Planar Case of the Maximum Box and Related Problems*

Ying Liu† Mikhail Nediak‡

Abstract

Given two finite sets of points X^+ and X^- in $\mathbb{R}^d$, the maximum box problem is the problem of finding a box (hyperrectangle) $B = \{\mathbf{x} : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$ such that $B \cap X^- = \emptyset$, and the cardinality of $B \cap X^+$ is maximized. The *maximum bichromatic discrepancy problem* is to find a box B maximizing the difference between the number of the points of X^+ and X^- inside the box, i.e. $\max ||B \cap X^+| - |B \cap X^-||$.

In this paper, we discuss an exact algorithm for the maximum box problem on the plane. In addition, we provide factor 2 approximation algorithms for planar cases of both problems and give an extension to the numerical discrepancy problem.

1 Introduction

The maximum box problem is an interesting discrete optimization problem with application in data analysis (see [7]). Given vectors $\mathbf{l} = (l_1, \ldots, l_d)$ and $\mathbf{u} = (u_1, \ldots, u_d)$ in $\mathbb{R}^d$, such that $l_i \leq u_i$ for $i = 1, \ldots, d$, we define a *(closed) box* or *hyperrectangle* $[\mathbf{l}, \mathbf{u}]$ to be the set

$$[l_1, u_1] \times [l_2, u_2] \times \ldots \times [l_d, u_d].$$

(or, equivalently, $[\mathbf{l}, \mathbf{u}] = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$). For a finite set X in $\mathbb{R}^d$, we denote its componentwise minimum and maximum as $\min X$ and $\max X$, respectively. We say that a box B is *spanned* by the set X if $B = [\min X, \max X]$.

Suppose we are given two finite disjoint sets $X^+, X^- \subseteq \mathbb{R}^d$, which will be referred to as *positive* and *negative*, respectively. Let us denote $|X^+| = r, |X^-| = s$. A box is called *homogeneous* if it contains no negative points. We define the *size* of a box as the cardinality of its intersection with X^+. The *maximum box problem* then consists in finding a homogeneous box of the maximum size. Note that the problem is equivalent to that of finding a maximum spanned homogeneous box.

A decision version of the maximum box problem, when dimension of the space d is not bounded, was proved to be NP-complete by a polynomial reduction of the independent set problem to it [7]. In this paper, we shall provide an algorithm with running time $O(r^2 \log r + rs + s \log s)$ for the maximum box problem on the plane ($d = 2$).

The maximum box problem is related to the maximum empty rectangle problem (MER) on the plane: find a rectangle of the maximum area (Euclidean measure $\mu(B)$) which does not contain any point of a given set X^-, and which is completely inside some rectangle containing X^- (typically, a rectangle spanned by X^-). The classical MER formulation presented in [9] considers only the rectangles whose sides are aligned with the coordinate axes (isothetic) which exactly matches our definition of a box. The algorithms running in $O(s \log^3 s)$ and $O(s \log^2 s)$ time in the worst case were presented in [2, 6]. Examples of more recent work on MER problem include removing the isotheticity requirement [1] and its extension to $\mathbb{R}^3$ [3]. Note that if, given a box B, we define its discrete measure $\mu_{X^+}(B)$ as $|B \cap X^+|/|X^+|$ then the maximum box problem can be viewed as the problem of finding an empty (with respect to X^-) rectangle B that maximizes a discrete measure $\mu_{X^+}(B)$.

We shall also consider the maximum bichromatic and numerical discrepancy problems in the plane which have important applications in computational learning theory, computational geometry and computer graphics (see [4, 8]). Given finite sets of points X^+, X^- in $\mathbb{R}^2$, the *bichromatic discrepancy* of a box B is the difference between the number of the positive and the negative points it contains, i.e. $||X^+ \cap B| - |X^- \cap B||$. The *maximum bichromatic discrepancy problem* is to find a box that maximizes the bichromatic discrepancy. Let now $\mathcal{F}$ be the set of all possible boxes inside $[0,1]^2$ and X be a finite set in $\mathbb{R}^2$. For any box B in $\mathcal{F}$, its *numerical discrepancy* with respect to X is defined as $|\mu(B) - \mu_X(B)|$. The *maximum numerical discrepancy problem* is to find a box that maximizes the numerical discrepancy.

