Bichromatic separability with two boxes: a general approach

C. Cortés[*], J.M. Díaz-Báñez[†], P. Pérez-Lantero[‡], C. Seara[§], J. Urrutia[¶], I. Ventura[||]

Abstract

Let S be a point set in general position on the plane such that its elements are colored red or blue. We study the following problem: Remove as few points as possible from S such that the remaining points can be enclosed by two isothetic rectangles, one containing all the red points, the other all the blue points, and such that each rectangle contains only points of one color. We prove that this problem can be solved in $O(n^2 \log n)$ time and $O(n)$ space. We show how our techniques can be generalized to solve other variants of the given problem such as the 3-dimensional problem and the trichromatic problem.

1 Introduction

In Data Mining and Classification problems, a natural method to analyze data, is to select prototypes representing different classes of data. A standard technique to achieve this, is to perform cluster analysis on the training data [6, 9]. The clustering can be obtained by using simple geometric shapes such as circles or boxes. In [1, 7], circles and parallel-axis boxes respectively, are considered for the selection. In [1], the following problem is studied: given a bicolored point set, find a ball that contains the maximum number of red points without containing any blue point inside it.

In some cases, as in Medical Data Analysis [8], methods can produce slanted classifications due to the fact that some data may be defective or contain values out of reasonable ranges. In other cases, we may obtain data hard to classify due to relatively small similarities between different classes. A possible way to find a better classification for the former problem is to remove some data-points from the input. Culling the minimum number of such points can be a suitable criterium to lose as less information as possible. Thus, in this paper we study the following problem:

Let S be a bi-chromatic point set on the plane in general position such that its elements are colored red or blue. Remove as few points as possible from S such that the remaining points can be enclosed by two isothetic rectangles, one containing all the red points, the other all the blue points, and such that each rectangle contains only points of one color.

Notice that the problem can be stated as follows:

Find the cardinal of the largest subset S' of S which can be enclosed by two isothetic rectangles $\mathcal{R}$ and $\mathcal{B}$ such that: $\mathcal{R}$ (resp. $\mathcal{B}$) contains all the red (resp. blue) points of S' and $\mathcal{R}$ (resp. $\mathcal{B}$) contains no blue (resp. red) points of S'.

[*]Departamento Matemática Aplicada I, Universidad de Sevilla, ccortes@us.es

[†]Departamento Matemática Aplicada II, Universidad de Sevilla, dbanez@us.es, partially supported by grants BFM2003-04062 and MTM2006-03909.

[‡]Departamento de Computación, Universidad de La Habana, pablo@matcom.uh.cu,partially supported by grant MTM2006-03909.

[§]Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, carlos.seara@upc.edu,partially supported by grants MEC MTM2006-01267 and DURSI 2005SGR00692.

[¶]Instituto de Matemáticas, Universidad Nacional Autónoma de México, urrutia@matem.unam.mx, partially supported by grant MTM2006-03909.

[||]Departamento de Matemáticas, Universidad de Huelva,iventura@us.es, partially supported by grants BFM2003-04062 and MTM2006-03909.

We will refer to this problem as the *Empty Intersection Enclosing Boxes problem* or the *EIEB-problem* for short. For example, the solution to the *EIEB-problem* for the point set illustrated in Figure 1 is $n-2$ (being n the cardinal of the original set), since by removing the points r_1 and b_1 we can obtain two rectangles, $\mathcal{R}$ and $\mathcal{B}$, each of them containing only red and blue points, respectively. It is easy to check that it is not possible to remove only one point in order to improve this number.

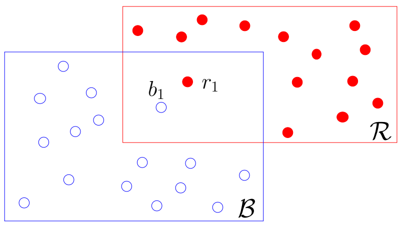


Figure 1: Getting a solution by removing the points r_1 and b_1.

A first $O(n^3)$-time and space algorithm has been proposed in [5]. The main goal of this paper consists on proving that this problem can be solved in $O(n^2 \log n)$ time and $O(n)$ space. We also apply our technique to solve other variants of the problem.

The outline of this paper is as follows. A simple algorithm to compute a solution in $O(n^4)$-time and $O(n^2)$-space is first proposed en Section 2. Properties stated for this approach will be used later. In Section 3 we introduce a data structure, *Maximum Consecutive Subsequence Tree*, *MCS-Tree*, to maintain dynamically the optimal subsequence of the *Maximum Consecutive Subsequence Problem*. In Section 4 we present the main algorithm for computing a solution for the *EIEB-problem* in $O(n^2 \log n)$ time and linear space. Section 5 is devoted to reduce the three-dimensional problem to $O(n^2)$ instances of two-dimensional problems. In Section 6 we show how the MCS-Tree can be modified to solve the trichromatic case by using three boxes.

