

Bichromatic separability with two boxes: a general approach

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

6th February 2009

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

†**Corresponding author:** Departamento Matemática Aplicada II, Universidad de Sevilla, Escuela Superior de Ingenieros, Avda. de los Descubrimientos, s/n, 41092, Sevilla, SPAIN. e-mail: dbanez@us.es, phone: 34-95-4486172, fax: 34-95-4486165, partially supported by Grant MEC MTM2006-03909.

‡Departamento de Computación, Universidad de La Habana, pablo@matcom.uh.cu, partially supported by Grant MAEC-AECI and MEC 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 MEC MTM2006-03909.

||Departamento Matemática Aplicada II, Universidad de Sevilla, iventura@us.es, partially supported by Grant MEC MTM2006-03909.

Abstract

Let S be a set of n points on the plane in general position such that its elements are colored red or blue. We study the following problem: *Find a largest subset of S which can be enclosed by the union of two, not necessarily disjoint, axis-aligned rectangles \mathcal{R} and \mathcal{B} such that \mathcal{R} (resp. \mathcal{B}) contains only red (resp. blue) points.* We prove that this problem can be solved in $O(n^2 \log n)$ time and $O(n)$ space. Our approach is based on solving some instances of Bentley's maximum-sum consecutive subsequence problem. We introduce the first known data structure to dynamically maintain the optimal solution of this problem. We show that our techniques can be used to efficiently solve a more general class of problems in data analysis.

Keywords: Discrete optimization, Maximum consecutive subsequence, Dynamic maintenance, Algorithm design, Classification problems, Bioinformatics.

1 Introduction

In data mining and classification problems, a natural method for analyzing data is to select prototypes representing different data classes. A standard technique for achieving this is to perform cluster analysis on the training data [8, 11]. The clusters can be obtained using simple geometric shapes such as circles or boxes. Aronov and Har-Peled [1] and Eckstein et al. [9] considered circles and axis-aligned boxes for the selection problem. Aronov and Har-Peled [1] studied the following problem: *Given a bicolored point set, find a ball that contains the maximum number of red points without containing any blue points.* This type of classification is *asymmetric* in the way it treats red and blue points. The goal is to separate the “red class” from the “blue class”. We are interested in generalizing to a *symmetric two-classes problem* by finding a witness set for each color.

In some cases, as in medical data analysis [10], methods can produce biased 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 which are hard to classify due to strong similarities between subsets of different classes. A potential way to find a better classification in the former problem is to remove some data points from the input. Culling the minimum number of such points can be a suitable criterion for losing the smallest amount of information possible. We will use the following notation. Given two (possibly unbounded) non-disjoint convex sets X and Y , we denote by $|X|$ the cardinality of the set X and by $X \setminus Y$ the set obtained by removing $X \cap Y$ from X .

In this paper we study the following problem:

The Two Enclosing Boxes problem (2-EB-problem): *Let S be a set of n points on the plane in general position such that the points are colored red or blue. Compute two open axis-aligned rectangles \mathcal{R} and \mathcal{B} such that the number of red points in $\mathcal{R} \setminus \mathcal{B}$ plus the number of blue points in $\mathcal{B} \setminus \mathcal{R}$ is maximized.*

We remark here that in the definition of our problem we require \mathcal{R} and \mathcal{B} to be open. This will facilitate our presentation, but in practice we may proceed in a similar way if our boxes are closed. Observe that \mathcal{R} and \mathcal{B} may intersect; however any point in $\mathcal{R} \cap \mathcal{B}$ has to be removed from S . For example, the solution to the 2-EB-problem for the point set S illustrated in Figure 1 is $n - 2$, where n is the cardinality of the input set. By removing r_1 and b_1 from S , we can obtain two rectangles \mathcal{R} and \mathcal{B} , each of them containing only red and blue points respectively. Notice that an asymmetric separation approach as the one used by Aronov and Har-Peled [1] does not give a solution to our problem, so we must design a procedure which consider \mathcal{R} and \mathcal{B} simultaneously. Bespamyatnikh and Segal [4] studied a two-box covering problem but using a different min-max criterion.

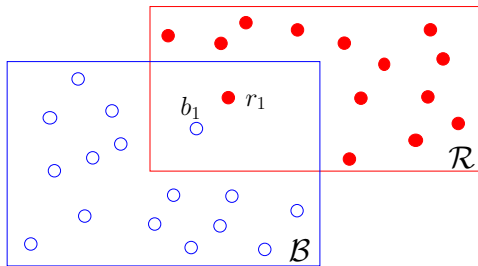


Figure 1: Getting a solution by removing the points r_1 and b_1 .

The 2-EB-problem was first introduced by Cortés et al. [5], solving the problem with an $O(n^3)$ -time and space algorithm. In this paper we show that the 2-EB-problem can be solved in $O(n^2 \log n)$ time and $O(n)$ space. We also introduce a new data structure that allows us to dynamically solve Bentley’s [2] well known *Maximum-Sum Consecutive Subsequence* problem (MCS-problem

for short) together with some other variants of this problem that have applications, for instance in sequence analysis in bioinformatics [6]. We also show a generalization of our approach that can be used to solve a general type of problem of interest in areas such as computer graphics or machine learning [7].

The outline of this paper is as follows. In Section 2 we introduce a data structure, the MCS-tree, to dynamically maintain an optimal solution of the MCS-problem. In Section 3 we introduce some notation and present the first results on the 2-EB-problem. In Section 4 we show our main result, an $O(n^2 \log n)$ time and linear space algorithm to solve the 2-EB-problem. In Section 5 we show how to solve the following related problem: *Let S be a set of points on the plane in general position such that each element of S is colored red, blue, or green. Find three pairwise-disjoint axis-aligned rectangles \mathcal{R} , \mathcal{B} , and \mathcal{G} such that the total number of red, blue, and green points contained in \mathcal{R} , \mathcal{B} , and \mathcal{G} respectively is maximized.* In Section 6 we present a generalization of our technique and show how to apply it to several variants of the original problem. Finally, in Section 7 we present the conclusions.

2 The dynamic MCS-problem

In this section we describe the main tool that will allow us to solve the 2-EB-problem in $O(n^2 \log n)$ time and $O(n)$ space. Later we show how this technique can be applied to other variants of the same problem. The key idea for solving the 2-EB-problem is a reduction to the computation of some dynamic instances of the following one-dimensional Bentley's problem [2]:

The Maximum-Sum Consecutive Subsequence problem (MCS-problem): *Given a sequence $X = (x_1, x_2, \dots, x_n)$ and a real weight function w over its elements where for each i , $w(x_i)$ is not necessarily positive, compute the consecutive subsequence $(x_i, x_{i+1}, \dots, x_j)$ of X such that $w(x_i) + w(x_{i+1}) + \dots + w(x_j)$ is maximum.*

Next we show how to construct a binary tree, the MCS-tree, that allows us to solve the MCS-problem in a dynamic way. Assume that n is a power of two, otherwise add a few elements with negative weights at the end of the sequence $X = (x_1, \dots, x_n)$ until we get a sequence of 2^k elements, $n < 2^k < 2n$.

