Maximum-Weight Planar Boxes in $O(n^2)$ Time (and Better)

Jérémy Barbay* Timothy M. Chan† Gonzalo Navarro‡ Pablo Pérez-Lantero§

Abstract

Given a set P of n points in $\mathbb{R}^d$, where each point p of P is associated with a weight $w(p)$ (positive or negative), the Maximum-Weight Box problem consists in finding an axis-aligned box B maximizing $\sum_{p\in B\cap P} w(p)$. We describe algorithms for this problem in two dimensions that run in the worst case in $O(n^2)$ time, and much less on more specific classes of instances. In particular, these results imply similar ones for the Maximum Bichromatic Discrepancy Box problem. These improve by a factor of $\Theta(\log n)$ on the best worst-case complexity previously known for these problems, $O(n^2 \lg n)$ [Cortés et al., J. Alg., 2009; Dobkin et al., J. Comput. Syst. Sci., 1996].

1 Introduction

Consider a set P of n points in $\mathbb{R}^d$, such that the points are in general position (i.e., no pair of points share the same x or y coordinate). Each point p of P is assigned a weight $w(p) \in \mathbb{R}$ that can be either positive or negative. For any subset $B \subseteq \mathbb{R}^d$ let $W(B) := \sum_{p\in B\cap P} w(p)$. A *box* is an axis-aligned hyper-rectangle, and we say that the *weight* of a box B is $W(B)$. We consider the Maximum-Weight Box problem, which given P and $w()$ consists in finding a box B with maximum weight $W(B)$.

Related work. In one dimension, the coordinates of the points matter only in the order they induce on their weights, and the problem reduces to the Maximum-Sum Consecutive Subsequence problem [3], which can be solved in $O(n)$ time if the coordinates are already sorted. Cortés et al. [6] solved the dynamic version of this problem supporting updates of weights for a fixed point set. They described a data structure called MCS-tree, which supports in $O(\lg n)$ time both updates and Maximum-Sum Consecutive Subsequence queries on any interval of the sequence of points. The Maximum-Weight Box problem in two dimensions was introduced by Cortés et al. [6], who gave an algorithm running in $O(n^2 \lg n)$ time within $O(n)$ space. Their algorithm is based on MCS-trees: they reduce any instance of the Maximum-Weight Box problem in two dimensions to $O(n^2)$ instances of the problem in one dimension, each solved dynamically in $O(\lg n)$ time by using the MCS-tree.

We consider the Maximum-Weight Box problem in two dimensions on a set P of n weighted points, such that no pair of points share the same x or y coordinate.

Basic definitions. A *strip* is the area delimited by two lines parallel to the same axis. Given the point set P, we say that a strip S is *monochromatic* if $S \cap P$ is not empty and the weights of all elements of $S \cap P$ have the same sign. A monochromatic strip S is *positive* (resp. *negative*) if S contains points of P with positive (resp. negative) weights. We say that P *is composed of* δ *strips* if P can be covered by δ pairwise disjoint monochromatic strips of alternating signs. Given any bounded set $S \subset \mathbb{R}^2$, let $\texttt{Box}(S)$ denote the smallest box covering S.

Results. We present the following results for the Maximum-Weight Box problem in two dimensions. All our algorithms use space linear in the number of input points. Over instances composed of n weighted points, our general algorithm runs in $O(n^2)$ time (Theorem 2). Although this result can be deduced from new results on the Klee's Measure problem [5], it is a more direct and simplified (non-trivial) solution, which further provides smaller running times on specific classes on instances. Namely, if the point set P is composed of $\delta \in [1..n]$ (either horizontal or vertical) strips, our algorithm executes adaptively in $\text{SORT}(n) + O(\delta n) \subset O(n \lg n + \delta n) \subset O(n^2)$ time (Theorem 4), where $\text{SORT}(n)$ is the time required to sort the elements of P by their x-coordinates and by their y-coordinates ($O(n \lg n)$ in the Comparison Model, $O(n \lg \lg n)$ in the RAM model, and $O(n\sqrt{\lg \lg n})$ with randomization in the RAM model if the coordinates of the points are integer numbers [10]).

*Department of Computer Science, University of Chile, Chile. jbarbay@dcc.uchile.cl. Partially supported by grant CONICYT, FONDECYT/Regular 1120054, Chile.

†Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada. tmchan@cs.uwaterloo.ca.

‡Department of Computer Science, University of Chile, Chile. gnavarro@dcc.uchile.cl. Partially funded by Millennium Nucleus Information and Coordination in Networks ICM/FIC P10-024F, Mideplan, Chile.

