# Faster Algorithms for Largest Empty Rectangles and Boxes

**Timothy M. Chan**
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

## Abstract

We revisit a classical problem in computational geometry: finding the largest-volume axis-aligned empty box (inside a given bounding box) amidst $n$ given points in $d$ dimensions. Previously, the best algorithms known have running time $O(n \log^2 n)$ for $d = 2$ (by Aggarwal and Suri [SoCG'87]) and near $n^d$ for $d \geq 3$. We describe faster algorithms with running time

- $O(n2^{O(\log^* n)} \log n)$ for $d = 2$,
- $O(n^{2.5+o(1)})$ time for $d = 3$, and
- $\widetilde{O}(n^{(5d+2)/6})$ time for any constant $d \geq 4$.

To obtain the higher-dimensional result, we adapt and extend previous techniques for Klee's measure problem to optimize certain objective functions over the complement of a union of orthants.

**2012 ACM Subject Classification** Theory of computation → Computational geometry

**Keywords and phrases** Largest empty rectangle, largest empty box, Klee's measure problem

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2021.24

**Related Version** *Full Version*: https://arxiv.org/abs/2103.08043

**Funding** Supported in part by NSF Grant CCF-1814026.

**Acknowledgements** I thank David Zheng for discussions on the 2D problem.

## 1 Introduction

**Two dimensions.** In the first part of this paper, we tackle the *largest empty rectangle* problem: Given a set $P$ of $n$ points in the plane and a fixed rectangle $B_0$, find the largest rectangle $B \subset B_0$ such that $B$ does not contain any points of $P$ in its interior. Here and throughout this paper, a "rectangle" refers to an axis-parallel rectangle; and unless stated otherwise, "largest" refers to maximizing the area.

The problem has been studied since the early years of computational geometry. While similar basic problems such as largest empty circle or largest empty square can be solved efficiently using Voronoi diagrams, the largest empty rectangle problem seems more challenging. The earliest reference on the 2D problem appears to be by Naamad, Lee, and Hsu in 1984 [26], who gave a quadratic-time algorithm. In 1986, Chazelle, Drysdale, and Lee [15] obtained an $O(n \log^3 n)$-time algorithm. Subsequently, at SoCG'87, Aggarwal and Suri [3] presented another algorithm requiring $O(n \log^3 n)$ time, followed by a more complicated second algorithm requiring $O(n \log^2 n)$ time. The $O(n \log^2 n)$ worst-case bound has not been improved since.[1]

A few results on related questions have been given. Dumitrescu and Jiang [20] examined the combinatorial problem of determining the worst-case number of maximum-area empty rectangles and proved an $O(n2^{\alpha(n)} \log n)$ upper bound; their proof does not appear to have

[1] Aggarwal and Suri's first algorithm can be sped up to run in near $O(n \log^2 n)$ time as well, since it relied on a subroutine for finding row minima in Monge staircase matrices, a problem for which improved results were later found [24, 12]; but these results do not appear to lower the cost of Aggarwal and Suri's second algorithm.


37th International Symposium on Computational Geometry (SoCG 2021).
Editors: Kevin Buchin and Éric Colin de Verdière; Article No. 24; pp. 24:1–24:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

any implication to the algorithmic problem of finding a maximum-area empty rectangle. If the objective is changed to maximizing the perimeter, the problem is a little easier and an optimal $O(n \log n)$-time algorithm can already be found in Aggarwal and Suri's paper [3]. Another related problem of computing a maximum-area rectangle contained in a polygon has also been explored [16].

We obtain a new randomized algorithm that finds the maximum-area empty rectangle in $O(n2^{O(\log^* n)} \log n)$ expected time. This is not only an improvement of almost a full logarithmic factor over the previous 33-year-old bound, but is also close to optimal, except for the slow-growing iterated-logarithmic-like factor (as $\Omega(n \log n)$ is a lower bound in the algebraic decision tree model).

Our solution interestingly uses *interval trees* to efficiently divide the problem into subproblems of logarithmic size, yielding a recursion with $O(\log^* n)$ depth.

**Higher dimensions.** The higher-dimensional analog of the problem is *largest empty box*: Given a set $P$ of $n$ points in $\mathbb{R}^d$ and a fixed box $B_0$, find the largest box $B \subset B_0$ such that $B$ does not contain any points of $P$ in its interior. Here and throughout this paper, a "box" refers to an axis-parallel hyperrectangle; and unless stated otherwise, "largest" refers to maximizing the volume.

Several papers [18, 20, 19, 30] have studied related questions in higher dimensions, e.g., proving combinatorial bounds on the number of optimal boxes, or proving extremal bounds on the volume, or designing approximation algorithms. For the original (exact) computational problem, it is not difficult to obtain an algorithm that finds the largest empty box in $\widetilde{O}(n^d)$ time (for example, as was done by Backers and Keil [4]).[2] At the end of their SoCG'16 paper, Dumitrescu and Jiang [20] explicitly asked whether a faster algorithm is possible:

> "Can a maximum empty box in $\mathbb{R}^d$ for some fixed $d \geq 3$ be computed in $O(n^{\gamma_d})$ time for some constant $\gamma_d < d$?"