2 Overview

From now on, an isothetic rectangle enclosing a set of red (resp. blue) points will be called *red rectangle*, denoted by $\mathcal{R}$ (resp. *blue rectangle*, denoted by $\mathcal{B}$). To solve our problem, we first observe that given a bicolored point set S, and two rectangles $\mathcal{R}$ and $\mathcal{B}$ that provide an optimal solution to the *EIEB-problem* for S, there are three types of relative positions of $\mathcal{R}$ with respect to $\mathcal{B}$, up to symmetry. These are depicted in Figure 2 *a.)* , *b.)* and *c.)*. Observe that we reject case *d.)* because it is always possible to move the left side of $\mathcal{R}$ towards the right until we get two disjoint rectangles $(\mathcal{R}', \mathcal{B})$ such that $\mathcal{R}'$ contains as many red points as $\mathcal{R} \setminus \mathcal{B}$ and that the number of blue points in $\mathcal{B}$ is at least the same than it was (it could even be improved, depending on the existence of blue points in the intersection area).

For any pair of rectangles $\mathcal{R}$ and $\mathcal{B}$, not necessarily providing a solution, we call $(\mathcal{R}, \mathcal{B})$ a *corner type* pair if $\mathcal{R}$ overlaps exactly one corner of $\mathcal{B}$; a *sandwich type* pair if $\mathcal{R}$ intersects properly two parallel sides of $\mathcal{B}$; and a *disjoint type* pair if $\mathcal{R}$ and $\mathcal{B}$ can be separated either by a horizontal or a vertical line. A pair $(\mathcal{R}, \mathcal{B})$ is called a *corner solution* if it is a *corner type* pair of rectangles that provides an optimal solution to the *EIEB-problem* for S. Similarly, we define *sandwich* and *disjoint solutions*.

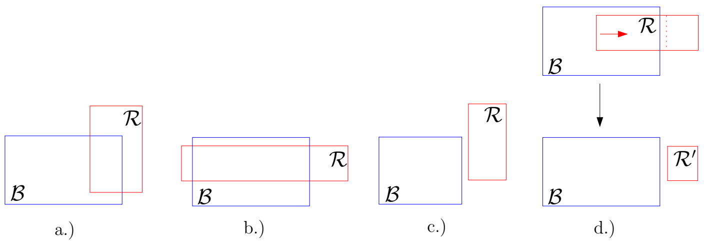
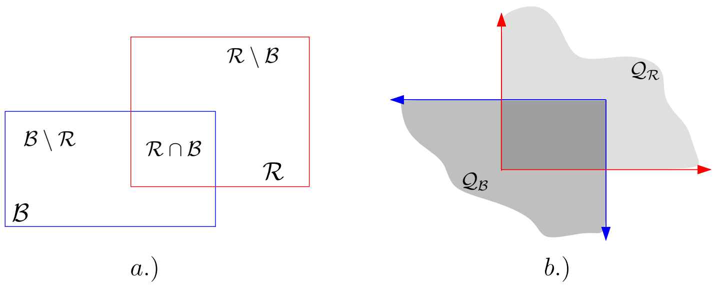


Figure 2: a.) Corner, b.) sandwich and, c.) disjoint type pairs of rectangles, d.) moving to a disjoint case.

We now introduce some properties and a first approach to solve a corner solution that can be stated in a similar way for the other cases as well.

Let $(\mathcal{R}, \mathcal{B})$ be a *corner type* pair of rectangles. This means that for example, $\mathcal{R}$ overlaps the topmost right corner of $\mathcal{B}$ (as in Figure 3 *a.*). We denote by $Red(X)(resp.\ Blue(X))$ the red(*resp.* blue) points of S that are in the set X. We remark that $\mathcal{R}$ and $\mathcal{B}$ are considered to be closed sets. We call a pair $(\mathcal{Q}_{\mathcal{R}}, \mathcal{Q}_{\mathcal{B}})$ a *corner type* pair of quadrants if the relative position of $\mathcal{Q}_{\mathcal{R}}$ with respect to $\mathcal{Q}_{\mathcal{B}}$ is as shown in Figure 3 *b.*). We will assume that the quadrants include their borders. It easy to prove the following result:



$\mathcal{R} \setminus \mathcal{B}$
$\mathcal{B} \setminus \mathcal{R}$
$\mathcal{R} \cap \mathcal{B}$
$\mathcal{R}$
$\mathcal{B}$
a.)
$\mathcal{Q}_{\mathcal{R}}$
$\mathcal{Q}_{\mathcal{B}}$
b.)