2.1 The MCS-tree

The MCS-tree is a balanced binary tree with n leaves representing the sequence $X = (x_1, x_2, \dots, x_n)$. The k -th leaf (from left to right) represents x_k . Each internal node u represents the consecutive subsequence formed by the descendants (leaves) of u . At each node u of the MCS-tree, we store some values, one of which will be the weight of the *maximum weight consecutive subsequence* contained in the subsequence of X defined by the descendants of u . The root vertex will have the solution of the MCS-problem.

Because we focus on the indexes of the *first* and the *last* elements of a subsequence, we use the term *interval* $[x_i, x_j]$ to denote the consecutive subsequence $(x_i, x_{i+1}, \dots, x_j)$ (Figure 2).

Construction of the MCS-tree. We build the MCS-tree as follows: Each node u stores the following intervals, as well as their weights, i.e., the sum of their elements:

1. $I(u)$: The interval formed by the descendants of u . If u is a leaf representing x_i , $I(u) = [x_i]$.
2. $L(u)$: The interval of maximum weight sum contained in $I(u)$ that contains the leftmost

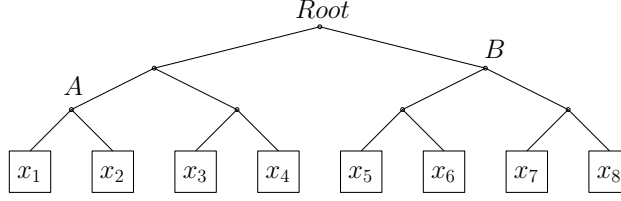


Figure 2: The MCS-tree for a sequence of 8 elements. The nodes A , B , and $Root$ represent the intervals $[x_1, x_2]$, $[x_5, x_8]$, and $[x_1, x_8]$, respectively.

element of $I(u)$. If all the intervals in $I(u)$ containing the leftmost element of $I(u)$ have negative weight, set $L(u) = \emptyset$ and its weight to 0.

3. $R(u)$: The interval of maximum weight sum contained in $I(u)$ that contains the rightmost element of $I(u)$. If all the intervals in $I(u)$ containing the rightmost element of $I(u)$ have negative weight, set $R(u) = \emptyset$ and its weight to 0.
4. $M(u)$: The interval of maximum weight sum that is a subinterval of $I(u)$. If all the intervals in $I(u)$ have negative weight, set $M(u) = \emptyset$ and its weight to 0.

If u is the root of the MCS-tree, then $I(u)$ is $[x_1, x_n]$. $L(u)$, $R(u)$, and $M(u)$ are respectively the intervals of the form $[x_1, x_i]$, $[x_j, x_n]$, and $[x_i, x_j]$ with maximum weight sum. We only store the indexes of the *first* and *last* elements of each of the above intervals.

Let u be an internal node of the MCS-tree, and let v and w be its left and right children, respectively. It is clear that if we have the values for $I(v)$, $I(w)$, $L(v)$, $L(w)$, $R(v)$, $R(w)$, $M(v)$, and $M(w)$, then we can calculate in constant time each of the values of $I(u)$, $L(u)$, $R(u)$, and $M(u)$. Notice that for $M(u)$ we have three possible cases: (i) $M(u)$ is contained in $I(v)$; (ii) $M(u)$ is contained in $I(w)$; or (iii) $M(u)$ overlaps both of $I(v)$ and $I(w)$. Thus, $M(u) = M(v)$, $M(u) = M(w)$, or $M(u) = R(v) \cup L(w)$ respectively. We choose $M(u)$ to be the interval of maximum weight among $M(v)$, $M(w)$, and $R(v) \cup L(w)$. The intervals $L(u)$ and $R(u)$ can be computed in a similar way.

If a leaf x_i of the MCS-tree has negative weight $w(x_i)$, then $L(x_i) = \emptyset$, $R(x_i) = \emptyset$ and $M(x_i) = \emptyset$. Otherwise $L(x_i) = [x_i]$, $R(x_i) = [x_i]$ and $M(x_i) = [x_i]$. By using these values for the leaves of the MCS-tree and a standard bottom-up traversal, we can calculate $L(u)$, $R(u)$, and $M(u)$ for all the nodes u of the tree in linear time. If the weight $w(x_i)$ of a leaf x_i changes, then the MCS-tree can be updated in a bottom-up traversal from x_i to the root, and therefore the weight of the maximum weight consecutive subsequence is recalculated in $O(\log n)$ time.

Moreover, for a given $x_i \in X$, by traversing from x_i to the root, the MCS-tree structure allows us to obtain in $O(\log n)$ time: (i) the optimal interval for the MCS-problem which contains x_i , and (ii) the maximum weight of the interval of X starting, or ending at x_i . We leave out the details to avoid repetitive arguments.

These properties of the MCS-tree structure will be used to solve the 2-EB-problem. From the discussion above we get the following result.

Theorem 2.1. *Given the sequence $X = (x_1, \dots, x_n)$ with a real weight function w over its elements, the MCS-tree can be built in $O(n)$ time. Moreover, if any $w(x_i)$ is updated, the MCS-tree can be updated in $O(\log n)$. The MCS-tree allows computation, in $O(\log n)$ time, of the interval of maximum weight sum of consecutive elements that includes, starts or ends at a specific element $x_k \in X$.*

3 Notation and preliminary results on the 2-EB-problem

An optimal solution to the 2-EB-problem for S consists of two axis-aligned rectangles \mathcal{R} and \mathcal{B} and the subset of S containing the red points of S in $\mathcal{R} \setminus \mathcal{B}$ together with the blue points of S in $\mathcal{B} \setminus \mathcal{R}$. For technical reasons which become necessary in Proposition 3.2 we assume that \mathcal{R} and \mathcal{B} are *open*, i.e., they do not include their boundaries, thus we do not count points on the boundaries of \mathcal{R} and \mathcal{B} .

Up to symmetry there are three possible relative positions of \mathcal{R} and \mathcal{B} . A pair $(\mathcal{R}, \mathcal{B})$ is called: a *corner-type* if \mathcal{R} overlaps exactly one corner of \mathcal{B} (Figure 3a), a *sandwich-type* if \mathcal{R} intersects only two parallel sides of \mathcal{B} (Figure 3b), and a *disjoint-type* if \mathcal{R} and \mathcal{B} are disjoint (Figure 3c). A fourth type could exist, however it can be reduced to a disjoint-type (Figure 3d).