Dumitrescu and Jiang attempted to give a subcubic algorithm for the 3D problem, but their conditional solution required a sublinear-time dynamic data structure for finding the 2D maximum empty rectangles containing a query point – currently, the existence of such a data structure is not known.

On the lower bound side, Giannopoulos, Knauer, Wahlström, and Werner [23] proved that the largest empty box problem is $W[1]$-hard with respect to the dimension. This implies a conditional lower bound of $\Omega(n^{\beta d})$ for some absolute constant $\beta > 0$, assuming a popular conjecture on the hardness of the clique problem.

We answer the above question affirmatively. For $d = 3$, we give an $O(n^{5/2+\varepsilon})$-time algorithm, where $\varepsilon > 0$ is an arbitrarily small constant. For higher constant $d \geq 4$, we obtain an algorithm with an intriguing time bound that improves over $n^d$ even more dramatically: $\widetilde{O}(n^{(5d+2)/6})$. For example, the bound is $O(n^{3.667})$ for $d = 4$, $\widetilde{O}(n^{4.5})$ for $d = 5$, and $O(n^{8.667})$ for $d = 10$.

Not too surprisingly, our 3D algorithm achieves subcubic complexity by applying standard range searching data structures (though the application is not be immediately obvious). Dynamic data structures are not used.

The techniques for our higher-dimensional algorithm are perhaps more original and significant, with potential impact to other problems. We first transform the largest empty box problem into a problem about a union of $n$ orthants in $D = 2d$ dimensions (the

[2] Throughout the paper, $\widetilde{O}$ notation hides polylogarithmic factor.

transformation is simple and has been exploited before, such as in [5]). The union of orthants is known to have worst-case combinatorial complexity $O(n^{\lfloor D/2 \rfloor})$ [7]. Interestingly, we show that it is possible to maximize certain types of objective functions over the complement of the union, in time significantly smaller than the worst-case combinatorial complexity.

We accomplish this by adapting known techniques on Klee's measure problem [27, 10, 8, 11]. Specifically, we build on a remarkable method by Bringmann [8] for computing the volume of a union of $n$ orthants in $D$ dimensions in $O(n^{D/3+O(1)})$ time (the $O(1)$ term in the exponent was 2/3 but has been later removed by author [11]). However, maximizing an objective function over the complement of the union is different from summing or integrating a function, and Bringmann's method does not immediately generalize to the former (for example, it exploits subtraction). We introduce extra ideas to extend the method, which results in a bigger time bound than $n^{D/3} = n^{2d/3}$ but nevertheless beats $n^{D/2} = n^d$. In particular, we use some simple graph-theoretical arguments, applied to graphs with $O(D)$ vertices.

**Organization.** We present our 2D algorithm in Sec. 2, our 3D algorithm in the full paper, and our higher-dimensional algorithms in Sec. 3–4 (all these parts may be read independently).

## 2 Largest empty rectangle in 2D

As in previous work [15, 3], we focus on solving a *line-restricted* version of the 2D largest empty rectangle problem: given a set $P$ of $n$ points below a fixed horizontal line $\ell_0$ and a set $Q$ of $n$ points above $\ell_0$, where the $x$-coordinates of all points have been pre-sorted, and given a rectangle $B_0$, find the largest-area rectangle $B \subset B_0$ that intersects $\ell_0$ and is empty of points of $P \cup Q$. By standard divide-and-conquer, an $O(T(n))$-time algorithm for the line-restricted problem immediately yields an $O(T(n)\log n)$-time algorithm for the original largest empty rectangle problem, assuming that $T(n)/n$ is nondecreasing.

We begin by reformulating the line-restricted problem as a problem about horizontal line segments. In the subsequent subsections, we will work with this re-formulation.

For each point $p \in P$, let $s(p)$ be the longest horizontal line segment inside $B_0$ such that $s(p)$ passes through $p$ and there are no points of $P$ above $s(p)$. See Figure 1(a). We can compute $s(p)$ for all $p \in P$ in $O(n)$ time: this step is equivalent to the construction of the standard *Cartesan tree* [31, 22], for which there are simple linear-time algorithms (for example, by inserting points from left to right and maintaining a stack, like Graham's scan, as also re-described in previous papers [15, 3]). Similarly, for each $q \in Q$, let $t(q)$ be the longest horizontal line segment inside $B_0$ such that $t(q)$ passes through $q$ and there are no points of $Q$ below $t(q)$. We can also compute $t(q)$ for all $q \in Q$ in $O(n)$ time.