§Escuela de Ingeniería Civil en Informática, Universidad de Valparaíso, Chile. pablo.perez@uv.cl. Partially supported by grant CONICYT, FONDECYT/Iniciación 11110069, Chile.

Applications to other known problems. Let P be a set of n planar points, each being colored either red or blue.

The Maximum Bichromatic Discrepancy Box problem [6, 7] consists in finding a box that maximizes the absolute difference between the numbers of red and blue points that it contains, and was solved in $O(n^2 \lg n)$ time by Dobkin et al. [7]. Any instance of this problem can be reduced to two particular instances of the Maximum-Weight Box problem [6]. In the first one red points have weight $+1$ and blue points weight -1, and conversely in the second one. Then our results can be applied and imply an $O(n^2)$ worst-case time algorithm for the Maximum Bichromatic Discrepancy Box problem, improving upon previous $O(n^2 \lg n)$-time algorithms [6, 7].

The Maximum Box problem [6,8,11] consists in finding a box B containing the maximum number of blue points and no red point. Eckstein et al. [8] introduced it in general dimension, proving that if the dimension d of the points is part of the input then the problem is NP-hard. In two dimensions it was later solved in $O(n^2 \lg n)$ time by Liu and Nediak [11]. In 2010 Backer et al. [1] showed that the Maximum Box problem in two dimensions can be solved in $O(n \lg^3 n)$ time and $O(n \lg n)$ space, and that for any fixed dimension $d \geq 3$ it can be solved in time within $O(n^d \lg^{d-2} n)$.

Any instance of the Maximum Box problem is equivalent to a particular case of the Maximum-Weight Box problem in which blue points have weight $+1$ and red points have weight $-\infty$ [6]. Then our techniques can be applied and imply $O(n^2)$ worst-case time algorithms for this problem. While this time complexity is worse than the best known solution [1], it requires only linear space, which in some cases can be an important advantage over the $O(n \lg n)$ space required by Backer et al.'s solution [1].

Note that our specialized results are faster on some classes of instances which arise naturally in applications, such as instances where one needs to find a maximum box over an imbalanced red-blue dataset in data mining and/or data analysis [8, 9, 13]. Generally, if the ratio of the number of blue points over the number of red points is within $\omega(1)$, then our techniques yield a running time within $\text{SORT}(n) + o(n^2)$ on an instance of n points.

Outline. In Section 2 we describe the general $O(n^2)$-time algorithm. In Section 3 we obtain the adaptive algorithm running in $\text{SORT}(n) + O(\delta n)$ time, where δ is the number of strips of the point set. Finally, in Section 4, we discuss further adaptive results, a connection to Klee's Measure problem, potential polylogarithmic-factor speedups, and open problems.

2 Quadratic worst-case time algorithm

Assume the elements of P are sorted twice, first by x-coordinates and second by y-coordinates, in $\text{SORT}(n)$ time.

We say that $X \subseteq P$ is a *box set* if X is the intersection of P with some box. For any box set $X \subseteq P$ we define the *score* of X, $\mathtt{S}(X)$, as the following four boxes (see Figure 1). Let $[x_1, x_2] \times [y_1, y_2] := \mathtt{Box}(X)$:

(1) $\mathtt{Box}(X)$;

(2) a maximum-weight box $\mathtt{B}_L(X) \subseteq \mathtt{Box}(X)$ of X of the form $[x_1, x] \times [y_1, y_2]$, $x_1 \leq x \leq x_2$;

(3) a maximum-weight box $\mathtt{B}_R(X) \subseteq \mathtt{Box}(X)$ of X of the form $[x, x_2] \times [y_1, y_2]$, $x_1 \leq x \leq x_2$; and

(4) a maximum-weight box $\mathtt{B}_0(X) \subseteq \mathtt{Box}(X)$ of X of the form $[x, x'] \times [y_1, y_2]$, $x_1 \leq x \leq x' \leq x_2$.

For each of these boxes we keep only two opposed vertices defining it and its weight, so representing a box set X by $\mathtt{S}(X) := (\mathtt{Box}(X), \mathtt{B}_L(X), \mathtt{B}_R(X), \mathtt{B}_0(X))$ requires only constant space.