Because $\mu(B)$ is the probability that the points fall inside the box B under uniform distribution hypothesis, and $\mu_X(B)$ is the empirical probability of points inside the box B, the numerical discrepancy of B can be used as a measure of the deviation of the empirical distribution from uniform. In computational learning theory, the problem of agnostic PAC-learning with simple geometric hypotheses can be reduced to the problem of computing the maximum bichromatic discrep-

*First author greatfully acknowledges a partial support of DIMACS. Authors would also like to thank Mario Szegedy of Rutgers University for valuable comments and suggestions.

†RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway, NJ 08854, USA. E-mail: yingliu@rutcor.rutgers.edu

‡Department of Mathematics and Statistics, McMaster University, 1280 Main Street West, Hamilton, ON, L8S 1A1, Canada. E-mail: nediakm@math.mcmaster.ca

ancy for simple geometric ranges. In computational geometry, efficient computation of the discrepancy of a two-colored point set is useful for the construction of ϵ-approximations of point sets. Finally, in computer graphics, the maximum numerical discrepancy of a point set is a good measure on how well a sampling pattern captures details in a picture [5].

In [5], the algorithms that compute the maximum bichromatic discrepancy in $\mathbb{R}^2$ in $O((r+s)^2 \log(r+s))$ time, and the maximum numerical discrepancy in $O(|X|^2 \log^2 |X|)$ time were given. The paper also presented the following results: first, in $\mathbb{R}^2$, an approximation of the maximum bichromatic discrepancy D such that $|D - D_{opt}| = O((r+s)/u)$, where D_{opt} is the optimum, can be computed in $O((r+s)\log(r+s+u^4 \log u)$ time (for $u = o(\sqrt{r+s})$). Second, given a point set $X \subset [0,1]^2$, an approximation of the maximum numerical discrepancy D such that $|D - D_{opt}| = O(1/u)$, where D_{opt} is the optimum, can be computed in $O(|X|\log|X| + u^4 \log^2 u)$ time (for $u = o(\sqrt{|X|})$). These results did not necessarily provide a guaranteed performance ratio. It was posed as an open problem in [5] to find better approximation algorithms.

We present factor 2 approximation algorithms in a plane with $O((r+s)\log^2(r+s))$ running time for the maximum box and maximum bichromatic discrepancy problems. We also present a factor 2 approximation algorithm in a plane with $O(|X|\log^3|X|)$ running time for the maximum numerical discrepancy problem.

2 An exact algorithm for the maximum box problem in $\mathbb{R}^2$

We will refer to an instance of the planar maximum box problem as *non-degenerate* if

$$\forall \mathbf{p} = (p_x, p_y),\ \mathbf{q} = (q_x, q_y) \in X^+ \cup X^- : p_x \neq q_x. \quad (1)$$

The main idea of our algorithm is the following. Suppose that we have a subroutine that takes a positive point $\mathbf{p}$ and the list of points to the right of $\mathbf{p}$ as its input, and computes a maximum homogeneous spanned box with $\mathbf{p}$ in its left boundary. Since a homogeneous spanned box must always have some positive point in its left boundary and there exists a maximum homogeneous box which is spanned, we can solve a problem by calling this subroutine for every positive point and selecting the maximum of all calls.

Let $\mathcal{Q}$ be a list of points in $X^+ \cup X^-$ in the increasing order of their x-coordinates. Note that due to the non-degeneracy assumption (1) the ordering is unique. For any $\mathbf{p} \in X^+$ we also define a similarly ordered list $\mathcal{Q}_\mathbf{p} = \{\mathbf{q} \in \mathcal{Q} : q_x \geq p_x\}$. We need not construct $\mathcal{Q}_\mathbf{p}$ explicitly as it is specified by the position of $\mathbf{p}$ in $\mathcal{Q}$.