Figure 3: a.) A *corner type* pair of rectangles b.) A *corner type* pair of quadrants.

Proposition 1. There exists a corner solution $(\mathcal{R}, \mathcal{B})$ to the EIEB-problem for a point set S if, and only if there exists a corner type pair of quadrants $(\mathcal{Q}_{\mathcal{R}}, \mathcal{Q}_{\mathcal{B}})$ that maximizes the sum $|Red(\mathcal{Q}_{\mathcal{R}} \setminus \mathcal{Q}_{\mathcal{B}})| + |Blue(\mathcal{Q}_{\mathcal{B}} \setminus \mathcal{Q}_{\mathcal{R}})|$ over all the possible choices of $\mathcal{Q}_{\mathcal{R}}$ and $\mathcal{Q}_{\mathcal{B}}$.

As a consequence, we will simplify our search by considering *corner type* pairs of quadrants instead of rectangles. We then reformulate our problem as follows: find the pair of quadrants $(\mathcal{Q}_{\mathcal{R}}, \mathcal{Q}_{\mathcal{B}})$ that maximize the sum $|Red(\mathcal{Q}_{\mathcal{R}} \setminus \mathcal{Q}_{\mathcal{B}})| + |Blue(\mathcal{Q}_{\mathcal{B}} \setminus \mathcal{Q}_{\mathcal{R}})|$. We call such a pair $(\mathcal{Q}_{\mathcal{R}}, \mathcal{Q}_{\mathcal{B}})$ an *optimum corner type* pair of quadrants. Next Proposition will lead to a discretization of the problem. We omit the proof in this version.

Proposition 2. Let $(\mathcal{Q}_{\mathcal{R}}, \mathcal{Q}_{\mathcal{B}})$ be a corner type pair of quadrants. Then, it is possible to find another corner type pair $(\hat{\mathcal{Q}}_{\mathcal{R}}, \hat{\mathcal{Q}}_{\mathcal{B}})$ such that $\hat{\mathcal{Q}}_{\mathcal{R}} \setminus \hat{\mathcal{Q}}_{\mathcal{B}}$ (resp. $\hat{\mathcal{Q}}_{\mathcal{B}} \setminus \hat{\mathcal{Q}}_{\mathcal{R}}$) contains at least $|Red(\mathcal{Q}_{\mathcal{R}} \setminus \mathcal{Q}_{\mathcal{B}})|$ red points (resp. $|Blue(\mathcal{Q}_{\mathcal{B}} \setminus \mathcal{Q}_{\mathcal{R}})|$ blue points) and the sides of $\hat{\mathcal{Q}}_{\mathcal{R}}$ (resp. $\hat{\mathcal{Q}}_{\mathcal{B}}$) go through red (resp. blue) points.

Corollary 1. There exists an optimum corner type pair of quadrants $(\mathcal{Q}_{\mathcal{R}}, \mathcal{Q}_{\mathcal{B}})$ such that the sides of $\mathcal{Q}_{\mathcal{R}}$ (resp. $\mathcal{Q}_{\mathcal{B}}$) go through red (resp. blue) points.

As a consequence of the previous results, in the following any quadrant $\mathcal{Q}_{\mathcal{R}}$ (resp. $\mathcal{Q}_{\mathcal{B}}$) will be considered to be determined by red (resp. blue) points of S.

We describe a simple method to compute a corner solution in $O(n^4)$ time and $O(n^2)$ space by considering all possible corner-type pair of rectangles. For each $p \in \mathbb{R}^2$, we denote as SW(p), SE(p), NW(p)and NE(p) the South-West, South-East, North-West, North-East Quadrants with respect to p (Figure 4).

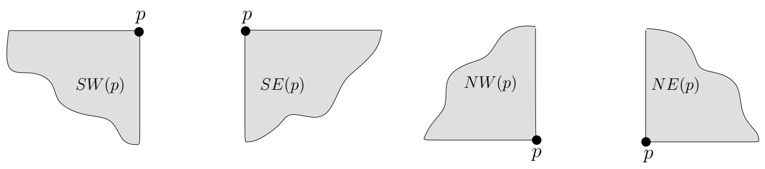


Figure 4: Quadrants with vertex on p.

Consider now the orthogonal grid generated by drawing horizontal and vertical lines through the elements of S. By using *range search* techniques [3], it is possible to perform a quadratic preprocessing on the nodes of the grid and to store the number of red and blue points of S laying in the four quadrants with vertex in each node.