For a horizontal segment $s$, let $x_s^-$ and $x_s^+$ denote the $x$-coordinates of its left and right endpoints respectively, and let $y_s$ denote its $y$-coordinate. We say that a set $S$ of horizontal segments is *laminar* if for every $s, s' \in S$, either the two intervals $[x_s^-, x_s^+]$ and $[x_{s'}^-, x_{s'}^+]$ are disjoint, or one interval is contained in the other (in other words, the intervals form a "balanceded parentheses" or tree structure). It is easy to see that for the segments defined above, $\{s(p) : p \in P\}$ is laminar and $\{t(q) : q \in Q\}$ is laminar.

The optimal rectangle must have some point $p^* \in P$ on its bottom side and some point $q^* \in Q$ on its top side (except when the optimal rectangle touches the bottom or top side of $B_0$, a case that can be easily dismissed in linear time). Chazelle, Drysdale, and Lee [15] already noted that the case when $[x_{s(p^*)}^-, x_{s(p^*)}^+]$ is contained in $[x_{t(q^*)}^-, x_{t(q^*)}^+]$ can be handled in $O(n)$ time (in their terminology, this is the case of "three supports in one half, one in the

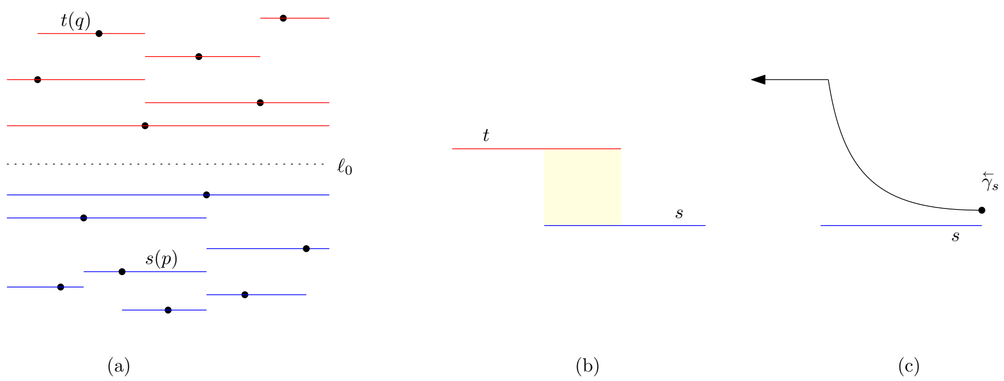


**Figure 1** (a,b) Transforming points into horizontal segments. (c) Pseudo-ray $\overleftarrow{\gamma}_s$.

other").[3] The key remaining case is when $x^-_{t(q^*)} < x^-_{s(p^*)} < x^+_{t(q^*)} < x^+_{s(p^*)}$, where the area of the optimal rectangle is $(x^+_{t(q^*)} - x^-_{s(p^*)})(y_{t(q^*)} - y_{s(p^*)})$. All other cases are symmetric. The problem is thus reduced to the following (see Figure 1(b)):

▶ **Problem 1.** *Given a laminar set $S$ of $n$ horizontal segments and a laminar set $T$ of $n$ horizontal segments, where all $x$-coordinates have been pre-sorted, find a pair $(s,t) \in S \times T$ such that $x^-_t < x^-_s < x^+_t < x^+_s$, maximizing $(x^+_t - x^-_s)(y_t - y_s)$.*

We find it more convenient to work with the corresponding *decision problem*, as stated below. By the author's randomized optimization technique [9], an $O(T(n))$-time algorithm for Problem 2 yields an $O(T(n))$-expected-time algorithm for Problem 1, assuming that $T(n)/n$ is nondecreasing:

▶ **Problem 2.** *Given a laminar set $S$ of $n$ horizontal segments and a laminar set $T$ of $n$ horizontal segments, where all $x$-coordinates have been pre-sorted, and given a value $r > 0$, decide if there exists a pair $(s,t) \in S \times T$ such that $x^-_t < x^-_s < x^+_t < x^+_s$ and $(x^+_t - x^-_s)(y_t - y_s) > r$, and if so, report one such pair. We call such a pair* good.

## 2.1 Preliminaries

To help solve Problem 2, we define a curve $\gamma_s$ for each $s \in S$:

$$\gamma_s(x) = \begin{cases} \frac{r}{x - x^-_s} + y_s & \text{if } x \geq x^-_s + \delta \\ M + x^-_s & \text{if } x < x^-_s + \delta, \end{cases}$$

for a sufficiently small $\delta > 0$ and a sufficiently large $M = M(\delta)$. (The main first part of the curve is a hyperbola.) The condition $(x^+_t - x^-_s)(y_t - y_s) > r$ is met iff the point $(x^+_t, y_t)$ (i.e., the right endpoint of $t$) is above the curve $\gamma_s$, assuming $x^+_t \geq x^-_s + \delta$. Note that these curves form a family of *pseudo-lines*, i.e., every pair of curves intersect at most once: this can be seen from the fact that for any two curves $\gamma_s$ and $\gamma_{s'}$ with $x^-_s \geq x^-_{s'}$, the difference $\gamma_s(x) - \gamma_{s'}(x) = \frac{r(x^-_s - x^-_{s'})}{(x - x^-_s)(x - x^-_{s'})} + y_s - y_{s'}$ is nonincreasing for $x \geq x^-_s$.

