *If R and B are separable by a 2-level tree, then one of the following cases holds:* (i) *$b_1 \in B_1$, $b_n \in B_2$;* (ii) *$b_1 \in B_2$, $b_n \in B_1$;* (iii) *$b_1, b_n \in B_2$;* (iv) *$b_1, b_n \in B_1$.*

The next lemma allows us to restrict the set of lines defining a 2-level tree.

Lemma 8. *If R and B are separable by a 2-level tree T, then for any point $b \in B_2$ there is a line ℓ through b that separates R into R_1 and R_2 according to the separation by T.*

Proof. Consider any 2-level tree, $T = (\ell_1, \ell_0, \ell_2)$, among the cases shown in Figure 11. Take a line ℓ through q and rotate ℓ from ℓ_2 to ℓ_0. During the rotation, ℓ passes through all of the points in B_2 and separates R into R_1 and R_2 according to the separation by T. □

We provide an algorithm for computing a 2-level tree T of type (2), (3), or (4) for R and B based on Lemmas 5, 6, 7, and 8.

First we show how to compute a point $b \in B_2$ in order to apply Lemma 8 to get the classification of R into R_1 and R_2 produced by T. By Lemma 7 we get a point $b \in B_2$ if either $b_1 \in B_2$ or $b_n \in B_2$.

Let $b_1, b_n \in B_1$. If T is of type (2) or (4), R and B are zigzag separable, as illustrated in Figure 14(a). If T is of type (3) we compute a point in B_2 as follows: Consider a configuration of red and blue points as in Figure 14(b). Sort the red points by increasing x-coordinate and extend a vertical red ray pointing downward from each red point. There is at least one red point $r \in R_1$ inside $CH(B)$; otherwise, R and B are zigzag separable. Thus, the downwards red ray from r intersects an edge of $CH(B)$, and at least one endpoint of the edge must lie in B_2, since the ray does not intersect $CH(B_1)$. This endpoint is the desired point $b \in B_2$, and is obtained in time $O(n \log n)$.

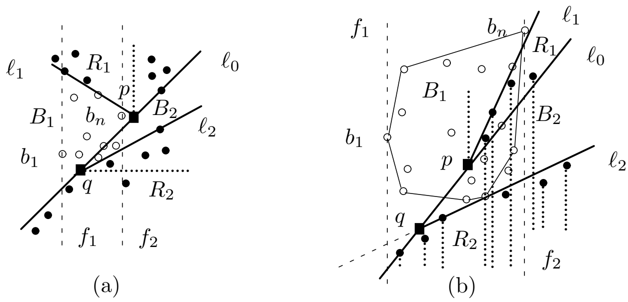


Fig. 14. Illustration for searching for separating 2-level trees of types (2), (3), or (4). Here, $b_1, b_n \in B_1$.

Types-(2)-(3)-(4)-algorithm
Input: Point sets R (red) and B (blue)
Output: All of the separating 2-level trees T of types (2), (3) or (4) for R and B

(1) In $O(n^2)$ time construct the arrangement, $\mathcal{A}$, of the lines dual to the points in $R \cup B$.
(2) In $O(n \log n)$ time do the following: (i) Check that $I_{R,B} \in \{0, 2, 4, 6\}$; (ii) check that R and B are not wedge, strip, or zigzag separable; and, (iii) compute a point $b \in B_2$ according to Lemma 7 and the discussion above. Let ℓ be a directed line through b. Sort the red points according to a rotational sweep with ℓ.
(3) Do a rotational sweep with ℓ, stopping each time ℓ encounters a red point; these events result in $O(n)$ corresponding partitions $\{R_1, R_2\}$ of R. Maintain $CH(R_1)$, $CH(R_2)$, and the directed supporting line, ℓ_R, between them, such that $CH(R_1)$ is in ℓ_R^- (the left half-plane) and $CH(R_2)$ is in ℓ_R^+ (the right half-plane). For each partition $\{R_1, R_2\}$, determine whether there exists a bipartition $\{B_1, B_2\}$ of B corresponding to a separating 2-level tree T for R and B in $O(n)$ time as follows:
    (a) In $O(n)$ time, utilize the arrangement $\mathcal{A}$ to obtain the clockwise order of the blue points with respect to $CH(R_1)$ (and separately with respect to $CH(R_2)$). (The ordered sequence of red points that are vertices of $CH(R_1)$ correspond to a sequence of dual red lines in $\mathcal{A}$; we traverse these lines, in the order given by $\mathcal{A}$, discovering the order in which dual blue lines cross them, thereby obtaining the rotational order of the blue points with respect to $CH(R_1)$.) Also check that $CH(R_1)$ and $CH(R_2)$ contain no blue points. Noting that all of the blue points in ℓ_R^+ have to belong to B_2, let b'_1 be the first blue point in ℓ_R^+ according to the clockwise rotation order. Denote by $(b'_1, b'_2, \ldots, b'_n)$ such an ordering (refer to Figure 15). It holds that b'_1 belongs to B_2, and b'_n belongs B_1. Let ℓ_n be the directed supporting line of $CH(R_1)$ through b'_n, and let ℓ_n^- (ℓ_n^+) be the left (right) half-plane defined by ℓ_n. Obviously, R_1 is contained in ℓ_n^+.