Let $(\mathcal{Q_R}, \mathcal{Q_B})$ be a corner type pair of quadrants and denote by I_1, I_2, I_3 and I_4 to the vertices (in clockwise order) of the rectangular overlapped region (see Figure 5).

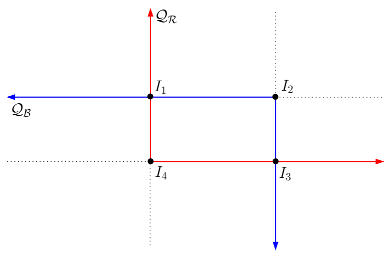


Figure 5: Looking for a *corner solution* in $O(n^4)$ time.

Then, $|Red(\mathcal{Q_R} \setminus \mathcal{Q_B})| = |Red(NE(I_1))| + |Red(NE(I_3))| - |Red(NE(I_2))|$ and

$|Blue(\mathcal{Q_B} \setminus \mathcal{Q_R})| = |Blue(SW(I_1))| + |Blue(SW(I_3))| - |Blue(SW(I_4))|$.

Therefore, obtaining a pair of quadrants $(\mathcal{Q_R}, \mathcal{Q_B})$ that maximizes the sum $|Red(\mathcal{Q_R} \setminus \mathcal{Q_B})| + |Blue(\mathcal{Q_B} \setminus \mathcal{Q_R})|$ can be done in $O(n^4)$ time.

Theorem 2.1. *A corner solution can be found in $O(n^4)$ time, given $O(n^2)$ preprocessing time and space.*

3 The Maximum Consecutive Subsequence

In this section we describe the main tool that allows to solve the problem efficiently and, the technique can be applied to other variants of the problem. The key idea to solve more efficiently the *EIEB-problem* is that, for all of the cases, we will reduce our two-dimensional problem to some instances of the following one-dimensional problem:

Maximum Consecutive Subsequence (MCS): Given a sequence $x_1, x_2, ..., x_n$ and a weight function w over its elements such that $w(x_i) \in \{-1, 0, 1\}$, compute the subsequence $x_i, x_{i+1}, ..., x_j$ of consecutive elements such that $w(x_i) + w(x_{i+1}) + \ldots + w(x_j)$ is maximized either on the entire sequence or so that it contains a specific element x_k.

Proposition 3. The *MCS* problem can be solved in linear time.

Proof. The proof is based on the same techniques used to solve Bentley's *Maximum segment sum* problem [2]. It suffices to observe that $w(x_i) + w(x_{i+1}) + \ldots + w(x_j)$ can be computed as the difference between the cumulative weight sums of $x_1, x_2, ..., x_j$ and $x_1, x_2, ..., x_{i-1}$. □

Next, we propose an $O(n)$-space data structure, the *Maximum Consecutive Subsequence Tree*, *MCS-Tree*, to dynamically maintain the optimal weight sum subsequence of the entire sequence. When the weight of an element x_i is changed, the optimal value is recalculated in $O(\log n)$ time. Our structure also permits to obtain in $O(\log n)$ time the optimal weight sum subsequence that includes a specific element x_k. By using *MCS-Tree*, we shall solve the *Empty Intersection Enclosing Boxes* problem in $O(n^2 \log n)$ time and $O(n)$ space.

3.1 The MCS-Tree

A *MCS-Tree* is a balanced binary tree with n leaves that represents the sequence $x_1, x_2, ..., x_n$ as follows: the kth leaf from left to right represents x_k and each internal node represents the subsequence interval corresponding to its leaves (see Figure 6).

Each node N stores the following intervals: $I(N)$: The interval it represents (If N is a leaf representing x_k, $I(N) = [x_k]$); $L(N)$: The non empty interval of maximum weight sum that is a prefix of

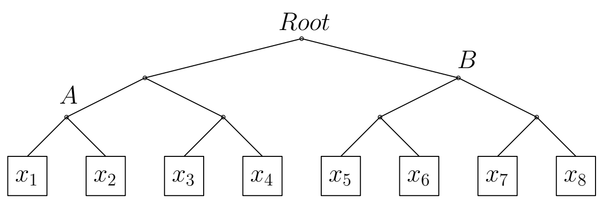


Figure 6: *MCS-Tree* for a sequence of 8 elements. Nodes A, B and *Root* represent intervals $[x_1, x_2]$, $[x_5, x_6, x_7, x_8]$ and $[x_1, x_2, ..., x_8]$ respectively

$I(N)$; $R(N)$: The non empty interval of maximum weight sum that is a suffix of $I(N)$; $M(N)$: The non empty interval of maximum weight sum that is a subinterval of $I(N)$.