[3] The solution is simple: for each $p \in P$, we find the lowest point $q_p \in Q$ with $x$-coordinate in the interval $[x^-_{s(p)}, x^+_{s(p)}]$, and take the maximum of $(x^+_{s(p)} - x^-_{s(p)})(y_{t(q_p)} - y_{s(p)})$. All these lowest points $q_p$ can be found "bottom-up" in the tree formed by the intervals $\{[x^-_{s(p)}, x^+_{s(p)}] : p \in P\}$, in linear total time.

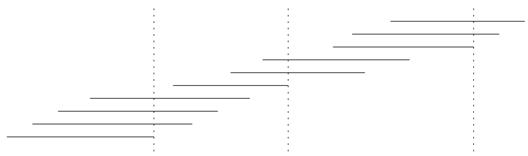

**Figure 2** Proof of Lemma 1(b): the $x$-projected intervals and the division into slabs.

Define the curve segment $\overleftarrow{\gamma}_s$ to be the part of $\gamma_s$ restricted to $x \le x_s^+$. (See Figure 1(c).) These curve segments form a family of *pseudo-rays*. The lower envelope of $n$ pseudo-rays has at most $2n$ edges, by known combinatorial bounds on order-2 Davenport-Schinzel sequences [29]. The following lemma summarizes known subroutines we need on the computation of lower envelopes (proofs are briefly sketched).

▶ **Lemma 1.** *Consider a set of $n$ pseudo-lines, sorted by their* pseudo-slopes*, such that if $\gamma$ and $\gamma'$ intersects and $\gamma$ has smaller pseudo-slope, then $\gamma$ is above $\gamma'$ to the left of the intersection. Assume that the intersection of any two pseudo-lines can be computed in constant time.*

**(a)** *Consider $n$ pseudo-rays that are parts of the given pseudo-lines, such that the $x$-coordinates of the left endpoints are all $-\infty$, and the $x$-coordinates of the right endpoints are monotone (increasing or decreasing) in the pseudo-slopes. Then the lower envelope of these pseudo-rays can be computed in $O(n)$ time.*

**(b)** *Consider $n$ pseudo-segments that are parts of the given pseudo-lines, such that $x$-coordinates of the left endpoints are monotone in the pseudo-slopes and the $x$-coordinates of the right endpoints are monotone in the pseudo-slopes. Then the lower envelope of these pseudo-segments can be computed in $O(n)$ time.*

**Proof.** Part (a) follows by a straightfoward variant of Graham's scan [17] (originally for computing planar convex hulls, or by duality, lower envelopes of lines). We insert pseudo-rays in decreasing order of their right endpoints' $x$-values, while maintaining the portion of the lower envelope to the left of the right endpoint of the current pseudo-ray. In each iteration, by the monotonicity assumption, a prefix or suffix of the lower envelope gets deleted (i.e., popped from a stack).

For part (b), the main case is when both the left and right endpoints are monotonically increasing in the pseudo-slopes (the case when both are monotonically decreasing is symmetric, and the case when they are monotone in different directions easily reduces to two instances of the pseudo-ray case). Greedily construct a minimal set of vertical lines that stab all the pseudo-segments: namely, draw a vertical line at the leftmost right endpoint, remove all pseudo-segments stabbed, and repeat. This process can be done in $O(n)$ time by a linear scan. These vertical lines divide the plane into slabs. (See Figure 2.) In each slab, the pseudo-segments behave like pseudo-rays, so we can compute the lower envelope inside the slab in linear time by applying part (a) twice, for the leftward rays and for the rightward rays (the two envelopes can be merged in linear time). Since each pseudo-segment participates in at most two slabs, the total time is linear. ◀

As an application of Lemma 1(b), we mention an efficient algorithm for a special case of Problem 2, which will be useful later.

▶ **Corollary 2.** *In the case when all segments in $S$ and $T$ intersect a fixed vertical line, Problem 2 can be solved in $O(n)$ time.*

**Proof.** Since $S$ and $T$ are laminar, the $x$-projected intervals in each set are nested. Let $s_1, s_2, \ldots$ be the segments in $S$ with $[x^-_{s_1}, x^+_{s_1}] \subseteq [x^-_{s_2}, x^+_{s_2}] \subseteq \cdots$, and let $t_1, t_2, \ldots$ be the segments in $T$ with $[x^-_{t_1}, x^+_{t_1}] \subseteq [x^-_{t_2}, x^+_{t_2}] \subseteq \cdots$. For each $s_i$, let $a(i)$ be the smallest index with $x^-_{t_{a(i)}} < x^-_{s_i}$, let $b(i)$ be the smallest index with $x^-_{s_i} < x^+_{t_{b(i)}}$, and let $c(i)$ be the largest index with $x^+_{t_{c(i)}} < x^+_{s_i}$. Note that $a(i)$ is monotonically increasing in $i$, and $b(i)$ is monotonically decreasing in $i$, and $c(i)$ is monotonically increasing in $i$. It is straightforward to compute $a(i), b(i), c(i)$ for all $i$ by a linear scan.