(b) Starting at b'_n and following the order above, in $O(n)$ time compute the location of the blue points in ℓ_n^+ or ℓ_n^-. Let b'_i be the last blue point in $\ell_n^+ \cap \ell_R^-$. If there exists an appropriate bipartition $\{B_1, B_2\}$, then $\{b'_i, b'_{i-1}, \dots, b'_1\} \subseteq B_2$.

(c) Let $B_1 = \{b'_n, \dots, b'_{i+1}\}$ and $B_2 = \{b'_i, b'_{i-1}, \dots, b'_1\}$. By construction, R_1 and B_1 are line separable by ℓ_n. In $O(n)$ time, check if R_2 and B_2 are line separable, and if $R_1 \cup B_1$ is line separable from $R_2 \cup B_2$. Otherwise, there does not exist a separating 2-level tree for the bipartition $\{R_1, R_2\}$. In the affirmative case, construct a separating 2-level tree T for R and B.

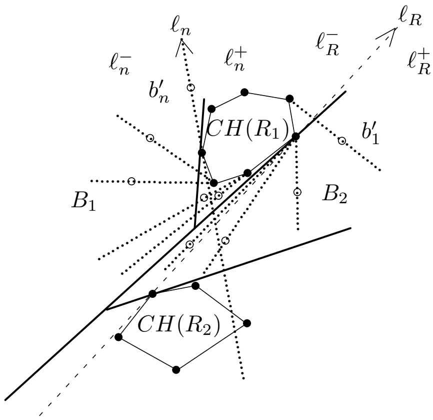


Fig. 15. Illustrating step 3 of the algorithm for a 2-level tree of type (2).

Theorem 2. *Computing all of the separating* 2*-level trees for* R *and* B *can be done in* $O(n^2)$ *time and space.*

Proof. The algorithm above spends $O(n^2)$ time and space constructing the dual arrangement $\mathcal{A}$. For each partition $\{R_1, R_2\}$ of R, the algorithm decides whether there exists a separating 2-level tree and computes it in $O(n)$ time. Lemma 8 ensures the existence of an appropriate bipartition of R at some step, if there exists a separating 2-level tree for R and B. By Lemma 6 we can assume that ℓ is a supporting line of $CH(R_1)$ and proceed analogously for ℓ being a supporting line of $CH(R_2)$. By the same lemma we can assume that ℓ_n is a supporting line of $CH(R_1)$ and $CH(B_1)$. $\square$

Remark. It is easy to see that all of the combinatorially different 2-level trees for a set of n red and blue points in $\mathbb{R}^d$ can be computed in $O(n^{d+1})$ time. For $d = 2$, the above theorem shows that an improved time bound of $O(n^2)$ is possible. It remains an open problem to determine if a separating 2-level tree for R and B can be computed in $o(n^2)$ time.

3.2. *Three or four colored point sets*

The 2-level tree problem for point sets having three or four distinct colors can be solved as follows. The four colors case can be solved in $O(n)$ time by checking the linear separability between pairs of point sets. The three colors case, with point sets R, B and G, can be viewed either as a zigzag of R and $G \cup B$, or as a separation with a 2-level tree of R and $G \cup B$ restricted to have linear separability between G and B (Figure 16). But, in both cases, we have as additional information the linear separability of G and B, which can be checked in advance in $O(n)$ time. We use this information to compute the corresponding 2-level tree. Thus, this problem can be solved in $O(n \log n)$ time. This time bound is optimal, as can be seen from the zigzag separability for R, B and G, with an easy adaptation of the lower bound construction in Theorem 1.