Clearly, we have that $I(Root)$ is $[x_1, x_2, ..., x_n]$ and $L(Root)$, $R(Root)$ and $M(Root)$ are respectively the intervals of the form $[x_1, x_2, ..., x_j]$, $[x_i, x_{i+1}, ..., x_n]$ and $[x_i, x_{i+1}, ..., x_j]$ that have maximum weight sum in the given sequence.

The tree has linear complexity and t he following operators are used for computing *MCS-Tree*:

- $+$: returns in $O(1)$ time the join between two contiguous intervals.
- $MaxSum\{I_1, I_2, ..., I_m\}$: returns in O(1) time the interval of maximum weight sum between the intervals $I_1, I_2, ..., I_m$.
- $CreateTree(X)$: given the sequence $X = \{x_1, x_2, ..., x_n\}$, creates in $O(n)$ time a *MCS-Tree* with empty data in all its nodes.
- $UpdateNode(N)$: creates(updates) in $O(1)$ time the internal data of node N if it is a leaf or if the data of its two children have already been created(updated).
- *UpdateTree*: performs in $O(n)$ time a complete update. It is done by making a *bottom-up* procedure from the leaves to the root invoking the operator *UpdateNode*.
- $UpdateLeaf(x_k)$: given an element x_k of the sequence in where $w(x_k)$ has been changed, performs *UpdateNode* at the nodes that are in the unique path from the leaf corresponding to x_k, to the root. It is done in $O(\log n)$ time.
- *MaxSumInterval*: returns $M(Root)$ in $O(1)$ time.
- *MaxSumLeftInterval*: returns $L(Root)$ in $O(1)$ time.
- *MaxSumRightInterval*: returns $R(Root)$ in $O(1)$ time.
- $MaxSumIntervalContains(x_k)$: computes in $O(\log n)$ time the interval of maximum weight sum that includes x_k.

Proposition 4. Given the sequence $X = \{x_1, x_2, \ldots, x_n\}$, the MCS-Tree can be built in $O(n)$ time and when some $w(x_i)$ is changed, the interval of maximum weight sum of consecutive elements can be updated in $O(\log n)$. The data structure also permits to compute in $O(\log n)$ time the interval of maximum weight sum of consecutive elements that includes a specific element x_k

4 Computing a solution in $O(n^2 \log n)$ time

In this Section, we are ready to solve more efficiently the problem. First, we consider a *corner type*-solution and the other cases are considered later. Let $(\mathcal{R}, \mathcal{B})$ be a *corner type*-solution over all the possible choices of *corner type* pairs of rectangles.

Consider again the orthogonal grid generated by drawing horizontal and vertical lines through the elements of the data set S. Suppose that these lines are colored red or blue according to the color of the point in S they contain. We observe that our problem can be reduced to solve $O(n^2)$ instances of the following problem:

Given a red-blue horizontal strip, that is, two horizontal blue (above) and red (below) lines, we want to find a red-blue vertical strip, that is, two vertical blue (right) and red (left) lines such that the corresponding red and blue quadrants are optimum.

By Corollary 1, we assume that the lines pass through points of the corresponding color. The key idea for our approach is to reduce this two-dimensional problem to the one-dimensional problem introduced in the above Section, the *Maximum Consecutive Subsequence (MCS)* problem.

The procedure is as follows. For a starting horizontal strip bounded above by a blue line h_b and below by a red line h_r, we will compute in linear time a pair of vertical lines (v_r, v_b) that provides a pair of quadrants $(\mathcal{Q}_\mathcal{R}, \mathcal{Q}_\mathcal{B})$ maximizing $|Red(\mathcal{Q}_\mathcal{R} \setminus \mathcal{Q}_\mathcal{B})| + |Blue(\mathcal{Q}_\mathcal{B} \setminus \mathcal{Q}_\mathcal{R})|$, over all the possible choices of v_r and v_b. Thus, by using *MSC-Tree* we can sweep a red horizontal line and update the solution in $O(n \log n)$ time. Finally, by repeating this process for each of the $O(n)$ blue lines, we obtain the claimed complexity.