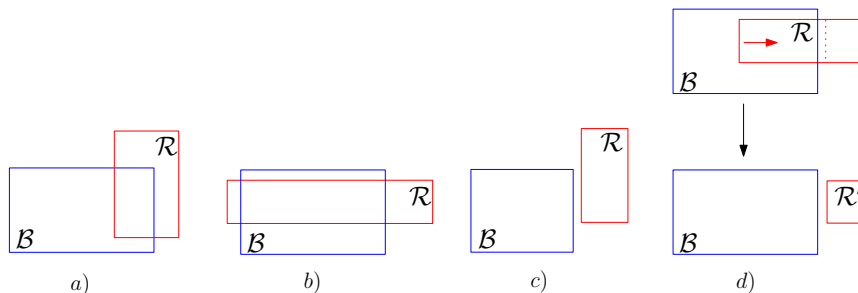


Figure 3: a) Corner-type, b) sandwich-type, c) disjoint-type, and d) getting a disjoint-type.

We show now how to obtain optimal solutions of the corner-type case. In section 4 we describe briefly how to solve sandwich-type solutions, the remaining cases can be handled in a simpler way. Thus, from now on $(\mathcal{R}, \mathcal{B})$ denotes a corner-type pair of rectangles. We assume without loss of generality that \mathcal{R} always contains the top-right corner of \mathcal{B} , as in Figure 4a.

Given $Y \subset \mathbb{R}^2$, $\text{Red}(Y)$ (resp. $\text{Blue}(Y)$) denotes the subset of red (resp. blue) points of S that belong to Y . For $p \in \mathbb{R}^2$, we respectively denote by $\text{SW}(p)$, $\text{SE}(p)$, $\text{NW}(p)$ and $\text{NE}(p)$, the South-West, South-East, North-West and North-East open quadrants with respect to p , e.g., if $p = (a, b)$, then $\text{NE}(p) = \{(x, y) \mid a < x, b < y\}$. The point p is the *apex* of $\text{SW}(p)$, $\text{SE}(p)$, $\text{NW}(p)$, and $\text{NE}(p)$.

Let $(\mathcal{Q}_{\mathcal{R}}, \mathcal{Q}_{\mathcal{B}})$ be a North-East and South-West pair of quadrants. We say that $(\mathcal{Q}_{\mathcal{R}}, \mathcal{Q}_{\mathcal{B}})$ is a *corner-type* pair of quadrants if the apex of $\mathcal{Q}_{\mathcal{R}}$ belongs to $\mathcal{Q}_{\mathcal{B}}$ (Figure 4b).

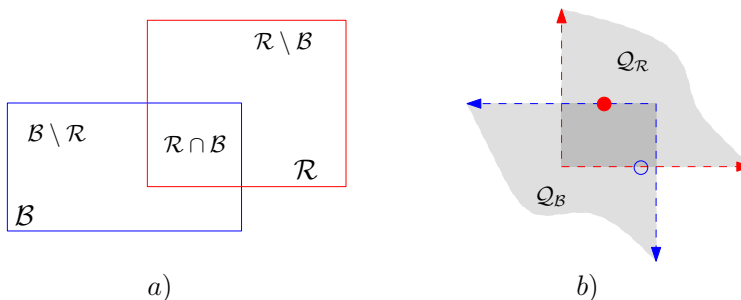


Figure 4: a) A corner-type pair of rectangles, b) a corner-type pair of quadrants.

Proposition 3.1. *If the 2-EB-problem for S has an optimal solution formed by a corner-type pair of rectangles, then it has an optimal solution formed by a corner-type pair of quadrants, or there is an optimal disjoint type solution.*

Proof. Let $(\mathcal{R}, \mathcal{B})$ be an optimal corner-type pair of rectangles for S . Let $\mathcal{Q}_{\mathcal{R}}$ be the North-East quadrant whose apex is the bottom-left corner of \mathcal{R} and let $\mathcal{Q}_{\mathcal{B}}$ be the South-West quadrant whose apex is the top-right corner of \mathcal{B} (Figure 4). Notice that $|\text{Red}(\mathcal{Q}_{\mathcal{R}} \setminus \mathcal{R})| = |\text{Blue}(\mathcal{Q}_{\mathcal{B}} \setminus \mathcal{B})| = 0$, otherwise $(\mathcal{R}, \mathcal{B})$ would not form an optimal corner-type solution for S . Hence $|\text{Red}(\mathcal{Q}_{\mathcal{R}} \setminus \mathcal{Q}_{\mathcal{B}})| + |\text{Blue}(\mathcal{Q}_{\mathcal{B}} \setminus \mathcal{Q}_{\mathcal{R}})| = |\text{Red}(\mathcal{R} \setminus \mathcal{B})| + |\text{Blue}(\mathcal{B} \setminus \mathcal{R})|$ and thus, the corner-type pair of quadrants $(\mathcal{Q}_{\mathcal{R}}, \mathcal{Q}_{\mathcal{B}})$ is an optimal solution for S . \square

Proposition 3.2. *There exists an optimal corner-type pair of quadrants $(\mathcal{Q}_{\mathcal{R}}, \mathcal{Q}_{\mathcal{B}})$ of the 2-EB-problem for S such that the horizontal ray bounding $\mathcal{Q}_{\mathcal{B}}$ contains a red point and the horizontal ray bounding $\mathcal{Q}_{\mathcal{R}}$ contains a blue point.*

Proof. Let $(\mathcal{Q}_{\mathcal{R}}, \mathcal{Q}_{\mathcal{B}})$ be an optimal corner-type pair of quadrants of the 2-EB-problem for S . Let $S' \subseteq S$ be the set of points of S in $\mathcal{Q}_{\mathcal{R}} \setminus \mathcal{Q}_{\mathcal{B}}$. Translate $\mathcal{Q}_{\mathcal{B}}$ vertically in the upward direction until its boundary hits a point $p \in S'$. If p is a blue point, ignore it (as it will not change the solution given by $\mathcal{Q}_{\mathcal{B}}$ and $\mathcal{Q}_{\mathcal{R}}$). Thus, we can translate $\mathcal{Q}_{\mathcal{B}}$ upwards until its horizontal ray hits a red point in S' . If no such red point exists, then $\mathcal{Q}_{\mathcal{B}}$ can become a half-plane, say $\mathcal{HP}_{\mathcal{B}}$. As no element of S' can be in $\mathcal{HP}_{\mathcal{B}} \cap \mathcal{Q}_{\mathcal{R}}$, we can then move $\mathcal{Q}_{\mathcal{R}}$ to the right until it no longer intersects $\mathcal{HP}_{\mathcal{B}}$. Then $\mathcal{Q}_{\mathcal{R}}$ can become a half-plane $\mathcal{HP}_{\mathcal{R}}$ which is disjoint with $\mathcal{HP}_{\mathcal{B}}$. Thus, we obtain an optimal solution to the 2-EB-problem for S which is not a corner-type quadrant. Analogously, we can prove that the horizontal ray bounding $\mathcal{Q}_{\mathcal{B}}$ contains a blue point (Figure 4b). \square