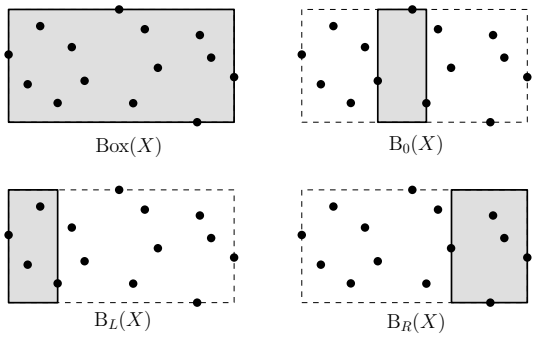

Figure 1: The score $\mathtt{S}(X) = (\mathtt{Box}(X), \mathtt{B}_L(X), \mathtt{B}_R(X), \mathtt{B}_0(X))$ of a box set $X \subseteq P$.

We say that a box set $X \subseteq P$ is *scored* if $\mathtt{S}(X)$ is computed, and we use $\mathtt{Box}(X)$ to represent X instead of X itself. Let the operator $\oplus : 2^P \times 2^P \to 2^P$ be defined over all pairs (X_1, X_2) of scored box sets of P such that: X_1 and X_2 can be separated with a vertical line, X_1 is to the left of X_2, and $X_1 \cup X_2$ is a box set. Then $X_1 \oplus X_2$ returns the scored set $X_1 \cup X_2$, and it can be computed in $O(1)$ time from the next observations:

$$W(X_1 \cup X_2) = W(X_1) + W(X_2).$$

$$W(\mathtt{B}_L(X_1 \cup X_2)) = \max \left\{ \begin{array}{l} W(\mathtt{B}_L(X_1)) \\ W(X_1) + W(\mathtt{B}_L(X_2)). \end{array} \right.$$

$$W(\mathtt{B}_R(X_1 \cup X_2)) = \max \left\{ \begin{array}{l} W(\mathtt{B}_R(X_2)) \\ W(X_2) + W(\mathtt{B}_R(X_1)). \end{array} \right.$$

$$W(\mathtt{B}_0(X_1 \cup X_2)) = \max \begin{cases} W(\mathtt{B}_0(X_1)) \\ W(\mathtt{B}_0(X_2)) \\ W(\mathtt{B}_R(X_1)) + W(\mathtt{B}_L(X_2)). \end{cases}$$

Notice that by applying the operators $\oplus$ to singletons $\{p\}$ over all points p of P in left-to-right order, we can compute $\mathtt{B}_0(P)$, i.e., the maximum-weight vertical strip, in $O(n)$ time. After projection to the x-axis, this immediately gives a linear-time algorithm for the Maximum-Sum Consecutive Subsequence problem, studied by Bentley [3] and often taught in undergraduate algorithms classes.

Let $\mathcal{S}$ be a horizontal strip such that exactly m points of P are not in $\mathcal{S}$. The vertical lines passing through the m points of $P \setminus \mathcal{S}$ split $\mathcal{S}$ into $m+1$ boxes denoted $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{m+1}$ from left to right. Let B be a box of maximum weight that has its top side above $\mathcal{S}$ and its bottom side below $\mathcal{S}$. Suppose that the left and right sides of B intersect $\mathcal{S}_i$ and $\mathcal{S}_j$ $(1 \le i \le j \le m+1)$, respectively. If $i \neq j$, then $W(B \cap \mathcal{S}_i)$ and $W(B \cap \mathcal{S}_j)$ are precisely $W(\mathtt{B}_R(P \cap \mathcal{S}_i))$ and $W(\mathtt{B}_L(P \cap \mathcal{S}_j))$, respectively (see Figure 2). Therefore we have $W(B) = W(\mathtt{B}_R(P \cap \mathcal{S}_i)) + \sum_{t=i+1}^{j-1} W(\mathcal{S}_t) + W(\mathtt{B}_L(P \cap \mathcal{S}_j)) + W(B \setminus \mathcal{S})$. On the other hand, if $i = j$, then $W(B)$ equals $W(\mathtt{B}_0(P \cap \mathcal{S}_i))$.

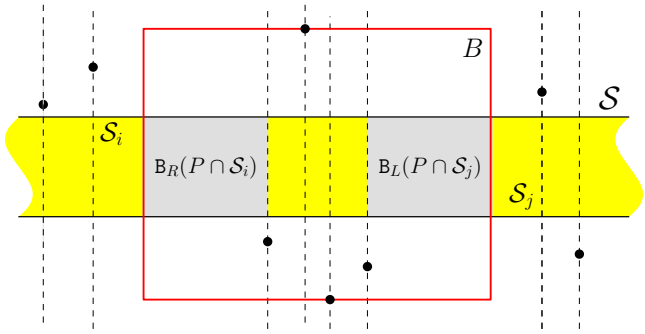

Figure 2: The strip $\mathcal{S}$ is partitioned into $m+1$ boxes $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{m+1}$ by the vertical lines passing through the m points in $P \setminus \mathcal{S}$. If the left and right sides of an optimal box B cross $\mathcal{S}_i$ and $\mathcal{S}_j$, respectively, then they are determined by $\mathtt{B}_R(P \cap \mathcal{S}_i)$ and $\mathtt{B}_L(P \cap \mathcal{S}_j)$.