The following algorithm for finding a maximum box uses a line sweep subroutine `Sweep`($\mathbf{p}$, $\mathcal{Q}_\mathbf{p}$) which returns a maximum box with $\mathbf{p}$ in its left boundary and box's size $s_\mathbf{p}$.

Algorithm 1 `MaxBox2`(X^+, X^-)
Input: Sets of positive and negative points X^+ and X^-.
Output: The spanned maximum box $B_{\max}$ and its size $s_{\max}$.

$B_{\max} := \emptyset$; $s_{\max} := 0$;
`construct` $\mathcal{Q}$;
for all `positive` $\mathbf{p} \in \mathcal{Q}$
  ($B_\mathbf{p}$, $s_\mathbf{p}$) = `Sweep`($\mathbf{p}$, $\mathcal{Q}_\mathbf{p}$);
  if $s_\mathbf{p} > s_{\max}$ **then**
    $B_{\max} := B$; $s_{\max} := s_\mathbf{p}$;
  endif
endfor
return ($B_{\max}$, $s_{\max}$);

The data structures used by `Sweep`($\mathbf{p}$, $\mathcal{Q}_\mathbf{p}$) are as follows. Let $x = p_x$ and $x = x'$, $x' \geq p_x$ be the initial and the current positions of the sweep line.

1. The algorithm maintains an open interval $I(x') = (y(x'),\ y'(x'))$ such that

$$y(x') = \max\left\{ q_y : \begin{array}{c} (q_x, q_y) \in X^-, \\ p_x \leq q_x \leq x',\ q_y \leq p_y \end{array} \right\} \quad (2)$$

$$y'(x') = \min\left\{ q_y : \begin{array}{c} (q_x, q_y) \in X^-, \\ p_x \leq q_x \leq x',\ q_y \geq p_y \end{array} \right\} \quad (3)$$

   The value $y(x')$ is the lower bound on the bottom boundary of a homogeneous box that contains $\mathbf{p}$ in its left boundary while its right boundary is at x'. Similarly, $y'(x')$ is the upper bound on the top boundary. If $I(x') = \emptyset$ the algorithms stops.

2. We shall use variables $B_\mathbf{p}$ and $s_\mathbf{p}$ to record the maximum box and its size over all boxes found while a position of the sweep line changed from p_x to x'.

3. Balanced binary search tree $\mathcal{P}(x')$ will contain all positive points between $x = p_x$ and $x = x'$ that can be potentially included in a homogeneous box in the increasing order of their y-coordinates. For each node ν of $\mathcal{P}(x')$ we will also maintain a counter of the number of positive points on the subtree rooted at ν. The tree $\mathcal{P}(\cdot)$ can be implemented by any of the standard techniques (such as red-black or splay trees).

While a position of the sweep line shifts to the right, the following events can happen:

- A positive point $\mathbf{q}$ with $q_y \in I(q_x)$ increases the size of the box with $\mathbf{p}$ in its left boundary and should be added to $\mathcal{P}(\cdot)$.
- A negative point $\mathbf{q}$ changes $I(\cdot)$ at q_x (that is, for some small ϵ, we have $q_y \in I(q_x - \epsilon)$).

If a negative or positive point does not belong to either of these categories, it can be disregarded.

The following algorithm is built around processing these events. For convenience of notation, we suppress argument of $I(\cdot)$ and $\mathcal{P}(\cdot)$ with an understanding that it always corresponds to the x-coordinate of a point currently being processed. We will also use shorthand notation "$I \cap \mathcal{P}$" to denote the set of points on the current $\mathcal{P}$ whose y-coordinates fall into the current I.