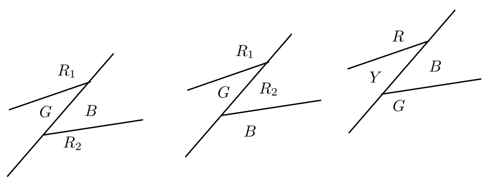


Fig. 16. 2-level trees for three and four colored point sets.

Theorem 3. *A separating* 2*-level tree for three-colored sets of* n *points can be computed in* $O(n \log n)$ *time, which is worst-case optimal. For four-colored sets of* n *points a* 2*-level tree can be computed in* $O(n)$ *time.*

4. *k*-Level Trees

We now consider separating $(k \geq 3)$-level trees for R and B. A separating $O(\log n)$-level tree for R and B can be computed as follows: Appealing to the Ham-Sandwich theorem, we can compute a line that gives an equitable bipartition $B_1 \cup R_1$, $B_2 \cup R_2$ of $B \cup R$, then proceed recursively on each part until we obtain monochromatic subsets. In the end, we obtain a k-level tree for $n \leq 2^k$ points.

Note that a k-level tree produces a subdivision of the plane into monochromatic convex cells, each one bounded by at most k lines. We can use a dynamic programming algorithm to compute a minimum-level tree for R and B in (quasi-polynomial) $n^{O(\log n)}$ time. In particular, a subproblem is specified by a convex polygon P having at most $k = O(\log n)$ sides, each defined by one of the $\binom{n}{2}$ lines determined by point pairs of $R \cup B$. The optimization for a subproblem selects among the $\leq \binom{n}{2}$ possible cuts, ℓ, and recursively solves the minimum-level tree problem on each side of ℓ.

Theorem 4. *A separating* k*-level tree for* R *and* B *exists with* $k \leq \lceil \log n \rceil$*. Furthermore, a minimum-level tree can be computed in* $n^{O(\log n)}$ *time.*

On the other hand, there exist configurations of points for which the depth k^* of a minimum-level tree is $\Omega(\log n)$. In particular, let S be a set of n points in general position. Replace each point $p_i \in S$ by a structure of four (very close) points, two red and two blue, as in Figure 17, obtaining the sets R and B of $2n$ red and $2n$ blue points, respectively. Any k-level tree for R and B has to separate each pair of red and blue points of the p_i structure and must, therefore, have $k = \Omega(\log n)$ levels.

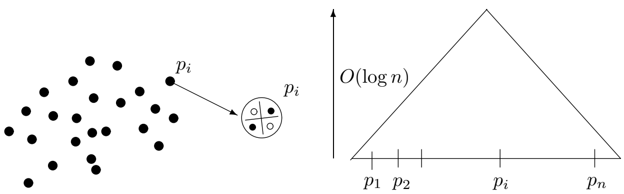


Fig. 17. An example (left) of a configuration of $R \cup B$ requiring an $\Omega(\log n)$-level separating tree.

It is unlikely that a substantially more efficient algorithm exists for computing separating k-level trees in general. In fact, in related work, Grigni *et al.*[10] considered the problem of designing a near-optimal linear decision tree T to classify two given point sets R and B in $\mathbb{R}^n$, so that T defines a linear decision at each internal node, such that for each leaf v of T, either only red or only blue points lead the algorithm to v. The authors considered two measures of such a classifier, the *number of internal nodes* and the *depth* of the tree, and prove a very strong negative result on high-dimensional classification trees: Unless NP=ZPP, no polynomial-time algorithm for optimizing the depth of a classifier can have approximation ratio better than any fixed constant. Further, Das and Goodrich[7] showed that the following problem is NP-complete: Given a set S of n points in $\mathbb{R}^3$, partitioned into two concept classes, red and blue, decide if there exists a decision tree T with at most k nodes that separates the red points from the blue points.

5. Separability with Axis-Parallel Partitions

In this section, we consider k-level trees defined by axis-parallel lines. First we show how to compute a 2-level tree as in Figure 18(a). We consider the case in which ℓ_0 is vertical and ℓ_1, ℓ_2 are horizontal; other cases, which also depend on the color assigned to the rectangles produced by the 2-level tree structure, can be handle analogously. A key observation is that, if there exists a separating 2-level tree, then during a sweep with a vertical line ℓ from left to right the sets of red and blue points on the left of ℓ must be separable by a horizontal line at least until the moment when ℓ reaches ℓ_0. This observation is utilized in the following $O(n)$ time algorithm.