Consider the following Strip-Constrained Maximum-Weight Box problem: *Let $\mathcal{P}$ be a weighted point set and $\mathcal{S}$ be a horizontal strip so that: $\mathcal{P} \setminus \mathcal{S}$ consists of n points already sorted from left to right; $\mathcal{S}$ splits $\mathcal{P} \setminus \mathcal{S}$ into two halves; the vertical lines through the points of $\mathcal{P} \setminus \mathcal{S}$ split $\mathcal{S}$ into the boxes $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{n+1}$ from left to right; and the points of $\mathcal{P} \cap \mathcal{S}$ are summarized by the scored box sets $\mathcal{P} \cap \mathcal{S}_1, \ldots, \mathcal{P} \cap \mathcal{S}_{n+1}$. Find a maximum-weight box of $\mathcal{P}$, with the top side above $\mathcal{S}$ and the bottom side below $\mathcal{S}$.*

The key to our new solution is an $O(n^2)$-time algorithm for this constrained problem, using an approach which may be nick-named "divide-summarize-and-conquer".

Lemma 1 *The* Strip-Constrained Maximum-Weight Box *problem admits a solution in $O(n^2)$ time and $O(n)$ space.*

Proof. Let $F(n)$ denote the time required to solve a given instance of the Strip-Constrained Maximum-Weight Box problem over n points. We apply divide-and-conquer: Split the points of $\mathcal{P}$ above (resp. below) $\mathcal{S}$ into two halves with a horizontal line ℓ_1 (resp. ℓ_2). Let P_1 denote the points above ℓ_1, P_2 denote the points in between ℓ_1 and $\mathcal{S}$, P_3 denote the points in between $\mathcal{S}$ and ℓ_2, and P_4 denote the points below ℓ_2. Then the problem can be reduced to the next four subproblems:

(1) the points of $P_1 \cup P_4$ outside a strip $\mathcal{S}'$ covering $P_2 \cup P_3 \cup \mathcal{S}$;

(2) the points of $P_1 \cup P_3$ outside a strip $\mathcal{S}'$ covering $P_2 \cup \mathcal{S}$;

(3) the points of $P_2 \cup P_3$ outside the strip $\mathcal{S}' = \mathcal{S}$; and

(4) the points of $P_2 \cup P_4$ outside a strip $\mathcal{S}'$ covering $P_3 \cup \mathcal{S}$.

The reduction to subproblem (1) can be done in $O(n)$ time as follows: Take each point p of $P_2 \cup P_3$ and compute the score $\mathtt{S}(\{p\})$. Simulate the merging of the left-to-right orders of $P_1 \cup P_4$, $P_2 \cup P_3$, and $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{n+1}$ (each of which can be obtained in $O(n)$ time) to compute the corresponding scored box sets in the new strip $\mathcal{S}'$. This computation can be done by applying the operator $\oplus$ to successive scored box sets in between consecutive points of $P_1 \cup P_4$ in the left-to-right order. The reductions to subproblems (2)–(4) are similar.

The base case occurs when $n \in \{1, 2\}$. In the most general setting $(n = 2)$ we have one point p_1 above S and one point p_2 below S, defining boxes $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ on $\mathcal{S}$. Assume w.l.o.g. that p_1 is to the left of p_2 and $w(p_1), w(p_2) > 0$ (for example, if $w(p_1) < 0$, we can eliminate p_1). Then the solution is $\mathtt{B}_0\big((\mathcal{P} \cap \mathcal{S}_1) \cup \{p_1\} \cup (\mathcal{P} \cap \mathcal{S}_2) \cup \{p_2\} \cup (\mathcal{P} \cap \mathcal{S}_3)\big)$, which can be computed in constant time by applying the $\oplus$ operator.