Algorithm 2 `Sweep`($\mathbf{p}$, $\mathcal{Q}_\mathbf{p}$)
Input: A positive point $\mathbf{p}$, queue of events $\mathcal{Q}_\mathbf{p}$.
Output: A maximum spanned homogeneous box with $\mathbf{p}$ on its left boundary and its size.

```
I := (−∞, +∞); P = ∅
B_p := ∅; s_p := 0;
i := Q_p.begin();
while i ≠ Q_p.end() and I ≠ ∅
  save Q_p[i] to q = (q_x, q_y);
  if q_y ∈ I then
    if q is positive then
      add q to P;
      s :=number of points in I ∩ P;
      if s > s_p then
        y_p :=smallest y-coordinate in I ∩ P;
        y'_p :=largest y-coordinate in I ∩ P;
        s_p := s;
        B_p := [p_x, q_x] × [y_p, y'_p];
      endif
    else
      if q_y > p_y then I := (y, q_y);
      else if q_y < p_y then I := (q_y, y');
      else I := ∅;
      endif
    endif
  endif
  i := i + 1;
endwhile
return (B_p, s_p);
```

Proposition 1 *The algorithm* `Sweep`($\mathbf{p}$, $\mathcal{Q}_\mathbf{p}$) *correctly finds a maximum box such that its left boundary contains* $\mathbf{p}$ *in* $O(r \log r + s)$ *time.*

Proof. First, observe that $\mathbf{p}$ and $\mathbf{q}$, $q_x \geq p_x$ can be inside the same homogeneous box if and only if $q_y \in I(q_x)$, which is defined by (2)-(3). Second, observe that s is the maximum size of any box with positive $\mathbf{q}$ satisfying $q_y \in I(q_x)$ in its right boundary. Third, the algorithm finds maximum $s_\mathbf{p}$ of s over the all positive $\mathbf{q}$ satisfying this condition. Fourth, it is immediate to see that $I(\cdot)$ is correctly computed over the execution of the algorithm. Correctness follows.

Handling a negative event point, takes $O(1)$ time. Handling a positive event point takes $O(\log |\mathcal{P}|)$ time due to operations on the balanced binary search tree $\mathcal{P}$. Since $|\mathcal{P}| = O(r)$, the complexity follows. □

Corollary 1 *The algorithm* `MaxBox2`(X^+, X^-) *correctly finds a maximum box and runs in* $O(r^2 \log r + rs + s \log s)$ *time.*

Proof. The correctness follwos from the fact that the maximum box problem is equivalent to finding a maximum spanned box which, in turn, must have at least one positive point on its left boundary.

Time spent in computing the next iteration of the `for`-loop is $O(r + s)$. Construction of $\mathcal{Q}$ takes $O((r + s)\log(r + s))$ time. Since $\log(r + s) = \Theta(\log r + \log s)$, construction of $\mathcal{Q}$ is $O(r \log r + s \log r + r \log s + s \log s)$. Together with the $O(r^2 \log r + rs)$ complexity of r calls to the `Sweep` subroutine (from Proposition 1), the total is $O(r^2 \log r + rs + s \log s)$. □

Remark 1 The non-degeneracy assumption (1) can be relaxed if we assume that negative points always have a priority over positive ones in $\mathcal{Q}$. The complexity stays the same, while the correctness proof requires a slightly more complicated analysis.

Remark 2 The algorithm `MaxBox2`(X^+, X^-) can be used as a subroutine to find a maximum box in $\mathbb{R}^d$ in $O(r^{2d-4}(r^2 \log r + rs + s \log s))$ time.

Remark 3 It is straightforward to modify the algorithm to solve a version of the problem (see [7]) with general positive weights on the points in X^+. In light of our discussion of relation to MER problem, we observe that this will solve a problem of finding a rectangle, empty with respect to X^- that maximizes an arbitrary discrete measure whose support is X^+.

Remark 4 Finally, the subroutine `Sweep`$(\mathbf{p}, \mathcal{Q}_\mathbf{p})$ can be used to construct an algorithm with running time $O(r^2 \log r + rs + s \log s)$ for finding a pair of disjoint homogeneous boxes in $\mathbb{R}^2$ that maximizes the sum of their sizes (*two box problem*, see [5]).