For the sake of clarity, we include here a first approximation to our techniques by using a simple method to compute a corner-type solution in $O(n^4)$ time and $O(n^2)$ space. First, observe that the size of the subset of S , say S' , we are seeking will be equal to $|S'| = |\text{Red}(\mathcal{Q}_{\mathcal{R}} \setminus \mathcal{Q}_{\mathcal{B}})| + |\text{Blue}(\mathcal{Q}_{\mathcal{B}} \setminus \mathcal{Q}_{\mathcal{R}})|$. Consider the orthogonal grid G formed by horizontal and vertical lines passing through points of S . Using range search techniques as in [3] we can perform a quadratic time preprocessing on the nodes of G such that for each node $p \in G$ we calculate and store the values $|\text{Red}(\text{SW}(p))|$, $|\text{Blue}(\text{SW}(p))|$, $|\text{Red}(\text{SE}(p))|$, $|\text{Blue}(\text{SE}(p))|$, $|\text{Red}(\text{NW}(p))|$, $|\text{Blue}(\text{NW}(p))|$, $|\text{Red}(\text{NE}(p))|$, and $|\text{Blue}(\text{NE}(p))|$.

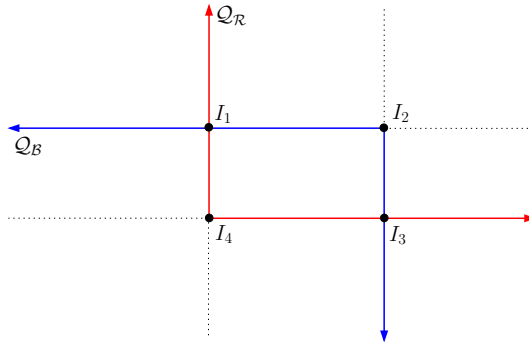


Figure 5: Looking for a corner-type solution.

Let $(\mathcal{Q}_{\mathcal{R}}, \mathcal{Q}_{\mathcal{B}})$ be a corner-type pair of quadrants on G . Denote by I_1, I_2, I_3 , and I_4 the four vertices of the rectangle $\mathcal{Q}_{\mathcal{R}} \cap \mathcal{Q}_{\mathcal{B}}$ as in Figure 5. From the following formulas:

$$|\text{Red}(\mathcal{Q}_{\mathcal{R}} \setminus \mathcal{Q}_{\mathcal{B}})| = |\text{Red}(\text{NE}(I_1))| + |\text{Red}(\text{NE}(I_3))| - |\text{Red}(\text{NE}(I_2))|,$$

$$|\text{Blue}(\mathcal{Q}_{\mathcal{B}} \setminus \mathcal{Q}_{\mathcal{R}})| = |\text{Blue}(\text{SW}(I_1))| + |\text{Blue}(\text{SW}(I_3))| - |\text{Blue}(\text{SW}(I_4))|,$$

it follows that $|S'|$ can be computed in constant time. This yields to an $O(n^4)$ time and $O(n^2)$ space algorithm to solve the 2-EB-problem.

4 The main result

In this section we describe an efficient algorithm which solves the 2-EB-problem for S in $O(n^2 \log n)$ time and $O(n)$ space.

The corner-type solution

First we show how to find a corner-type pair of quadrants $(\mathcal{Q}_{\mathcal{R}}, \mathcal{Q}_{\mathcal{B}})$ that yields an optimal solution to the 2-EB-problem for S .

Let h_p denote the horizontal line passing through a point $p \in S$. We color h_p as follows: if p is red (blue), then color h_p blue (red). By Proposition 3.2 we can assume that the horizontal ray that bounds $\mathcal{Q}_{\mathcal{B}}$ has a red point on it, and that the horizontal ray bounding $\mathcal{Q}_{\mathcal{R}}$ has a blue point. Thus, for each pair (r, b) of red and blue points of S , without loss of generality supposing that the y -coordinate of r is larger than the y -coordinate of b , we solve the following problem.

Let H be the horizontal strip bounded by the lines h_r and h_b . Let ℓ be the vertical line passing through r (Figure 6a). For each red point r' on the right of ℓ and below h_r , let $\mathcal{Q}_{\mathcal{B}}(r')$ be the South-West blue quadrant defined by h_r and the vertical line passing through r' . Similarly, for each blue point b' on the left of ℓ and above h_b , let $\mathcal{Q}_{\mathcal{R}}(b')$ be the North-East red quadrant bounded by h_b and the vertical line passing through b' (Figure 6b).

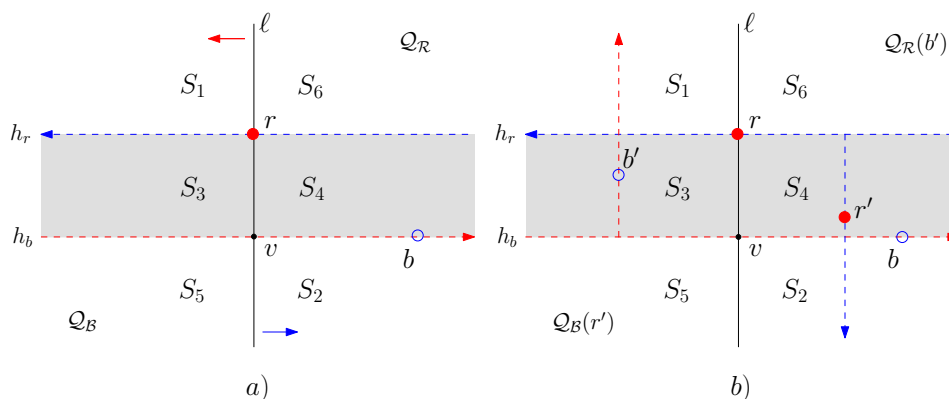


Figure 6: a) Starting position, b) optimal position.

We consider the following problem for a given input H .