Let us give some details. Consider an initial horizontal strip, bounded above by a blue line h_b and below by a red line h_r (see Figure 7 *a.*). Let H_b and H_r be the points onto h_b and h_r respectively. Let $v_b = v_r = v_0$ the vertical line passing through H_b and denote by I_1 the intersection point between v_r and h_r. Lines h_b, h_r and v_0 divide S into six subsets denoted by S_1 S_2, S_3, S_4, S_5, and S_6 as in Figure 7. Consider a red quadrant $\mathcal{Q}_{\mathcal{R}_0} = NE(I_1)$ and a blue one $\mathcal{Q}_{\mathcal{B}_0} = SW(H_b)$. $\mathcal{Q}_{\mathcal{R}_0}$ and $\mathcal{Q}_{\mathcal{B}_0}$ form a candidate corner solution $\mathcal{CS}_0$ (because they have point H_b in common) with assigned value $|Red(\mathcal{Q}_{\mathcal{R}_0} \setminus \mathcal{Q}_{\mathcal{B}_0})| + |Blue(\mathcal{Q}_{\mathcal{B}_0} \setminus \mathcal{Q}_{\mathcal{R}_0})|$. In order to improve this solution we move v_r to the left and v_b to the right (see Figure 7 *b.*). When v_r is moved to the left, the red points in S_1 that are between v_r and v_0 will be considered inside $\mathcal{CS}_0$ (*i.e.* the value of $\mathcal{CS}_0$ is increased in one for each of them) and the blue points in S_3 that are also between v_r and the initial position v_0 will be discarded from $\mathcal{CS}_0$ (*i.e.* the value of $\mathcal{CS}_0$ is decreased in one for each of them). In a similar way, when v_b is moved to the right, the blue points in S_2 that are between v_0 and v_b will be considered and the red points in S_4 that are between v_0 and v_b will be discarded.

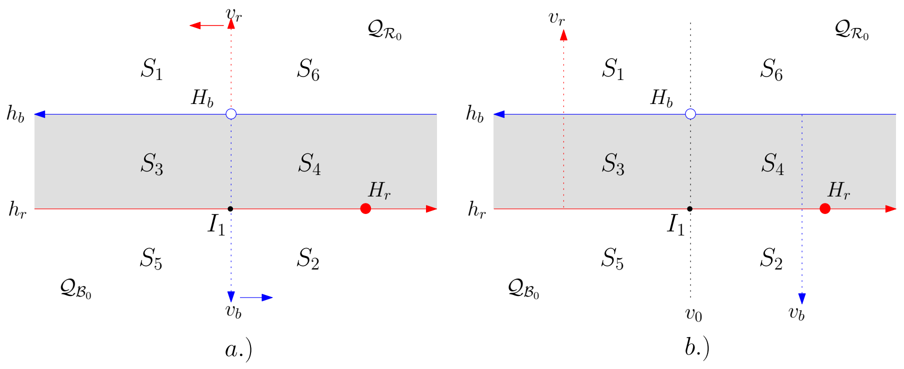


Figure 7: a.) Initial b.) Improved

We assign weights to the elements of S according to how each of them affects the value of the candidate solution $\mathcal{CS}_0$ when v_r and v_b are moved to the left and to the right, respectively. The assignment of weights to points in S is as follows:

- Red points in S_1 (*resp.* S_4) have weight 1 (*resp.* -1) because they can be included into (*resp.* excluded from) $\mathcal{CS}_0$ when v_r (*resp.* v_b) is moved to the left (*resp.* right).
- Blue points in S_2 (*resp.* S_3) have weight 1 (*resp.* -1) because they can be included into (*resp.* excluded from) $\mathcal{CS}_0$ when v_b (*resp.* v_r) is moved to the right (*resp.* left).
- The rest of the elements of S will have weight 0 because they do not affect the value of $\mathcal{CS}_0$ when v_r and v_b are moved.

Once the weights have been assigned, we observe that if we have the elements of S sorted by their $x-$coordinate giving the sequence X_S, the points in S that indicate where to move v_r and v_b to improve $\mathcal{CS}_0$ in the best way are respectively, the beginning and the end of the interval of consecutive elements in X_S that has maximum weight sum and contains the element H_b. Thus in our approach, we start by sorting the points of S to obtain X_S and then, we build the *MCS-Tree*. By using *MCS-Tree*, the interval of maximum weight sum containing the current H_b is computed in $O(\log n)$ time per strip. We update our tree invoking $UpdateTree$ every time in which the current H_b changes and making an $UpdateLeaf(P)$ for at most all points P below h_b for which the red sweeping-line h_r passes through. The algorithm computes a corner solution in $O(n^2 \log n)$ time and $O(n)$ space.

By using a similar algorithm, the sandwich solution can be found in $O(n^2 \log n)$ time and $O(n)$ space and, the disjoint case can be found in $O(n \log n)$ time and $O(n)$ space. Combining the results from the three preceding cases, we arrive to the main result of this section:

Theorem 4.1. *A solution for the EIEB-problem can be found in $O(n^2 \log n)$ time and $O(n)$ space.*

5 The three-dimensional case

In the *EIEB-problem* in $I\!R^3$ we have more solution types than in $I\!R^2$ but the good new is that we can transform each of them to problems in $I\!R^2$. Furthermore, the two-dimensional problems can be solved with our techniques. We proceed to explain how to transform a *Corner Solution* case in $I\!R^3$ to some similar *Corner Solution* cases in $I\!R^2$.