The problem reduces to finding a pair $(s_i, t_j)$ such that $\max\{a(i), b(i)\} \leq j \leq c(i)$ and the right endpoint of $t_j$ is above $\gamma_{s_i}$. Define the curve segment $\overline{\gamma}_{s_i}$ to be the part of $\gamma_{s_i}$ restricted to $x \in [\max\{x^+_{t_{a(i)}}, x^+_{t_{b(i)}}\}, x^+_{t_{c(i)}}]$. The problem reduces to finding a $t_j$ whose right endpoint is above some curve segment $\overline{\gamma}_{s_i}$, i.e., above the lower envelope of these curve segments. We can compute this lower envelope in $O(n)$ time by Lemma 1(b) (more precisely, by two invocations of the lemma, as $\max\{x^+_{t_{a(i)}}, x^+_{t_{b(i)}}\}$ consists of a monotonically increasing and a monotonically decreasing part). The problem can be then be solved by linear scan over the envelope and the endpoints of $t_j$. ◀

## 2.2 Algorithm

We are now ready to describe our new algorithm for solving Problem 2, using interval trees and an interesting recursion with $O(\log^* n)$ depth.

▶ **Theorem 3.** *Problem 2 can be solved in $O(n2^{O(\log^* n)})$ time.*

**Proof.** As a first step, we build the standard *interval tree* for the given horizontal segments in $S \cup T$. This is a perfectly balanced binary tree of with $O(\log n)$ levels, where each node corresponds to a vertical slab. The root slab is the entire plane, the slab at a node is the union of the slabs of its two children, and each leaf slab contains no endpoints in its interior. Each segment is stored in the lowest node $v$ whose slab contains the segment (i.e., the segment is contained in $v$'s slab but is not contained in either child's subslab). Note that each segment is stored only once (unlike in another standard structure called the "segment tree"). We can determine the slab containing each segment in $O(1)$ time by an LCA query [6] (which is easier in the case of a perfectly balanced binary tree).

For each node $v$, let $S_v$ (resp. $T_v$) be the set of all segments of $S$ (resp. $T$) stored in $v$. Define the *level* of a segment to be the level of the node it is stored in.

**Case 1.** There exists a good pair $(s^*, t^*)$ where $s^*$ and $t^*$ have the same level. Here, $s^*$ and $t^*$ must be stored in the same node $v$ of the interval tree. Thus, a good pair can be found as follows:

1. For each node $v$, solve the problem for $S_v$ and $T_v$ by Corollary 2 in $O(|S_v| + |T_v|)$ time. Note that all segments in $S_v \cup T_v$ indeed intersect a fixed vertical line (the dividing line at $v$).

The total running time of this step is $O(n)$, since each segment is in only one $S_v$ or $T_v$.

**Case 2.** There exists a good pair $(s^*, t^*)$ where $s^*$ is on a strictly lower level than $t^*$. To deal with this case, we perform the following steps, for some choice of parameter $b \geq \log n$:

**2a.** For each node $v$, compute the lower envelope of the pseudo-rays $\{\overleftarrow{\gamma}_s : s \in S_v\}$ by Lemma 1(a) in $O(|S_v|)$ time; let $\mathcal{E}_v$ denote this envelope restricted to $v$'s slab. Note that because all segments in $S_v$ intersect a fixed vertical line and $S_v$ is laminar, the $x^+_s$ values are monotonically decreasing in the $x^-_s$ values for $s \in S_v$ and so are indeed monotone in the pseudo-slopes of these pseudo-rays.

**2b.** Divide the plane into a set $\Sigma$ of $n/b$ vertical slabs each containing $b$ right endpoints of $T$.

**2c.** For each slab $\sigma \in \Sigma$,

- let $T_\sigma$ be the set of all segments $t \in T$ with right endpoints in $\sigma$, and
- let $S_\sigma$ be the set of all segments $s \in S$ such that $\overleftarrow{\gamma}_s$ appears on $\mathcal{E}_v \cap \sigma$ for some node $v$.

Divide $S_\sigma$ (arbitrarily) into blocks of size $b$ and recursively solve the problem for $T_\sigma$ and each block of $S_\sigma$.

**Correctness.** Consider a good pair $(s^*, t^*)$ with $s^*$ on a strictly lower level than $t^*$. Let $\sigma$ be the slab in $\Sigma$ containing the right endpoint of $t^*$, i.e., $t^* \in T_\sigma$. Let $v$ be the node $s^*$ is stored in. Then $t^*$ intersects the left wall of the slab at $v$ (since $t^*$ must be stored in a proper ancestor of $v$). Now, the right endpoint of $t^*$ is below $\overleftarrow{\gamma}_{s^*}$ and is thus below $\mathcal{E}_v$. Let $\overleftarrow{\gamma}_s$ be the curve on $\mathcal{E}_v$ that the right endpoint of $t^*$ is below, with $s \in S_v$. Then $\overleftarrow{\gamma}_s$ appears on $\mathcal{E}_v \cap \sigma$, and so $s \in S_\sigma$. Since the right endpoint of $t^*$ is below $\overleftarrow{\gamma}_s$, we have $x_s^- < x_{t^*}^+ < x_s^+$, and since $t^*$ intersects the left wall of $v$'s slab, we have $x_{t^*}^- < x_s^-$. So, $(s, t^*)$ is good, and the recursive call for $T_\sigma$ and some block of $S_\sigma$ will find a good pair.