This yields the recurrence

$$F(n) \in 4F(n/2) + O(n),$$

where $F(1) \in O(1)$. Then $F(n) \in O(n^2)$. As for the space $G(n)$, since the four subproblems are solved independently one after the other, the recurrence is $G(n) \in G(n/2) + O(n)$, whose solution is within $O(n)$. □

The reduction from the original Maximum-Weight Box problem to the constrained problem follows from a more straightforward divide-and-conquer:

Theorem 2 *The* Maximum-Weight Box *problem admits a solution in $O(n^2)$ time and $O(n)$ space on instances of n points.*

Proof. We first sort the points of P by their x-coordinates in $\mathrm{SORT}(n)$ time and then apply a recursive procedure, whose time over n weighted points will be $T(n)$. The recursion applies divide-and-conquer as follows: Draw a horizontal strip $\mathcal{S}$ (a line) splitting P into two halves P_1 and P_2, where P_1 is above $\mathcal{S}$ and P_2 is below $\mathcal{S}$. Then we can find a maximum-weight box B_1 for P_1, a maximum-weight box B_2 for P_2, and a maximum-weight box $B_{1,2}$ for $P_1 \cup P_2$ restricted to intersect $\mathcal{S}$. Then the box among B_1, B_2, and $B_{1,2}$ maximizing $W()$ is the solution. To compute $B_{1,2}$ we will use the solution for the Strip-Constrained Maximum-Weight Box problem over P and $\mathcal{S}$, for which we split $\mathcal{S}$ into $n+1$ empty scored boxes $\mathcal{S}_1, \ldots, \mathcal{S}_n$ according to all the x-coordinates of P. This requires $O(n)$ time and then Lemma 1 allows us to compute $B_{1,2}$ in $O(n^2)$ time and $O(n)$ space. Since B_1 and B_2 are computed recursively, the time complexity is

$$T(n) \in 2T(n/2) + O(n^2),$$

where $T(1) \in O(1)$. Hence $T(n) \in O(n^2)$. As for the space $S(n)$, the three subproblems are solved independently one after the other, and thus it holds $S(n) \in \max\{S(n/2), S(n/2), O(n)\} \subseteq O(n)$. □

3 δ-sensitive analysis

Assume that P is composed of $\delta \in [1..n]$ strips, and suppose w.l.o.g. that these strips are horizontal. These strips can be identified in $O(n)$ time from the sorting of the points in P by their y-coordinates. One does not need to consider boxes whose horizontal sides are in the middle of some of these strips: there always exists an optimal box such that each horizontal side is aligned with an edge of some strip; specifically, the top (resp. bottom) of an optimal box will align with a positive point at the top (resp. bottom) of a positive strip. Using this observation we refine the results of Section 2.

Lemma 3 *The* Strip-Constrained Maximum-Weight Box *problem admits a solution in $O(\delta n)$ time and $O(n)$ space if the points of $\mathcal{P}$ above (resp. below) $\mathcal{S}$ are composed of $\delta/2$ strips.*

Proof. Let $F(n, \delta)$ denote the time required to solve the problem. We modify the divide-and-conquer algorithm from the proof of Lemma 1 as follows: We split the points above $\mathcal{S}$ with a horizontal line ℓ_1 and the points below $\mathcal{S}$ with a horizontal line ℓ_2, and define $P_1, \ldots, P_4$ as before. However, we choose ℓ_1 and ℓ_2 differently, not to ensure that each P_i has $n/4$ points, but to ensure that each P_i is composed of $\delta/4$ strips. Let n_i denote the size of P_i (so that $n_1 + n_2 + n_3 + n_4 = n$).

The base case arises when there is at most one strip above (resp. below) $\mathcal{S}$, and can be solved as follows: Assume w.l.o.g. that the weights of these at most two strips are positive (if one of the strips has all negative weights, we can eliminate all of its points). Then the solution is $\mathtt{B}_0(P)$, which can be computed by applying the operator $\oplus$ to the sequence, arranged in left-to-right order, consisting of $\mathcal{P} \cap \mathcal{S}_1, \ldots, \mathcal{P} \cap \mathcal{S}_{n+1}$ together with singletons $\{p_i\}$ over all p_i in $\mathcal{P} \setminus \mathcal{S}$. The base case then requires $O(n)$ time.