The Horizontal Strip problem (HS-problem): *Given H , find $(\mathcal{Q}_{\mathcal{R}}(b'), \mathcal{Q}_{\mathcal{B}}(r'))$ such that: (i) the horizontal ray of $\mathcal{Q}_{\mathcal{B}}(r')$ is contained in h_r , (ii) the horizontal ray of $\mathcal{Q}_{\mathcal{R}}(b')$ is contained in h_b and passes through r , (iii) r belongs to the interior of $\mathcal{Q}_{\mathcal{R}}(b')$, and (iv) $|\text{Red}(\mathcal{Q}_{\mathcal{R}}(b') \setminus \mathcal{Q}_{\mathcal{B}}(r'))| + |\text{Blue}(\mathcal{Q}_{\mathcal{B}}(r') \setminus \mathcal{Q}_{\mathcal{R}}(b'))|$ is maximized.*

By Propositions 3.1 and 3.2, finding an optimal corner-type pair of quadrants for the 2-EB-problem for S can be done by solving the HS-problem for $O(n^2)$ instances, where each instance corresponds to a pair (r, b) of points of S defining a horizontal strip H . Next we show how to solve the HS-problem for the $O(n^2)$ instances in a dynamic way by solving $O(n^2)$ instances of the MCS-problem.

Let v be the intersection point of ℓ and h_b , and consider the quadrants $\text{NE}(v)$ and $\text{SW}(r)$. To solve the HS-problem for H , we slide $\text{NE}(v)$ to the left and $\text{SW}(r)$ to the right, until we reach an optimal solution. To do this task, we assign weights to the points of S as follows: The lines ℓ , h_r and h_b divide the plane into six regions S_1, \dots, S_6 as in Figure 6.

- All the red points in S_1 and all the blue points in S_2 receive weight 1.
- All the blue points in S_3 and all the red points in S_4 receive weight -1 .
- All the remaining points receive weight 0.

Store in r the number of blue points in $\text{SW}(r)$ and in v the number of red points in $\text{NE}(v)$. Project vertically all the red and blue points on h_b . The following result is easy to prove.

Lemma 4.1. *To find the optimal $(\mathcal{Q}_{\mathcal{R}}(b'), \mathcal{Q}_{\mathcal{B}}(r'))$ to the HS-problem for H is equivalent to finding the maximum weight interval I on h_b such that I contains r . Moreover, $|\text{Red}(\mathcal{Q}_{\mathcal{R}}(b') \setminus \mathcal{Q}_{\mathcal{B}}(r'))| + |\text{Blue}(\mathcal{Q}_{\mathcal{B}}(r') \setminus \mathcal{Q}_{\mathcal{R}}(b'))| = |\text{Red}(\text{NE}(v))| + |\text{Blue}(\text{SW}(r))| + w(I)$ where $w(I)$ is the weight of I .*

Suppose that h_r is fixed. We slide h_b down, stopping each time h_b meets a point p of S , and following the rules: (i) if p is a red point in S_2 , p will enter into S_4 and its weight will change to -1 ; (ii) if p is a red point in S_5 , p will enter into S_3 and its weight will remain 0; (iii) if p is blue point and it enters into S_4 , its weight changes to 0; and (iv) if p is blue point and it enters into S_3 , its weight changes to -1 . Each time h_b hits a blue point, we update the number of red points in $\text{NE}(v)$, and recalculate the optimal solution to the HS-problem for the new H (bounded above by h_r and below by the new position of h_b).

This immediately suggests using the dynamic version of the computation of the interval of maximum weight sum on h_b . By using the MCS-tree and Theorem 2.1, the interval of maximum weight sum containing the current r can be computed in $O(\log n)$ time per stop-point of the slide line h_b . Thus, the time cost for the fixed red point r is $O(n) + O(n \log n) = O(n \log n)$.

Now, for each one of the $O(n)$ red points r in S , we set the horizontal line h_r and start again the process by rebuilding the MCS-tree in linear time, and then dynamically solving the MCS-problem. Therefore, the overall time complexity of the algorithm is $O(n)(O(n) + O(n \log n)) = O(n^2 \log n)$.

Theorem 4.1. *An optimal corner-type solution for the 2-EB-problem for S can be found in $O(n^2 \log n)$ time and $O(n)$ space.*

The sandwich-type solution

We now give a brief description of how find a sandwich-type solution. The method is similar to the corner-type solution. The following propositions are given without proof.

Proposition 4.1. *There exists a sandwich-type solution $(\mathcal{R}, \mathcal{B})$ to the 2-EB-problem if and only if there exists a pair of strips $(\mathcal{S}_{\mathcal{R}}, \mathcal{S}_{\mathcal{B}})$, one vertical and the other horizontal, that maximize the sum $|\text{Red}(\mathcal{S}_{\mathcal{R}} \setminus \mathcal{S}_{\mathcal{B}})| + |\text{Blue}(\mathcal{S}_{\mathcal{B}} \setminus \mathcal{S}_{\mathcal{R}})|$.*

Proposition 4.2. *There exists a pair of strips $(\mathcal{S}_{\mathcal{R}}, \mathcal{S}_{\mathcal{B}})$, $\mathcal{S}_{\mathcal{R}}$ vertical and $\mathcal{S}_{\mathcal{B}}$ horizontal, that maximize the sum $|\text{Red}(\mathcal{S}_{\mathcal{R}} \setminus \mathcal{S}_{\mathcal{B}})| + |\text{Blue}(\mathcal{S}_{\mathcal{B}} \setminus \mathcal{S}_{\mathcal{R}})|$ such that there is a red element of S on the top side of the rectangle determined by their intersection (see Figure 7b).*

Therefore, we can proceed in a similar way to the corner-type solution. For each pair of blue lines h_{r_1} and h_{r_2} (Figure 7a), we consider the blue strip $\mathcal{S}_{\mathcal{B}}$ bounded by them, plus a starting red strip $\mathcal{S}_{\mathcal{R}}$ consisting of the vertical red line ℓ passing through a red point r_1 in h_{r_1} (Figure 7a). They form an initial candidate sandwich-type solution with value $|\text{Blue}(\mathcal{S}_{\mathcal{B}})|$. As we widen $\mathcal{S}_{\mathcal{R}}$ by translating two vertical lines v_1 and v_2 to the left and right of r_1 , respectively, the value of the solution will change according to the following weight rules applied to the points of S , where regions S_1 , S_2 , and S_3 are as in Figure 7b:

- Blue points in S_1 , S_2 , and S_3 receive weight 0, -1 , and 0, respectively.
- Red points in S_1 , S_2 , and S_3 receive weight 1, 0, and 1, respectively.