**Analysis.** The total number of edges in all envelopes $\mathcal{E}_v$ is at most $2\sum_v |S_v| \le 2n$. Since the envelopes $\mathcal{E}_v$ have disjoint $x$-projections for nodes $v$ at the same level, and since there are $O(\log n)$ levels, the $O(n/b)$ dividing vertical lines of $\Sigma$ intersect at most $O((n/b)\log n)$ edges among all the envelopes. Thus, $\sum_{\sigma\in\Sigma} |S_\sigma| \le 2n + O((n/b)\log n) = O(n)$ if $b \ge \log n$, and so the total number of recursive calls in step 2c is $O(n/b)$.

**Case 3.** There exists a good pair $(s^*, t^*)$ where $s^*$ is on a strictly higher level than $t^*$. This remaining case is symmetric to Case 2 (by switching $S$ and $T$ and negating $y$-coordinates).

**Total time.** By running the algorithms for all three cases, a good pair is guaranteed to be found if one exists. The running time satisfies the recurrence $T(n) \le O(n/b)T(b) + O(n)$. Setting $b = \log n$ gives $T(n) \le n2^{O(\log^* n)}$. ◀

By the observations from the beginning of this section, we can now solve Problem 1 and the line-restricted problem in $O(n2^{O(\log^* n)})$ expected time, and the original largest empty rectangle problem in $O(n2^{O(\log^* n)} \log n)$ expected time.

▶ **Corollary 4.** *Given $n$ points in $\mathbb{R}^2$ and a rectangle $B_0$, we can compute the maximum-area empty rectangle inside $B_0$ in $O(n2^{O(\log^* n)} \log n)$ expected time.*

## 3 Largest empty anchored box in higher dimensions (warm-up)

To prepare for our solution to the largest empty box problem in higher constant dimensions, we first investigate a simpler variant, the *largest empty anchored box* problem: given a set $P$ of $n$ points in $\mathbb{R}^d$ and a fixed box $B_0$, find the largest-volume anchored box in $B_0$ that does not contain any points of $P$ in its interior, where an *anchored* box has the form $B = (0, x_1) \times \cdots (0, x_d)$ (having the origin as one of its vertices).

Let $\bigcup S$ denote the union of a set $S$ of objects. By mapping a box $B = (0, x_1) \times \cdots (0, x_d)$ to the point $(x_1, \ldots, x_d)$, and mapping each input point $(p_1, \ldots, p_d)$ to the orthant $(p_1, \infty) \times \cdots \times (p_d, \infty)$, the largest empty anchored box problem reduces to:

▶ **Problem 3.** *Define the function $H_{\text{vol}}(x_1, \ldots, x_d) = x_1 x_2 \cdots x_d$. Given a set $S$ of $n$ orthants in $\mathbb{R}^d$ and a box $B_0$, find the maximum of $H_{\text{vol}}$ over $B_0 - \bigcup S$.*

By known results [7], the union of $n$ orthants in $\mathbb{R}^d$ has worst-case combinatorial complexity $O(n^{\lfloor d/2 \rfloor})$ and can be constructed in $\widetilde{O}(n^{\lfloor d/2 \rfloor})$ time. We will show that Problem 3 can be solved faster than explicitly constructing the union.

## 3.1 Preliminaries

A key tool we need is a spatial partitioning scheme due to Overmars and Yap [27] (originally developed for solving Klee's measure problem in $\widetilde{O}(n^{d/2})$ time). The version stated below is taken from [11, Lemma 4.6]; see that paper for a short proof. (The partitioning scheme is also related to "orthogonal BSP trees" [21, 14].)

▶ **Lemma 5.** *Given a set of $n$ axis-parallel flats (of possibly different dimensions) in $\mathbb{R}^d$, and given a parameter $r$, we can divide $\mathbb{R}^d$ into $O(r^d)$ cells (bounded and unbounded boxes) so that each cell intersects $O(n/r^j)$ $(d-j)$-flats.*

*The construction of the cells, along with the conflict lists (lists of all flats intersecting each cell), can be done in $\widetilde{O}(n + r^d + K)$ time,[4] where $K$ is the total size of the conflict lists.*