The recurrence is now modified to the following:

$$\begin{aligned} F(n,\delta) \quad \in \quad & F(n_1+n_3,\delta/2) + F(n_1+n_4,\delta/2) \\ & + F(n_2+n_3,\delta/2) + F(n_2+n_4,\delta/2) \\ & + O(n). \end{aligned}$$

where $F(n, 1) \in O(n)$. Observe that the recursion tree for $F(n, \delta)$ has at most $\lg \delta$ levels, and that in the i-th level the computation time besides recursive calls is $O(2^i n)$. Then $F(n, \delta) \in O(\delta n)$. The space is within $O(n)$ as in Theorem 2. □

Theorem 4 *The* Maximum-Weight Box *problem admits a solution in* $\mathrm{SORT}(n) + O(\delta n)$ *time and $O(n)$ space on instances of n points composed of δ strips.*

Proof. Let $T(n, \delta)$ denote the time required to solve the Maximum-Weight Box problem over n points composed of δ strips. We apply divide-and-conquer as in Theorem 2, but selecting strip $\mathcal{S}$ such that both resulting sets P_1 and P_2 are composed of $\delta/2$ strips, and n_1 points and n_2 points respectively. If there is only $\delta = 1$ strip then the solution is either empty (if the strip is negative) or all the points (if it is positive), so in the base case it holds $T(n, 1) \in O(n)$. In the recursive case we have:

$$\begin{aligned} T(n,\delta) \quad &= \quad T(n_1,\delta/2) + T(n_2,\delta/2) + F(n,\delta) \\ &\in \quad T(n_1,\delta/2) + T(n_2,\delta/2) + O(\delta n). \end{aligned}$$

The recursion tree of $T(n, \delta)$ has at most $\lg \delta$ levels and in the i-th level the computation time besides recursive calls is $O(\delta n/2^i)$, and thus $T(n, \delta) \in O(\delta n)$. Again, the space is $O(n)$ as before. □

Some naturally occurring instances will have a low number of strips. For example, instances with an unbalanced number of positive and negative points are due to contain few strips. The following corollary captures this observation.

Corollary 5 *Let n_+ and n_- be the number of points with positive and negative weight of an instance of $n = n_+ + n_-$ points, respectively. Then the* Maximum-Weight Box *problem admits a solution in* $\mathrm{SORT}(n) + O(\min\{n_+, n_-\} \cdot n)$ *time.*

Proof. Observe that $\delta \leq 2\min\{n_+, n_-\} + 1$ and apply Theorem 4. □

4 Discussion

Improved adaptive results. Our SORT$(n) + O(\delta n)$ time algorithm adapts well to instances where points associated with weights of same sign can be clustered into a small number of vertical or horizontal strips. It improves a previous $O(\delta n \lg(n/\delta))$ result [2].

To obtain better adaptive algorithms, we can consider more general clusterings into rectangles. One approach is as follows: Call $(C_1, \ldots, C_k)$ a *cluster partition* of P if $\{C_1, \ldots, C_k\}$ is a partition of P and in every axis the orthogonal projections of $\texttt{Box}(C_1), \ldots, \texttt{Box}(C_k)$ are pairwise disjoint.

Given a single rectangular cluster, the optimal box of the whole instance can intersect the cluster boundaries in 10 distinct ways. Considering the top, bottom, left or right edges of the cluster, an optimal box can either intersect none of them (1 case where the optimal box is strictly contained in the cluster), exactly two (4 cases where it contains exactly one of the corners of the cluster), exactly three of them (4 cases where it entirely contains exactly one cluster edge), or exactly four (1 case where it contains the whole cluster). Note that if a box intersects exactly one edge, or exactly two opposite edges (e.g., top and bottom), then there is a box of the same score which intersects no cluster boundaries, since by the definition of a cluster partition, there are no other points exactly above, below, to the left or to the right of the cluster.

In the extended version of this article we will show how, given a partition of the n points into k clusters of respective sizes $n_1, \ldots, n_k$, one can compute the 10 optimal boxes (extending the 4 boxes $\texttt{Box}$, $\texttt{B}_L$, $\texttt{B}_R$, and $\texttt{B}_0$ from Section 2) corresponding to the cases described above in time within $O(\sum_{i=1}^{k} n_i^2)$ and combine these results in time $O(k^2)$ to obtain the optimal box of the whole instance. This yields an $O(\sum_{i=1}^{k} n_i^2 + k^2)$ time algorithm. Finding an optimal cluster partition seems hard.

Connection to Klee's measure problem and higher dimensions. Our $O(n^2)$ worst-case time algorithm is actually a special case of a more general result for a problem related to the well known Klee's Measure problem (computing the volume of a union of n boxes).