Axis-parallel 2-level tree algorithm. Let $T(n)$ denote the running time for an input of size n. First, compute (in $O(n)$ time[6]) the median M of the x-coordinates of the points. Let ℓ be the vertical line through M. Let R_1 and B_1 (resp., R_2 and B_2) be the

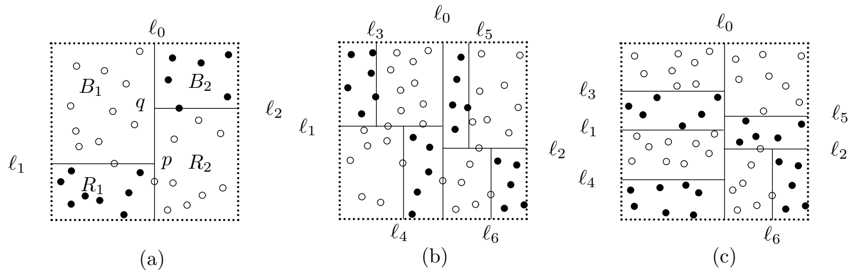


Fig. 18. (a) Axis-parallel 2-level tree, (b) and (c) axis-parallel 3-level trees.

subsets of red and blue points on the left (resp., right) side of ℓ. Check, in time $O(n)$, whether R_1 and B_1 (resp., R_2 and B_2) are line separable with a horizontal line. If the two answers are negative, the algorithm concludes that there is no separating 2-level tree. If the two answers are positive, the algorithm concludes with a separating 2-level tree, with $\ell_0 = \ell$. Otherwise, assume that the positive answer is on the left of ℓ; compute and store the y-interval of horizontal separators for points (R_1, B_1) left of ℓ. Now, proceed recursively, in time $T(n/2)$, for the $n/2$ points on the right of ℓ. Specifically, we compute the median M' of the x-coordinates of the points in $R_2 \cup B_2$, and determine the y-interval of horizontal separators (if any exist) for the red and blue points of $R_2 \cup B_2$ on each side of a vertical line, ℓ', through M', intersecting the y-interval for points in $R_2 \cup B_2$ left of ℓ' with the y-interval for points ($R_1 \cup B_1$) left of ℓ. The recursive search continues, either on the left or on the right of each successive median vertical line, according to the existence of horizontal separators (recursing on the side that has no separator). The search concludes when we discover a vertical separator such that there either exist horizontal separators on both sides (yielding the desired 2-level separating tree), or we discover that there is no possible horizontal separator on both sides (showing that no 2-level separating tree exists). The running time, $T(n)$, satisfies $T(n) = T(n/2) + O(n)$, implying that $T(n) = O(n)$.

Theorem 5. *A separating axis-parallel* 2*-level tree for* R *and* B *can be computed in* $O(n)$ *time.*

A *crossing* 2*-level tree* is a 2-level tree for R and B defined by two axis-parallel perpendicular lines ($p = q$). A separating crossing 2-level tree is also a *separating horizontal/vertical double-wedge* for R and B. In Arkin *et al.*,[1] the authors showed an $\Omega(n \log n)$ time lower bound for the horizontal/vertical double-wedge separability problem; this lower bound applies also to the crossing 2-level tree problem. An $O(n \log n)$ time algorithm for the separability by a crossing 2-level tree can be obtained by first sorting the points of $R \cup B$ by both x- and y-coordinate and then applying an easy modification of the linear-time algorithm above.

Axis-parallel 3-level tree algorithm. A key observation is that if there exists a 3-level tree for R and B with a configuration as in Figure 18(b), then for any vertical line ℓ_0 crossing the axis-parallel bounding box of $R \cup B$, either (i) the subset of points of $R \cup B$ on the left of ℓ_0 is separable by a 2-level tree or (ii) the subset of points of $R \cup B$ on the right of ℓ_0 is separable by a 2-level tree.

Thus, analogous to the search described above for the 2-level tree problem, we can do a binary search for a possible vertical line ℓ_0 such that both subsets of points of $R \cup B$ on the left and on the right of ℓ_0 are separable by a 2-level tree, or the conclusion that no such ℓ_0 exists. The result is an $O(n \log n)$ time algorithm. (Other configurations, as in Figure 18(c), can be handled analogously.) We now show, however, that there is a linear-time algorithm for the axis-parallel 3-level tree problem.