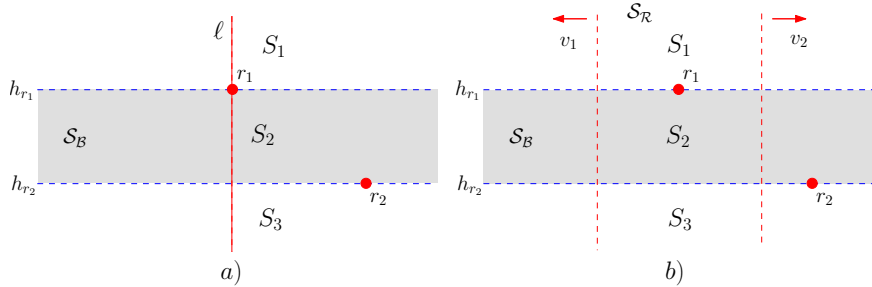


Figure 7: a) Starting position, b) finding an optimal position.

As we did for the corner-type solution, we now use a MCS-tree to obtain the optimal positions of v_1 and v_2 , and then maintain dynamically the MCS-tree as we slide h_{r_2} down using Theorem 2.1.

The disjoint-type solution. Computing a disjoint-type solution is straightforward. It can be solved in $O(n \log n)$ time by using a simple sweep. We omit the details.

Therefore, putting together the results above, we get to the following theorem.

Theorem 4.2. *The 2-EB-problem for S can be solved in $O(n^2 \log n)$ time and $O(n)$ space.*

5 The three chromatic planar case with three disjoint boxes

In this section we study the following problem as an extension of the 2-EB-problem.

The Disjoint Three-Chromatic Enclosing Boxes problem (DTEB-problem): *Let S be a set of n points on the plane in general position such that the points are colored blue, red, and green. The DTEB-problem for S consist of finding three pairwise-disjoint isothetic rectangles \mathcal{B} , \mathcal{R} , and \mathcal{G} such that $|\text{Blue}(\mathcal{B})| + |\text{Red}(\mathcal{R})| + |\text{Green}(\mathcal{G})|$ is maximum.*

Theorem 5.1. *The DTEB-problem for S can be solved in $O(n \log n)$ time and $O(n)$ space.*

Proof. First, we observe that given any three pairwise-disjoint isothetic rectangles \mathcal{B} , \mathcal{R} , and \mathcal{G} we can always find two isothetic lines (or a line and a half-line) ℓ_1 and ℓ_2 such that for any pair of elements of $\{\mathcal{B}, \mathcal{R}, \mathcal{G}\}$ either ℓ_1 or ℓ_2 separates them (Figure 8). We solve the two cases separately.

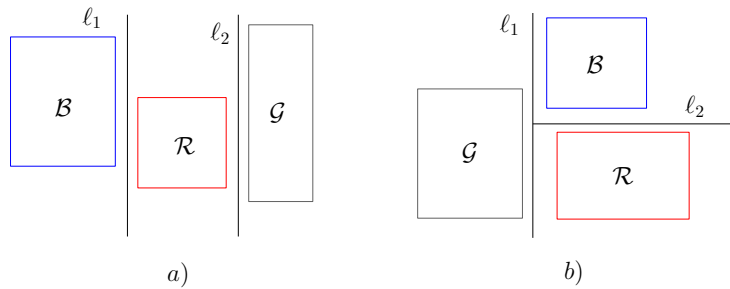


Figure 8: Separation by two isothetic lines: a) parallel lines, b) perpendicular lines.

Parallel case: We show first how to solve the parallel case by reducing it to the *Longest Increasing Subsequence problem* [12]: *Given a sequence (x_1, \dots, x_n) of numbers, find a largest subset of indexes $\alpha_1 < \dots < \alpha_k$ such that $x_{\alpha_1} \leq \dots \leq x_{\alpha_k}$.* It is well known that this problem can be solved in $O(n \log n)$ time. In fact, the instance we have to solve here can be solved in linear time, as it involves a sequence whose elements have values 1, 2, or 3. Suppose first that ℓ_1 and ℓ_2 are vertical and that \mathcal{B} , \mathcal{R} , and \mathcal{G} form an optimal solution. We have to consider six cases for the relative positions of \mathcal{B} , \mathcal{R} , and \mathcal{G} with respect to ℓ_1 and ℓ_2 . Suppose that \mathcal{B} is to the left of ℓ_1 , \mathcal{R} lies between ℓ_1 and ℓ_2 , and \mathcal{G} is to the right of ℓ_2 (Figure 8a). The remaining five cases are solved in a similar way.

Assign weight 1 to the points colored blue, weight 2 to those colored red, and weight 3 to the green points. Project all the points in S on the x -axis obtaining a sequence Σ of 1's, 2's, and 3's. Observe that all the blue, red, and green points contained in \mathcal{B} , \mathcal{R} , and \mathcal{G} , respectively, as projected into the x -axis induce an increasing subsequence of Σ . The result follows.

Perpendicular case: Suppose, without loss of generality, that the relative positions of ℓ_1 , ℓ_2 , \mathcal{B} , \mathcal{R} , and \mathcal{G} are as in Figure 8b. We show how to solve this case using dynamic binary trees.

Suppose first that the elements of S are labelled p_1, \dots, p_n such that the y -coordinate of p_i is smaller than the y -coordinate of p_j , $i < j$. Construct a balanced binary tree T such that its set of leaves $S = \{p_1, \dots, p_n\}$ are colored red, blue, and black. Given an index i , $1 \leq i \leq n$, let $R(i)$ (resp. $B(i)$) be the number of red elements p_j with $j \leq i$ (resp. blue p_j 's with $j \geq i$). Our objective is to store information on the vertices of T such that the following problem, which we call the *Maximum Sum problem*, or the MS-problem for short, can be solved dynamically in $O(\log n)$ time:

The Maximum Sum problem (MS-problem): *Find an index i that maximizes $R(i) + B(i)$. At each point in time, a red or blue point can change color to black.*

As in Section 2.1, for every internal node p of T let $I(p) = [p_{l_p}, p_{r_p}]$ be the interval of S formed by the descendants of p in T . If $p = p_j$ for some j , $I(p) = [p_j]$. For an index i , $l_p \leq i \leq r_p$, let $R_p(i)$ (resp. $B_p(i)$) be the number of red p_j 's such that $l_p \leq j \leq i$ (resp. the number of blue p_k 's with $i \leq k \leq r_p$). We define $I_R(p)$ ($I_B(p)$) to be the number of red (blue) points in $I(p)$.

At every node p of T we will store the following information: an index i_p , $l_p \leq i_p \leq r_p$, such that $R_p(i_p) + B_p(i_p)$ is maximized. If p is a leaf p_j of T , then $i_p = j$.

If w is an internal node of T whose left and right children are p and q , respectively, it is easy to see that i_w is either i_p or i_q , according to the following criterion: If $R(i_p) + B(i_p) + I_B(q) \geq I_R(p) + R(i_q) + B(i_q)$ then $i_w = i_p$, otherwise $i_w = i_q$.