3 An approximation algorithm for the maximum box problem in $\mathbb{R}^2$

While a notion of an approximation algorithm typically arises in the context of problems for which polynomial algorithm is not known, it may also make sense in any application where the exact algorithm would be prohibitatively slow. In data classification applications, the number of points of either kind (positive or negative) can be extremely large. Thus, the $r^2 \log r$ component in the complexity of the algorithm in the previous section may make it useless. Moreover, an approximate answer may be quite satisfactory in many heuristic approaches for classifier construction.

First, we describe a generalized version of the line sweep subroutine presented in the previous section. The generalized version `GeneralSweep`$(x', \mathcal{Q}')$ takes as an input the initial position of the sweep line x' and the queue of points $\mathcal{Q}'$ to the left or right from x'. We assume that points on $\mathcal{Q}'$ can be traversed in both decreasing and increasing order of their x-coordinates but in either case negative points have a priority over positive. The purpose of `GeneralSweep`$(x',\ \mathcal{Q}')$ is to find a maximum box with its left or right vertical boundary at x'. The main idea of the `Sweep` was to look at a single interval I of y-coordinates that would bound any homogeneous box containing a particular positive point in its left boundary. Here, we look at all such potential intervals by fixing a value of the x-coordinate instead of a positive point. The generalized version will use additional data structures:

1. Binary search tree $\mathcal{L}$ will contain disjoint open intervals formed by the consecutive y-coordinates of the negative points between the lines $x = x'$ and $x = q_x$. Since these intervals are disjoint, they can be ordered by their left end-points.

2. Each interval $I \in \mathcal{L}$ together with $[p_x, q_x]$ determines a box $[p_x, q_x] \times I$. We shall use a binary search tree $\mathcal{S}$ to record the sizes of the boxes corresponding to the intervals in BST $\mathcal{L}$. Also, each interval record in $\mathcal{L}$ will keep a pointer to its size record in $\mathcal{S}$ and vice versa.

Algorithm 3 `GeneralSweep`$(x',\ \mathcal{Q}')$
Input: Initial position of the sweep line $x = x'$, queue of events $\mathcal{Q}'$ to the left or right from x'.
Output: A maximum box with a vertical boundary at x' and its size.

```
L := {(−∞, +∞)};
S := {0};
P := ∅;
B_x' := {0} × {(−∞, +∞)};  s_x' := 0;
i := Q'.begin();
while i ≠ Q'.end()
  save Q'[i] to q;
  if ∃I ∈ L such that q_y ∈ I then
    if q is positive then
      add q to P;
      update size of I in S;
      s := maximum of S;
      if s > s_x' then
        s_x' := s;  B_x' := [x', q_x] × I;
      endif
    else
      delete I = (y, y') from L;
      delete size record of I from S;
      add I_1 := (y, q_y) and I_2 := (q_y, y') to L;
      find sizes s_i of I_i, i = 1, 2 by querying P;
      add s_i, i = 1, 2 to S;
    endif
  endif
  i := i + 1;
endwhile
return (B_max, s_max);
```

Proposition 2 *Let $k^+ = |\mathcal{Q}' \cap X^+|$ and $k^- = |\mathcal{Q}' \cap X^-|$. The algorithm* `GeneralSweep`$(x',\ \mathcal{Q}')$ *correctly finds a maximum box with the vertical boundary at x' in $O((k^+ + k^-)(\log k^+ + \log k^-))$ time.*

Proof. The proof is similar to the one of the Proposition 1. □

Assume that the queue of all points $\mathcal{Q}$, in the order of increase of their x-coordinates, has already been constructed. For any $x \in \mathbb{R}$, let the "right" event subqueues $\mathcal{Q}_{\geq x} = \{\mathbf{q} \in \mathcal{Q} : q_x \geq x\}$ and $\mathcal{Q}_{>x} = \{\mathbf{q} \in \mathcal{Q} : q_x > x\}$ be ordered similarly to $\mathcal{Q}$ and the "left" event subqueues $\mathcal{Q}_{\leq x} = \{\mathbf{q} \in \mathcal{Q} : q_x \leq x\}$ and $\mathcal{Q}_{<x} = \{\mathbf{q} \in \mathcal{Q} : q_x < x\}$ be ordered in the decreasing order of the x-coordinate. Note that we require negative points to have a priority over positive on all of these queues. In the following algorithm, we assume that the median computation takes into account potential multiplicity of x-coordinates:

Algorithm 4 `ApproxMaxBox2`$(\mathcal{Q})$
Input: $\mathcal{Q}$ – list of positive and negative points sorted by increasing x-coordinate.
Output: B_2, s_2 – a box within factor two of the maximum and its size.

```
if |Q| ≤ 2 then
  return the maximum box and its size;
endif
x_0.5 :=median of x-coordinates of points on Q;
call GeneralSweep(x_0.5, Q_≤x_0.5);
call GeneralSweep(x_0.5, Q_≥x_0.5);
call ApproxMaxBox2(Q_<x_0.5);
call ApproxMaxBox2(Q_>x_0.5);
s_2 :=maximum size over all calls;
B_2 :=corresponding box;
return (B_2, s_2);
```

Proposition 3 *The algorithm* `ApproxMaxBox2`$(\mathcal{Q})$ *finds a box of size within factor two of the maximum in $O((r+s)\log^2(r+s))$ time.*

Proof. We shall prove the correctness by induction. If $|\mathcal{Q} \cap X^+| \leq 2$ the output is correct. The calls to the line sweep subroutine find the maximum boxes having $x = p_x$ as its left and right boundaries. If a maximum box B intersects with $x = p_x$, then the box of at least half of its size will be found in these calls because $x = p_x$ splits the B into two boxes of the form examined in the `GeneralSweep`$(x',\ \mathcal{Q}')$ subroutine. Otherwise, the recursive calls will return a box within a factor two of the maximum either entirely to the left or to the right of $x = p_x$.

Let $T(k)$ be the worst case time for a recursive call when $|\mathcal{Q}| = k$. Finding the median of $\mathcal{Q}$ is at most $O(k)$. Sizes of $\mathcal{Q}_{\leq x_{0.5}}$ and $\mathcal{Q}_{\geq x_{0.5}}$ are at most k, while sizes of $\mathcal{Q}_{<x_{0.5}}$ and $\mathcal{Q}_{>x_{0.5}}$ are at most $k/2$. Since $\mathcal{Q}$ contains a union of X^+ and X^- and $\log r + \log s = \Theta(\log(r+s))$, by using Proposition 2 we get the following recursion:

$$T(k) = 2T(k/2) + f(k),$$

where $f(k) = O(k \log k)$. The solution of this recursion is $O(k \log^2 k)$. Stated complexity follows. □

4 Approximation algorithms for the maximum bichromatic and numerical discrepancy

In this section, we shall present approximation algorithms finding maximum bichromatic discrepancy and the maximum numerical discrepancy within factor 2 of the optimum and running in $O((r+s)\log^2(r+s))$ and $O(|X|\log^3|X|)$ time, respectively.

Given sets X^+, X^- and a point $(x_0, y_0) \in X^+ \cup X^-$, paper [5] presented an algorithm to find the rectangle $[\mathbf{l}, \mathbf{u}]$ with the maximum bichromatic discrepancy among all the rectangles in the left half space $x \leq x_0$ such that $x_0 = u_x \geq l_x$. Symmetrically, one can find a rectangle $[\mathbf{l}, \mathbf{u}]$ with the maximum bichromatic discrepancy among all the rectangles in the right half space $x \geq x_0$ such that $x_0 = l_x \leq u_x$. Let us denote these two calls to the algorithm for the rectangles in the left and right half spaces as $D(x_0, \mathcal{Q}_{\leq x_0})$ and $D(x_0, \mathcal{Q}_{\geq x_0})$, respectively. The running time of each call is $O((r+s)\log(r+s))$.