Consider the case in Figure 18(b); other cases are similar. In $O(n)$ we compute the vertical line ℓ'_3 containing the ray ℓ_3 using the median technique above such that the points on the left of ℓ'_3 are separable by a horizontal line (say, ℓ'_1); we also compute a vertical interval I_1 where this horizontal line ℓ'_1 can be located. Then we proceed analogously (using the median technique) with the points on the right of the computed ℓ'_3 until we find a vertical line ℓ'_4 containing a ray ℓ_4 such that the points between ℓ'_3 and ℓ'_4 are monochromatic, spending $O(n)$ time in this second process. Then we proceed analogously with the points on the right of ℓ'_4 until we find a vertical line ℓ_0 such that the points between ℓ'_4 and ℓ_0 are separable by a horizontal line located in the computed vertical interval I_1; again we spend $O(n)$ time in this third process. Thus, the computation of the line ℓ_0 and the 2-level tree on the left of ℓ_0 takes $O(n)$ time. Similarly, in additional $O(n)$ time we check that the points on the right of the computed ℓ_0 are separable by a 2-level tree.

Notice that depending on the vertical/horizontal choices for ℓ_0, ℓ_1, ℓ_2, ℓ_3, ℓ_4, ℓ_5, ℓ_6, and the colors assigned to each region, the number of different types of axis-parallel 3-level trees is 2^{2^3-1}. So the overall algorithm takes $O(n)$ time.

Theorem 6. *A separating axis-parallel* 3*-level tree for R and B can be computed in $O(n)$ time.*

Remark. The linear-time method for determining the existence of an axis-parallel 3-level tree implies that, using the binary search previously discussed, we can solve the 4-level tree problem in time $O(n \log n)$, and, more generally, the k-level tree problem in time $O(n \log^{k-3} n)$, for fixed k, with a huge dependence on k hidden in the big-Oh notation. In fact, the linear-time method we gave for 3-level trees can be extended to 4 or more levels, yielding a linear-time method for any fixed k; however, the dependence on k reflects the fact that there are 2^{2^k-1} k-level trees, leading to a very high $(O(2^{2^k} n))$ time bound in terms of n and k. Below, we obtain polynomial time in both n and k for computing a minimum-level axis-parallel tree, using dynamic programming.

Minimum-level axis-parallel tree algorithm. The general problem consists of computing a minimum-level axis-parallel tree for points R and B, in general position. We can assume that each of the horizontal/vertical lines defining the tree pass through the input points, $R \cup B$; we consider a point p that lies on a horizontal (resp., vertical) line ℓ to lie in the (closed) region to the left (resp., below) ℓ.

Our algorithm employs dynamic programming. Let $x_1 < x_2 < \cdots < x_n$ denote the x-coordinates of the n input points $R \cup B$, indexed in sorted order; similarly, let $y_1 < y_2 < \cdots < y_n$ denote the y-coordinates. A subproblem is specified by a rectangle, $\mathcal{R} = (x_i, x_j] \times (y_k, y_l]$. Thus, there are $O(n^4)$ subproblems. The *value* of a subproblem $\mathcal{R}$, $f(\mathcal{R})$, is the minimum number of levels in an axis-parallel classification tree of the points of $R \cup B$ within $\mathcal{R}$. If $\mathcal{R}$ is monochromatic (i.e., has points only of R or only of B within it), $f(\mathcal{R}) = 0$; this forms the base case for the dynamic programming recursion. In general, for subproblem $\mathcal{R}$ we have

$$f(\mathcal{R}) = \begin{cases} 0 & \text{if } \mathcal{R} \text{ is monochromatic} \\ 1 + \min_\ell \max\{f(\mathcal{R}_{\leq \ell}), f(\mathcal{R}_{>\ell})\} & \text{otherwise,} \end{cases}$$

where the minimization is over all horizontal/vertical cuts ℓ that pass through points of $R \cup B$ and intersect $\mathcal{R}$, and $\mathcal{R}_{\leq \ell}$ (resp., $\mathcal{R}_{>\ell}$) denotes the subrectangle of $\mathcal{R}$ that is on or below/left (resp., strictly above/right) horizontal/vertical line ℓ. The algorithm tabulates the values $f(\mathcal{R})$ in order of increasing values of $j - i$ and $l - k$, in the standard way. Since there are $O(n)$ candidate cuts ℓ to consider for each R, the overall running time is $O(n^5)$, using a table of size $O(n^4)$. We thus conclude with the following theorem.