It follows immediately that using a bottom-up traversal of T , we can calculate the values $I_R(p)$, $I_B(p)$, i_p , $R(i_p)$, and $B(i_p)$ for all nodes of T in linear time by Theorem 2.1. Moreover, by Theorem 2.1, it is straightforward to see that if a red or blue point p_i of T is re-colored black, then we can update T in $O(\log n)$ time by traversing the path from p_i to the root of T , and that if $Root$ is the root vertex of T , $R(i_{Root}) + B(i_{Root})$ is the solution to the MS-problem.

We are now ready to solve the perpendicular case. Take a copy S' of S and re-color black to all the green elements of S' . Construct a binary tree T as described above such that the elements of S' are the leaves of T . $R(i_{Root}) + B(i_{Root})$ is the optimal solution for which the green box contains no points.

We now perform a line sweep using a vertical line ℓ_1 from left to right. Initially all the elements of S are to the right of ℓ_1 , at the end all the elements of S are to the left of ℓ_1 . Each time ℓ_1 meets a point in S we change the color of its corresponding copy in S' to black. In $O(\log n)$ time, we update T , and recalculate the boxes \mathcal{B} , \mathcal{R} that maximize the number of blue points plus the

number of red points contained in them. Each time ℓ_1 meets a green point, the number of elements in \mathcal{G} increases by one. By keeping the maximum number of red point plus blue points plus green points contained in \mathcal{R} , \mathcal{B} , and \mathcal{G} , respectively, we obtain the optimal solution to our problem. \square

6 Generalization and other applications

We have shown how to solve the 2-EB-problem and the DTEB-problem by using dynamic trees over a sequence of elements for which some attribute is dynamically maintained. Notice that in the case of the 2-EB-problem (resp. the DTEB-problem) the interval of maximum weight (resp. the position in which the number of red elements to its left plus the number of blue points to its right is maximum) is maintained in $O(\log n)$ time. The approach can easily be generalized as follows.

Generalization: *Let $X = (x_1, x_2, \dots, x_n)$ be a sequence of n elements and let $\mathcal{A}(X)$ be a set of attributes of X that depends on its elements. Suppose that $\mathcal{A}(X)$ can be obtained by applying a recursive $O(n)$ -time algorithm as follows: if the length of X is at most one compute $\mathcal{A}(X)$ in constant time, otherwise $\mathcal{A}(X)$ is computed in constant time from $\mathcal{A}(X_1)$ and $\mathcal{A}(X_2)$, where X_1 and X_2 are the two halves of X . Then the recursive tree having the elements of X as its leaves is a balanced binary tree and representing it we can, whenever some x_i changes, recompute $\mathcal{A}(X)$ in $O(\log n)$ time by traversing the path from x_i to the root.*

Note that although the goal may be to maintain only one attribute, we maintain a set of them because in many applications the calculation of an attribute of the sequence depends on others. For the sake of clarity see the MCS-tree (Section 2.1), where the property of the sequence is the weight of its elements and other three attributes are considered in order to maintain the interval of maximum weight.

By applying this generalization we are now ready to present efficient algorithms for a collection of problems.

The Maximum Weighted Box problem (MWB-problem): *Given a set S of n points on the plane and a weight function $w : S \rightarrow \mathbb{R}$, compute the axis-aligned box H such that $\sum_{x \in H \cap S} w(x)$ is maximized.*

Note that there exists a box H that gives an optimal solution such that H contains on its boundary only elements of S with positive weight. A solution can be calculated as follows:

Make a top-bottom sweep of the elements of S with a horizontal line ℓ_1 and whenever it stops at a positive element of S , make a sweeping of the elements of S that lie below ℓ_1 with another horizontal line ℓ_2 that starts at ℓ_1 and stops only on positive-weight points. Let X be the points of S ordered by abscissa and let w' be a weight function for each $x \in S$ defined as follows:

$$w'(x) = \begin{cases} w(x) & \text{if } x \text{ lies between } \ell_1 \text{ and } \ell_2; \\ 0 & \text{if } x \text{ lies either above } \ell_1 \text{ or below } \ell_2. \end{cases}$$

For a given position of ℓ_1 and ℓ_2 , the optimal box whose top and bottom sides lie on ℓ_1 and ℓ_2 respectively is determined by the interval of maximum weight sum on X . This interval is dynamically computed in $O(\log n)$ time by using an MCS-tree. We obtain a simple $O(n^2 \log n)$ -time algorithm since there are $O(n^2)$ possible positions of ℓ_1 and ℓ_2 .

The Maximum Box problem (MB-problem): *Given a set of blue points B and a set of red points R on the plane, where $|R \cup B| = n$, find an axis-aligned box H such that $H \cap R = \emptyset$ and $|H \cap B|$ is maximized.*

The MB-problem can be solved in $O(n^2 \log n)$ time by using $O(n)$ space since it is an instance of the MWB-problem by considering $S = B \cup R$ and the weight function:

$$w(x) = \begin{cases} 1 & \text{if } x \in B; \\ -\infty & \text{if } x \in R. \end{cases}$$

The MB-problem was solved in [13] with $O(b^2 \log b + br + r \log r)$ time, where $r = |R|$ and $b = |B|$. However our approach is a simpler method with the same complexity in the worst case.

The Maximum Bichromatic Discrepancy Box problem (MBDB-problem): *Given a set of blue points B and a set of red points R on the plane, where $|R \cup B| = n$, find an axis-aligned box H such that $||H \cap B| - |H \cap R||$ is maximized.*

The solution to the MBDB-problem can be obtained by solving the following two instances of the MWB-problem, in both consider that $S = R \cup B$:

$$(1) \quad w(x) = \begin{cases} 1 & \text{if } x \in B, \\ -1 & \text{if } x \in R; \end{cases} \quad (2) \quad w(x) = \begin{cases} -1 & \text{if } x \in B, \\ 1 & \text{if } x \in R. \end{cases}$$

In this way, the MBDB-problem can be solved in $O(n^2 \log n)$ time by using $O(n)$ space. This complexity matches the result given in [7].

The Weak Strip Separation problem (WSS-problem): *Let S be a set of n points on the plane in general position such that its elements are colored red or blue. Find a corridor C bounded by two parallel lines in any direction such that the number of blue points inside C plus the number of red points outside C is maximized.*

Suppose that we have a direction given by a line ℓ and we want to compute the best corridor C_ℓ that is orthogonal to ℓ . It can be done as follows: Project the points of S onto ℓ , obtaining the sequence of elements $X = (x_1, \dots, x_n)$ ordered from left to right. Given an index i , $1 \leq i \leq n$, let $R^-(i)$ (resp. $R^+(i)$) be the number of red elements x_k with $k \leq i$ (resp. $k \geq i$) and given two indexes i and j , $1 \leq i \leq j \leq n$, let $B(i, j)$ be the number of blue elements x_k with $i \leq k \leq j$. It is easy to see that C_ℓ is determined by the indexes i and j , $1 \leq i \leq j \leq n$, such that $R^-(i) + B(i, j) + R^+(j)$ is maximum.