Cubes $\mathcal{R}$ and $\mathcal{B}$ form a *corner type* pair of cubes if they intersect each other in a corner (see Figure 8 *a.*). The idea to obtain a corner solution is as follows. For each pair of horizontal planes $(\mathcal{P}_\mathcal{R}, \mathcal{P}_\mathcal{B})$ ($\mathcal{P}_\mathcal{R}$ passing through a red point and $\mathcal{P}_\mathcal{B}$ passing through a blue point as shown in Figure 8 *b.*), we consider the orthogonal projection of the blue points below or in $\mathcal{P}_\mathcal{B}$ and the red points above or in $\mathcal{P}_\mathcal{R}$ onto a horizontal plane $\mathcal{P}$. Let T_1 be the set of projected points x such that x is blue and comes from below $\mathcal{P}_\mathcal{R}$ or x is red and comes from above $\mathcal{P}_\mathcal{B}$. Let T_2 be the set of the rest of the projected points. Then, we have to find in $\mathcal{P}$ a pair of quadrants $(\mathcal{Q}_\mathcal{R}, \mathcal{Q}_\mathcal{B})$ (see Figure 8 *c.*) maximizing $|Red(\mathcal{Q}_\mathcal{R} \cap T_1)| + |Blue(\mathcal{Q}_\mathcal{B} \cap T_1)| + |Red((\mathcal{Q}_\mathcal{R} \setminus \mathcal{Q}_\mathcal{B}) \cap T_2)| + |Blue((\mathcal{Q}_\mathcal{B} \setminus \mathcal{Q}_\mathcal{R}) \cap T_2)|$.

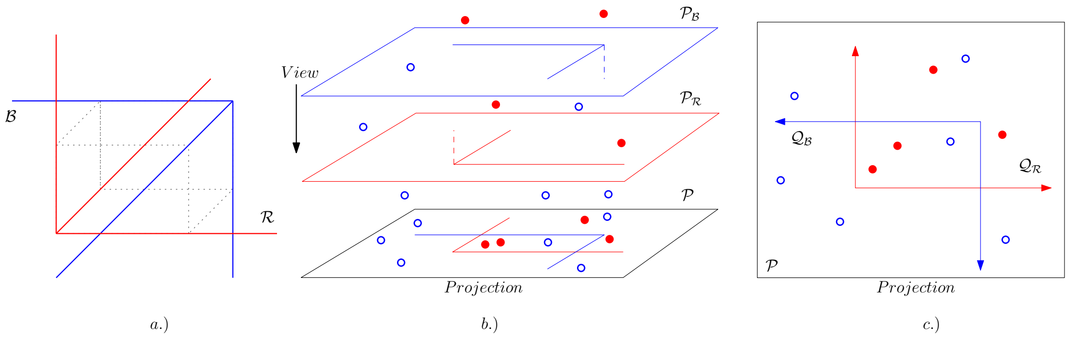

$\mathcal{B}$
$\mathcal{R}$
a.)
View
$\mathcal{P}_\mathcal{B}$
$\mathcal{P}_\mathcal{R}$
$\mathcal{P}$
Projection
b.)
$\mathcal{Q}_\mathcal{B}$
$\mathcal{Q}_\mathcal{R}$
$\mathcal{P}$
Projection
c.)

Figure 8: Corner Solution in $I\!R^3$ is transformed to $I\!R^2$

This planar problem can be solved in $O(n^2 \log n)$ time and $O(n)$ space by using a similar method to the corner solution in above Section but assigning weights to the points of $\mathcal{P}$ in a convenient way (see again Figure 7).

Thus, this case in $I\!R^3$ can be reduced to solve $O(n^2)$ instances of problems in $I\!R^2$ and we have the following result:

Theorem 5.1. *A Solution to EIEB-problem in $I\!R^3$ can be found in $O(n^4 \log n)$ time and $O(n)$ space.*

6 The trichromatic case with three disjoint boxes

In this section we study the following problem as an extension of *EIEB-Problem*:

Let S be a trichromatic point set on the plane in general position such that its elements are colored red, blue or green. Compute three isothetic pair-wise disjoint rectangles $\mathcal{B}$, $\mathcal{R}$ and $\mathcal{G}$ such that $|Blue(\mathcal{B})| + |Red(\mathcal{R})| + |Green(\mathcal{G})|$ is maximum.