In the D-dimensional Weighted Depth problem, we are given a set of n weighted boxes in $\mathbb{R}^D$ and we want a point $p \in \mathbb{R}^D$ that maximizes the *depth*, defined as the sum of the weights of the boxes that contain p. All known algorithms for Klee's Measure problem can be modified to solve the Weighted Depth problem. In particular, Overmars and Yap's algorithm [12] runs in $O(n^{D/2} \lg n)$ time, Chan's algorithm [4] runs in $O(n^{D/2} 2^{O(\lg^* n)})$ time, and a new simple algorithm by Chan [5] runs in $O(n^{D/2})$ time.

The following observation has not been noted before:

Observation 6 *The* Maximum-Weight Box *problem in any constant dimension d can be reduced to the* Weighted Depth *problem in dimension $D = 2d$.*

Proof. Given a point set P in $\mathbb{R}^d$, we map each point $p = (a_1, \ldots, a_d) \in P$ to a region R_p in $\mathbb{R}^{2d}$, consisting of all $2d$-tuples $(x_1, \ldots, x_d, x'_1, \ldots, x'_d)$ such that p lies inside the box with opposite corners $(x_1, \ldots, x_d)$ and $(x'_1, \ldots, x'_d)$; in other words, $R_p = \{(x_1, \ldots, x_d, x'_1, \ldots, x'_d) \mid [(x_1 \leq a_1 \leq x'_1) \vee (x'_1 \leq a_1 \leq x_1)] \wedge \cdots \wedge [(x_d \leq a_d \leq x'_d) \vee (x'_d \leq a_d \leq x_d)]\}$. We can decompose R_p into a constant number of boxes in $\mathbb{R}^{2d}$. The maximum-weight box for P corresponds to a point $(x_1, \ldots, x_d, x'_1, \ldots, x'_d)$ that has the maximum depth among these regions. □

According to the above observation, our $O(n^2)$ result for the Maximum-Weight Box problem in two dimensions is thus not new, but can be deduced from Chan's latest result for the Weighted Depth problem in $D = 4$ dimensions [5]. In fact, the $O(n^2)$ time algorithm presented in this paper is inspired by the algorithm in [5], which is also based on a "divide-summarize-and-conquer" approach. We feel that the algorithm here is nevertheless interesting, because it is a more direct solution, and can be viewed as a further simplification of [5], avoiding the need to work explicitly in 4-dimensional space. (Besides, our $O(n^2)$ time algorithm is a stepping stone towards our SORT$(n) + O(\delta n)$ time algorithm.)

The above observation also implies that the Maximum-Weight Box problem in d dimensions can be solved in $O(n^d)$ time by Chan's new algorithm. Previously, only an $O(n^{2d-2} \lg n)$ time bound was reported [6].

Polylogarithmic-factor speedups and applications. Chan [5] also showed how to further speed up his algorithm by a polylogarithmic factor for the Weighted Depth problem, but only when the dimension is sufficiently large (in particular, not for $D = 4$).

However, in the unweighted case of the Depth problem, it is shown [4, 5] that polylogarithmic speedup is possible for any $D \geq 3$: the running time can be improved to $O((n^{D/2}/\lg^{D/2} n)(\lg \lg n)^{O(1)})$. This extends to the case where the weights are integers with absolute value bounded by $O(1)$, since we can replace a box with positive weight c by c copies of the box, and we can replace a box with negative weight $-c$ by c copies of its complement (which can be decomposed into a constant number of boxes).

In particular, we can thus solve the MAXIMUM-WEIGHT BOX problem for the case of $+1$ and -1 weights in $O((n^d/\lg^d n)(\lg\lg n)^{O(1)})$ time. The same bound thus follows for the MAXIMUM BICHROMATIC DISCREPANCY problem mentioned in the introduction. Previously, only an $O(n^2 \lg n)$ bound was known for $d = 2$ [6, 7]. Similarly, by straightforward changes to incorporate $-\infty$ weights, the MAXIMUM BOX problem can be solved in $O((n^d/\lg^d n)(\lg\lg n)^{O(1)})$ time, improving the previous $O(n^d \lg^{d-2} n)$ bound for $d \geq 3$ [1].