Call a function $H : \mathbb{R}^d \to \mathbb{R}$ *simple* if it has the form

$$H(x_1, \ldots, x_d) \;=\; h_1(x_1) \cdots h_d(x_d),$$

where each $h_i$ is a univariate step function. The *complexity* of $H$ refers to the total complexity (number of steps) in these step functions. As an illustration of the usefulness of Lemma 5, we first how to maximize simple functions over the complement of a union of orthants in $\widetilde{O}(n^{d/2})$ time:

▶ **Lemma 6.** *Let $H$ be a simple function with $O(n)$ complexity. Given a set $S$ of $n$ orthants in $\mathbb{R}^d$ and a box $B_0$, we can compute the maximum of $H$ in $B_0 - \bigcup S$ in $\widetilde{O}(n^{d/2})$ time for any constant $d \geq 2$.*

**Proof.** Apply Lemma 5 to the $O(n)$ $(d-2)$-flats that pass through the $(d-2)$-faces of the given orthants. This yields a partition of $B_0$ into cells.

Consider a cell $\Delta$. The number of $(d-2)$-flats intersecting $\Delta$ is bounded by $O(n/r^2)$, which can be made 0 by setting $r := \Theta(\sqrt{n})$. Consequently, only $(d-1)$-faces of the given orthants may intersect $\Delta$, i.e., all orthants are 1-sided inside $\Delta$. The union of 1-sided orthants simplifies to the complement of a box (we can use orthogonal range searching or intersection data structure to identify the 1-sided orthants intersecting $\Delta$ and compute this box in $\widetilde{O}(1)$ time [1, 17]). For a simple function $H(x_1, \ldots, x_d) = h_1(x_1) \cdots h_d(x_d)$, we can maximize $H$ over a box by maximizing $h_i(x_i)$ over an interval for each $i \in \{1, \ldots, d\}$ separately. This corresponds to a 1D range maximum query for each $i$, which can be done straightforwardly in $O(\log n)$ time (or more carefully in $O(1)$ time [6]). As the number of cells is $O(r^d) = O(n^{d/2})$, the total running time is $\widetilde{O}(n^{d/2})$. ◀

## 3.2 Algorithm

To improve over $n^{d/2}$, we adapt an approach by Bringmann [8] (originally for solving Klee's measure problem for orthants in $O(n^{d/3+O(1)})$ time). The approach involves first solving the 2-sided special case, and then applying Overmars and Yap's partitioning scheme. A *2-sided orthant* is the set of all points $(x_1, \ldots, x_d) \in \mathbb{R}^d$ satisfying a condition of the form $[x_i \ ? \ a] \wedge [x_j \ ? \ b]$ for some $i, j \in \{1, \ldots, d\}$, where each occurrence of "?" is either $\leq$ or $\geq$. We will adapt the author's subsequent re-interpretation [11, Section 4.1] of Bringman's technique, described in terms of monotone step functions.

▶ **Theorem 7.** *In the case when all the input orthants are 2-sided, Problem 3 can be solved in $\widetilde{O}(n^{\lfloor d/2 \rfloor/2})$ time for any constant $d \geq 4$.*

---

[4] A weaker time bound was stated in [11, Lemma 4.6], but the output-sensitive time bound follows directly from the same construction.

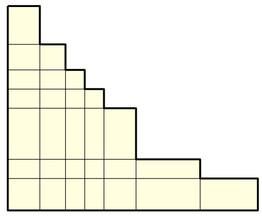

Figure 3 The union of one type of 2-sided orthants.

**Proof.** The boundary of the union of 2-sided orthants of the form $[x_i\ ?\ a] \wedge [x_j\ ?\ b]$ with a fixed $i, j$ and a fixed choice for the two "?"s is a staircase, i.e., the graph of a univariate monotone (increasing or decreasing) step function. (See Figure 3.) Thus, the complement of the union of 2-sided orthants can be expressed as the set of all points $(x_1, \ldots, x_d) \in \mathbb{R}^d$ satisfying an expression $E(x_1, \ldots, x_d)$ which is a conjunction of $O(d^2)$ predicates each of the form $[x_i\ ?\ f(x_j)]$, where $i, j \in \{1, \ldots, d\}$, "?" is $\leq$ or $\geq$, and $f$ is a monotone step function. The total complexity of these step functions is $O(n)$. Conversely, any such expression can be mapped back to the complement of a union of $O(n)$ 2-sided orthants.

We first observe a few simple rules for rewriting expressions:

1. $[x_i \leq f(x_j)] \wedge [x_i \leq g(x_j)]$ can be rewritten as $[x_i \leq \min\{f, g\}(x_j)]$ if $f$ and $g$ are both increasing or both decreasing. Note that the lower envelope $\min\{f, g\}$ is still a monotone step function with $O(n)$ complexity. A similar rule applies for $\geq$.
2. $[x_i \leq f(x_j)]$ can be rewritten as $[x_j \geq f^{-1}(x_i)]$ if $f$ is increasing (the inequality is flipped if $f$ is decreasing). Note that the inverse $f^{-1}$ is still a monotone step function.
3. More generally, $[f(x_i) \leq g(x_j)]$ can be rewritten as $[x_j \geq (f^{-1} \circ g)(x_i)]$ if $f$ is increasing (the inequality is flipped if $f$ is decreasing). Note that the composition $f^{-1} \circ g$ is still a monotone step function with $O(n)$ complexity.
4. $[x_i \leq f(x_j)] \wedge [x_i \leq g(x_k)]$ can be rewritten as the disjunction of $[x_i \leq f(x_j)] \wedge [f(x_j) \leq g(x_k)]$ and $[x_i \leq g(x_k)] \wedge [g(x_k) \leq f(x_j)]$. A similar rule applies for $\geq$.