In the following, we shall use the algorithm $D(x, S)$ as a subroutine to obtain an approximation algorithm for maximum bichromatic discrepancy with complexity $O((r+s)\log^2(r+s))$ and factor 2:

Algorithm 5 `ApproxMaxDiscrep`($\mathcal{Q}$)
Input: $\mathcal{Q}$ – list of positive and negative points sorted by increasing x-coordinate.
Output: B, s – a box with at least half of the maximum bichromatic discrepancy and its size.

if $|\mathcal{Q}| \leq 2$ **then**
 return the maximum box and its size;
endif
$x_{0.5}$:=`median of` x`-coordinates of points on` $\mathcal{Q}$;
`call` $D(x_{0.5}, \mathcal{Q}_{\leq x_{0.5}})$;
`call` $D(x_{0.5}, \mathcal{Q}_{> x_{0.5}})$;
`call ApproxMaxDiscrep`($\mathcal{Q}_{< x_{0.5}}$);
`call ApproxMaxDiscrep`($\mathcal{Q}_{> x_{0.5}}$);
s_2 :=`maximum discrepancy over all calls;`
B_2 :=`corresponding box;`
return (B_2, s_2);

Proposition 4 *The algorithm* `ApproxMaxDiscrep`($\mathcal{Q}$) *finds a box of a size within factor 2 of the maximum discrepancy in* $O((r+s)\log^2(r+s))$ *time.*

Proof. We will use an inductive proof similar to that one of Proposition 3.

When a box B is split into two boxes B_L, B_R by a line $x = x_{0.5} + \epsilon$ such that there is no any point of X^+, X^- on the line, the bichromatic discrepancy of B is less or equal to the sum of the bichromatic discrepancies of B_L and B_R. So one of the bichromatic discrepancies of B_L and B_R is at least half of the bichromatic discrepancy of B. The calls $D(x_{0.5}, \mathcal{Q}_{\leq x_{0.5}})$ and $D(x_{0.5}, \mathcal{Q}_{> x_{0.5}})$ will find boxes with discrepancies of at least those of B_L and B_R. So, by the same reasoning as in the proof of Proposition 3, this algorithm is a factor 2 approximation algorithm.

The running time is $O((r+s)\log^2(r+s))$ by the analysis of Proposition 3. □

Similarly, we shall obtain a factor 2 approximation algorithm for maximum numerical discrepancy.

Proposition 5 *A box of size within factor two of the maximum numerical discrepancy can be found in* $O(|X|\log^3|X|)$ *time.*

References

[1] Jeet Chaudhuri, Subhas C. Nandy, and Sandip Das. Largest empty rectangle among a point set. *Journal of Algorithms*, 46(1):54–78, 2003.

[2] B. Chazelle, R. L. Drysdale, and D. T. Lee. Computing the largest empty rectangle. *SIAM Journal on Computing*, 15(1):300–315, 1986.

[3] Amitava Datta and Subbiah Soundaralakshmi. An efficient algorithm for computing the maximum empty rectangle in three dimensions. *Information Sciences*, 128(1-2):43–65, 2000.

[4] D. Dobkin and D. Gunopulos. Computing the rectangle discrepancy. In *3rd Annual video review of computational geometry*, pages 385–386, 1994.

[5] 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):453–470, 1996.

[6] David P. Dobkin, Herbert Edelsbrunner, and Mark H. Overmars. Searching for empty convex polygons. *Algorithmica*, 5(4):561–571, 1990.

[7] J. Eckstein, P. Hammer, Y. Liu, M. Nediak, and B. Simeone. The maximum box problem and its application to data analysis. *Computational Optimization and Applications*, 23(3):285–298, 2002.

[8] W. Maass. Efficient agnostic PAC-learning with simple hypotheses. In *Proc. of the 7th annual ACM conference on computational learning theory*, pages 67–75, 1994.

[9] A. Naamad, D. T. Lee, and W.-L. Hsu. On the maximum empty rectangle problem. *Discrete Applied Mathematics*, 8(3):267–277, 1984.