Lower bounds? We conjecture that $O(n^d)$ is the best possible time complexity for the MAXIMUM-WEIGHT BOX problem, ignoring polylogarithmic factors. Unconditional lower bounds are probably difficult to prove. If one could show a converse to Observation 6 (a reduction from some problem related to KLEE'S MEASURE problem in $2d$ dimensions to the MAXIMUM-WEIGHT BOX problem in d dimensions), that might provide evidence for the conjecture. We are only able to show the following:

Observation 7 *The* WEIGHTED DEPTH *problem in any constant dimension d can be reduced to the* MAXIMUM-WEIGHT BOX *problem in dimension d.*

Proof. We first reduce the WEIGHTED DEPTH problem to a special case of the WEIGHTED DEPTH problem where all the input boxes are "dominance" ranges of the form $(-\infty, b_1] \times \cdots \times (-\infty, b_d]$. To see this, for a given $i \in [1..d]$, we replace any input box $[a_1, b_1] \times \cdots \times [a_d, b_d]$ of weight w with two boxes: $[a_1, b_1] \times \cdots \times [a_{i-1}, b_{i-1}] \times (-\infty, b_i] \times [a_{i+1}, b_{i+1}] \times \cdots \times [a_d, b_d]$ of weight w, and $[a_1, b_1] \times \cdots \times [a_{i-1}, b_{i-1}] \times (-\infty, a_i] \times [a_{i+1}, b_{i+1}] \times \cdots \times [a_d, b_d]$ of weight $-w$. By repeating this for each $i \in [1..d]$, each original box is replaced with 2^d boxes of the desired special form.

Now, given an instance of this special case of the WEIGHTED DEPTH problem, we map each input box $b = (-\infty, b_1] \times \cdots \times (-\infty, b_d]$ to the point $p_b = (b_1, \ldots, b_d)$, of the same weight. We have the obvious property that p_b lies inside the box $[x_1, \infty) \times \cdots \times [x_d, \infty)$ iff $(x_1, \ldots, x_d)$ lies inside b. We add an extra point at $(\infty, \ldots, \infty)$ with weight M for a sufficiently large number M. The maximum-weight box containing the resulting point set must be of the form $[x_1, \infty) \times \cdots \times [x_d, \infty)$ because of this extra point, and so corresponds to a point of maximum depth of the given boxes. □

The above observation implies the $W[1]$-hardness of the MAXIMUM-WEIGHT BOX problem with respect to d, since KLEE'S MEASURE problem and the WEIGHTED DEPTH problem are $W[1]$-hard [4]. It also implies the unlikeness of an algorithm that runs faster than $n^{d/2}$ time with current knowledge about KLEE'S MEASURE problem.

References

[1] J. Backer and J. Keil. The mono- and bichromatic empty rectangle and square problems in all dimensions. In *Proceedings of the 9th Latin American Theoretical Informatics Symposium (LATIN'10)*, pages 14–25. 2010.

[2] J. Barbay, G. Navarro, and P. Pérez-Lantero. Adaptive techniques to find optimal planar boxes. In *Proceedings of the 24th Canadian Conference on Computational Geometry (CCCG'12)*, pages 71–76, 2012.

[3] J. Bentley. Programming pearls: algorithm design techniques. *Commun. ACM*, 27(9):865–873, 1984.

[4] T. M. Chan. A (slightly) faster algorithm for Klee's measure problem. *Comput. Geom.*, 43(3):243–250, 2010.

[5] T. M. Chan. Klee's measure problem made easy. Submitted, `https://cs.uwaterloo.ca/~tmchan/easyklee4_13.pdf`, 2013.

[6] C. Cortés, J. M. Díaz-Báñez, P. Pérez-Lantero, C. Seara, J. Urrutia, and I. Ventura. Bichromatic separability with two boxes: A general approach. *J. Algorithms*, 64(2-3):79–88, 2009.

[7] D. P. Dobkin, D. Gunopulos, and W. Maass. Computing the maximum bichromatic discrepancy, with applications to computer graphics and machine learning. *J. Comput. Syst. Sci.*, 52(3):453–470, 1996.

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

[9] X. Guo, Y. Yin, C. Dong, G. Yang, and G. Zhou. On the class imbalance problem. In *Proceedings of the Fourth International Conference on Natural Computation*, pages 192–201, 2008.

[10] Y. Han and M. Thorup. Integer sorting in $O(n\sqrt{\log\log n})$ expected time and linear space. In *Proceedings of the Thirty-Third IEEE Symposium on Foundations of Computer Science*, pages 135–144, 2012.

[11] Y. Liu and M. Nediak. Planar case of the maximum box and related problems. In *Proceeding of the 15th Canadian Conference on Computational Geometry (CCCG'03)*, pages 14–18, 2003.

[12] M. H. Overmars and C.-K. Yap. New upper bounds in Klee's measure problem. *SIAM J. Comput.*, 20(6):1034–1045, 1991.

[13] S. Visa and A. Ralescu. Issues in mining imbalanced data sets - a review paper. In *Proceedings of the Sixteen Midwest Artificial Intelligence and Cognitive Science Conference*, pages 67–73, 2005.