Theorem 7. *A minimum-level separating axis-parallel tree for R and B can be computed in $O(n^5)$ time, using $O(n^4)$ space.*

Remark. A minimum-level separating tree using only vertical (or only horizontal) lines can easily be computed in $O(n \log n)$ time by considering color transitions in the x-sorted (y-sorted) list of points $R \cup B$.

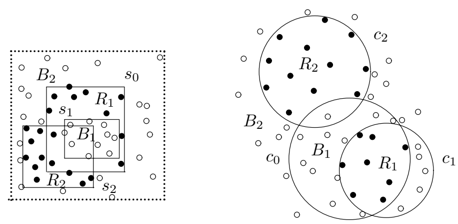
6. Conclusion

We have initiated a study of k-level linear classification trees. Table 1 summarizes the time and space complexities of the algorithms presented. (As we remarked after Theorem 6, we note that the method we presented for axis-parallel 2- and 3-level trees can be extended to yield a linear (in n) time algorithm for any constant number, k, of levels, but the dependence on k is prohibitive ($O(2^{2^k} n)$).)

In future work, we hope to consider other k-level trees defined by cuts other than lines or hyperplanes, e.g., circles or axis-aligned boxes; see Figure 19. Multilevel trees based on separation by circles or axis-aligned boxes have potential applications in bounding volume hierarchies, which are useful for intersection detection and shape approximation.

Table 1. Summary of the time and space complexities.

Classification trees	Time	Space
Zigzag	$\Theta(n \log n)$	$O(n)$
2-level tree	$O(n^2)$	$O(n^2)$
Minimum-level tree ($3 \leq k \leq \log n$)	$n^{O(\log n)}$	$n^{O(\log n)}$
Axis-parallel 2-level tree	$O(n)$	$O(n)$
Axis-parallel 3-level tree	$O(n)$	$O(n)$
Minimum-level axis-parallel tree	$O(n^5)$	$O(n^4)$



(a) (b)

Fig. 19. Example of 2-level trees based on (a) axis-aligned boxes and (b) circles.

Acknowledgments

We thank to two anonymous referees for their many useful suggestions and comments, which helped improve the paper substantially.

References

1. E. M. Arkin, F. Hurtado, J. S. B. Mitchell, C. Seara and S. S. Skiena, Some lower bounds on geometric separability problems, *Int. J. Comput. Geom. Appl.* **16**(1) (2006) 1–26.
2. E. M. Arkin, F. Hurtado, J. S. B. Mitchell, C. Seara and S. S. Skiena, Some separability problems in the plane, *Abstracts of the 16th European Workshop on Computational Geometry*, Eilat, Israel (2000), pp. 51–54.
3. T. Asano, J. Hershberger, J. Pach, E. Sontag, D. Souvaine and S. Suri, Separating bichromatic points by parallel lines, *Proc. 2nd Canadian Conf. Computational Geometry* (1990), pp. 46–49.
4. M. de Berg, O. Cheong, M. van Kreveld and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd edn. (Springer-Verlag, 2008).
5. G. Brodal and R. Jacob, Dynamic planar convex hull, *Proc. 43rd Symp. Foundations of Computer Science*, IEEE Computer Society (2002), pp. 617–626.
6. T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms* (MIT Press, 2001).
7. G. Das and M. T. Goodrich, On the complexity of optimization problems for convex polyhedra and decision trees, *Comput. Geom.: Theor. Appl.* **8** (1997) 123–137.
8. H. Edelsbrunner and F. P. Preparata, Minimum polygonal separation, *Infor. Comput.* **77** (1988) 218–232.
9. S. Fekete, On the complexity of min-link red-blue separation problem (1992).

10. M. Grigni, V. Mirelli and C. H. Papadimitriou, On the difficulty of designing good classifiers, *SIAM J. Comput.* **30**(1) (2000) 318–323.
11. F. Hurtado, M. Mora, P. A. Ramos and C. Seara, Separability by two lines and by nearly-straight polygonal chains, *Discr. Appl. Math.* **144** (2004) 110–122.
12. F. Hurtado, M. Noy, P. A. Ramos and C. Seara, Separating objects in the plane by wedges and strips, *Discr. Appl. Math.* **109** (2000) 109–138.
13. N. Megiddo, Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems, *SIAM J. Comput.* **12**(4) (1983) 759–776.
14. F. P. Preparata and M. I. Shamos, *Computational Geometry, An Introduction* (Springer-Verlag, 1988).