This problem is named the *Disjoint Trichromatic Enclosing Boxes* problem (*DTEB-Problem*). We observe that $\mathcal{B}$, $\mathcal{R}$ and $\mathcal{G}$ must be separated by either two parallel isothetic lines (Figure 9 *a.*) or two perpendicular isothetic lines (Figure 9 *b.*). Thus we have two possible configurations for a solution and we have to solve two different problems.

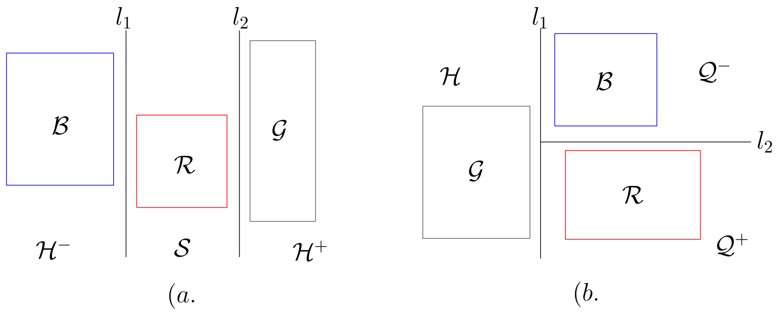


Figure 9: Separated by two isothetic lines a.) Case 1: parallel b.) Case 2: perpendicular

Case 1: Compute two vertical lines l_1 and l_2 that partition the plane into three regions: a strip $\mathcal{S}$ and two half-planes $\mathcal{H}^-$ and $\mathcal{H}^+$, and a permutation $\mathcal{P} = [\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3]$ of $\{\mathcal{H}^-, \mathcal{S}, \mathcal{H}^+\}$ such that $|Blue(\mathcal{R}_1)| + |Red(\mathcal{R}_2)| + |Green(\mathcal{R}_3)|$ is maximum. Once l_1, l_2 and $\mathcal{P}$ are found, blue, red and green points in $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$ are enclosed with boxes $\mathcal{B}$, $\mathcal{R}$ and $\mathcal{G}$ respectively. (see Figure 9 *a.*)

Case 2: Similar to *Case 1* but finding a vertical line l_1 and a horizontal half-line l_2 partitioning the plane into three regions: a half-plane $\mathcal{H}$ and two quadrants $\mathcal{Q}^-$ and $\mathcal{Q}^+$, and a permutation $\mathcal{P} = [\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3]$ of $\{\mathcal{H}, \mathcal{Q}^-, \mathcal{Q}^+\}$. (see Figure 9 *b.*)

Both problems can be solved in $O(n \log n)$ time and $O(n)$ space. In *Case 1*, we can reduce it to some instances of the *Longest Increasing Subsequence Problem* [4] and, for *Case 2*, we use a modification of our structure *MCS-Tree*.

Theorem 6.1. *A Solution to the DTEB-Problem can be found in $O(n \log n)$ time and $O(n)$ space.*

References

[1] B. Aronov, S. Har-Peled. On approximating the depth and related problems. *Proceedings 16th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA 2005), 2005.

[2] J. Bentley, "Programming pearls: algorithm design techniques," Comm. ACM, vol. 27, no. 9, pp. 865-873, 1984.

[3] J.L. Bentley and M.I. Shamos, *A problem in multivariate statistics: Algorithms, data structure and applications*, Proceedings of the 15th annual Allerton Conference on Communications, Control, and Computing, pp. 193-201, 1977.

[4] Bespamyatnikh S., Segal M., Enumerating longest increasing subsequences and patience sorting, Information Processing Letters, 76, 7-11, 2000.

[5] C. Cortés, J. M. Díaz-Báñez, and J. Urrutia. Finding enclosing boxes with empty intersection. *Proceedings of the 23rd. European Workshop on Computational Geometry*, Delphi (Greece), pp. 185–188, 2006.

[6] R. Duda, P. Hart, D. Stork. *Pattern Classification. John Wiley and Sons, Inc.*, New York, 2001.

[7] Eckstein, J., Hammer, P.L., Liu, Y., Nediak,M., Simeone, B. The maximum box problem and its applications to data analysis. Comput. Optim. Appl. 23, 285-298, 2002.

[8] Hammer P.L., Bonates, T. Logical analysis of data: from combinatorial optimization to medical applicatios. Rutcor Research Report, RRR 10-2005. Rutger University, New Jersey, USA. 2005.

[9] Hand,H., Mannila, H., Smyth, P. Principles of Data Mining. The MIT Press, 2001.