If we rotate the line ℓ about the origin, the order of the projected points on ℓ changes a quadratic number of times. Thus we make a rotational sweeping passing from the current critical direction to the next by swapping two consecutive elements of X . When the swap occurs, the solution is dynamically computed using a tree constructed by using similar arguments as in Section 5 and also applying the general approach. This method has $O(n^2 \log n)$ -time and $O(n^2)$ -space complexities.

The Weak Cross Separation problem (WCS-problem): *Let S be a point set on the plane in general position such that its elements are colored red or blue. Find a point p on the plane such that $|Blue(NE(p))| + |Red(NW(p))| + |Blue(SW(p))| + |Red(SE(p))|$ is maximized.*

The point p can be seen as the intersection point between a horizontal line ℓ_h and a vertical line ℓ_v defining a cross. Suppose we are given ℓ_h and we want to compute the best location of ℓ_v . It can be done as follows: Project the points of S onto ℓ_h , obtaining the sequence of elements $X = (x_1, \dots, x_n)$ ordered from left to right. Given an index i , $1 \leq i \leq n$, let $R_a(i)$ (resp. $R_b(i)$) be the number of red elements x_k with $k \leq i$ (resp. $k \geq i$) such that its corresponding point in S is above (resp. below) ℓ_h , and let $B_a(i)$ (resp. $B_b(i)$) be the number of blue elements x_k with $k \geq i$ (resp. $k \leq i$) such that its corresponding point in S is above (resp. below) ℓ_h . Note that the best position of ℓ_v is determined by the index i such that $R_a(i) + B_a(i) + B_b(i) + R_b(i)$ is maximum.

We make a top-bottom sweep with ℓ_h dynamically computing the best position of ℓ_v at each step. This fits the general approach and we obtain an $O(n \log n)$ -time and $O(n)$ -space algorithm.

7 Conclusions

In this paper we have shown the connection between maximum boxes problems and Bentley’s maximum consecutive sum problem. We have developed a dynamic data structure that allows us to maintain the solution of the Bentley’s maximum consecutive sum problem in $O(\log n)$ time when an element of the sequence changes its value. This data structure can be computed in linear time. The key idea used to dynamically solve Bentley’s maximum consecutive sum problem was extended as a general technique useful for solving other data analysis problems.

A natural problem for further research is the three-dimensional case. Some cases of the 2-EB-problem in \mathbb{R}^3 can be reduced to $O(n^2)$ instances of the 2-EB-problem in \mathbb{R}^2 by using projections of the bicolored point set on the plane. (see Figure 9 for a corner-type solution). Thus we could solve the problem in $O(n^4 \log n)$ time and $O(n)$ space. It would be interesting to solve the other cases and to improve this complexity as well. Finally, it is worthy to know if the 2-EB-problem in \mathbb{R}^2 is 3SUM-hard.

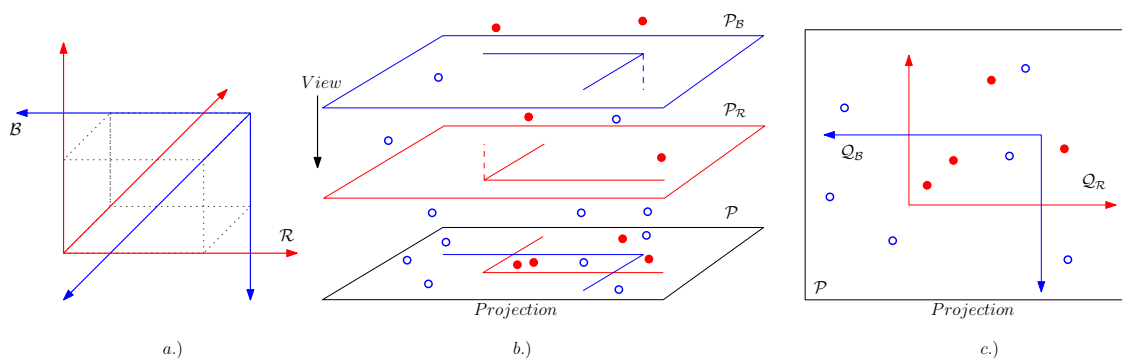


Figure 9: A corner-type solution in \mathbb{R}^3 is projected to \mathbb{R}^2 .

References

- [1] B. Aronov and S. Har-Peled. On approximating the depth and related problems. *SIAM Journal on Computing*, 38(3), , pp. 899-921, 2008.
- [2] J. L. Bentley. Programming pearls: algorithm design techniques. *Comm. ACM* 27 (1984) 865–873.
- [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*, (1977), pp. 193–201.
- [4] S. Bespamyatnikh and M. Segal. Covering a set of points by two axis-parallel boxes. *Information Processing Letters* 75 (2000) 95–100.
- [5] C. Cortés, J. M. Díaz-Báñez, and J. Urrutia. Finding enclosing boxes with empty intersection. *23rd. European Workshop on Computational Geometry*, 2006, pp. 185–188.
- [6] K.-Y. Chen and K.-M. Chao. On the range maximum-sum segment query problem. *R. Fleischer and G. Trippen Eds., ISAAC 2004, LNCS 3341*, 2004, pp. 294–305.

- [7] D. P. Dobkin, D. Gunopulos, and W. Maass. Computing the maximum bichromatic discrepancy, with applications to computer graphics and machine learning. *J. Computer and Systems Sciences*, 52(3) (1996) 453–470.
- [8] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley and Sons, Inc., 2001.
- [9] J. Eckstein, P. L. Hammer, Y. Liu, M. Nediak, and B. Simeone. The maximum box problem and its applications to data analysis. *Comput. Optim. Appl.* 23 (2002) 285–298.
- [10] P. L. Hammer and T. Bonates. Logical analysis of data: from combinatorial optimization to medical applicatios. *Rutcor Research Report, RRR 10-2005*. Rutgers University, New Jersey, USA, 2005.
- [11] H. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. The MIT Press, 2001.
- [12] D. E. Knuth. *Sorting and searching. The Art of Computer Programming*. Addison-Wesley, 2000.
- [13] Y. Liu and M. Nediak. Planar case of the maximum box and related problems. In Proc. 15th Canad. Conf. Comp. Geom., Halifax, Nova Scotia, 2003.