The plan is to decrease the dimension by repeatedly eliminating variables:

We maintain a simple function $H$. Initially, $H(x_1, \ldots, x_d) = \sigma(x_1) \cdots \sigma(x_d)$, where $\sigma(x)$ denotes the successor of $x$ among the $O(n)$ input coordinate values ($\sigma$ is a step function). We call an index $i$ *free* if the variable $x_i$ appears exactly once in $H$ and is "unaltered", i.e., $h_i(x_i) = \sigma(x_i)$. All indices are initially free.

In each iteration, we pick a free index $i$. Whenever $x_i$ appears more than twice in $E$, we can apply rule 4 (in combination with rules 1–3) to obtain a disjunction of 2 subexpressions, where in each subexpression, the number of occurrences of $x_i$ is decreased. By repeating this process $O(1)$ times (recall that $d$ is a constant), we obtain a disjunction of $O(1)$ subexpressions, where in each subexpression, only at most two occurrences of $x_i$ remain – in at most one predicate of the form $[x_i \leq f(x_j)]$, and at most one predicate of the form $[x_i \geq g(x_k)]$.

We branch off to maximize $H$ over each of these subexpressions separately. In such a subexpression, to eliminate the variable $x_i$ while maximizing $H$, we replace the two predicates $[x_i \leq f(x_j)]$ and $[x_i \geq g(x_k)]$ with $[f(x_j) \geq g(x_k)]$, and replace $x_i$ with $f(x_j)$ in $H$ (i.e., reset $h_j(x_j)$ to $h_j(x_j)\sigma(f(x_j))$, which is still a step function with $O(n)$ complexity). Now, $i$ and $j$ are not free.

We stop a branch when there are no free indices left. At the end, we get a large but $O(1)$ number of subproblems, where in each subproblem, at least $\lceil d/2 \rceil$ variables have been eliminated, i.e., the dimension is decreased to $d' \leq \lfloor d/2 \rfloor$. We solve each subproblem by Lemma 6 in $\widetilde{O}(n^{d'/2})$ time. ◀

We now combine Theorem 7 and Lemma 5 to solve Problem 3:

▶ **Corollary 8.** *Problem 3 can be solved in* $\widetilde{O}(n^{d/3+\lfloor d/2\rfloor/6})$ *time for any constant* $d \geq 4$.

**Proof.** Apply Lemma 5 to the $O(n)$ $(d-3)$-flats and $(d-2)$-flats through the $(d-3)$-faces and $(d-2)$-faces of the given orthants. This yields a partition of $B_0$ into cells.

Consider a cell $\Delta$. The number of $(d-3)$-flats intersecting $\Delta$ is $O(n/r^3)$, which can be made 0 by setting $r := \Theta(n^{1/3})$. The number of $(d-2)$-flats intersecting $\Delta$ is $O(n/r^2) = O(n^{1/3})$. So, inside the cell $\Delta$, all orthants are 2-sided or 1-sided, with $O(n^{1/3})$ 2-sided orthants. The union of 1-sided orthants simplifies to the complement of a box (we can use orthogonal range searching or intersection data structures [1, 17] to identify the 1-sided orthants intersecting $\Delta$ and compute this box). We can thus apply Theorem 7 to maximize $H$ over the cell $\Delta$ in $\widetilde{O}((n^{1/3})^{\lfloor d/2\rfloor/2})$ time. As there are $O(r^d) = O(n^{d/3})$ cells, the total running time is $\widetilde{O}(n^{d/3} \cdot (n^{1/3})^{\lfloor d/2\rfloor/2})$. ◀

▶ **Corollary 9.** *Given* $n$ *points in* $\mathbb{R}^d$ *and a box* $B_0$*, we can compute the maximum-volume empty anchored box inside* $B_0$ *in* $\widetilde{O}(n^{d/3+\lfloor d/2\rfloor/6}) \leq \widetilde{O}(n^{5d/12})$ *time for any constant* $d \geq 4$.

## 4 Largest empty box in higher dimensions

We now adapt the approach from Section 3 to solve the original largest empty box problem in higher constant dimensions. By $d$ levels of divide-and-conquer, it suffices to solve the *point-restricted* version of the problem: given a set $P$ of $n$ points in $\mathbb{R}^d$, a fixed box $B_0$, and a fixed point $o$, find the largest-volume box $B \subset B_0$ that contains $o$ and is empty of points of $P$. An $O(T(n))$-time algorithm for the point-restricted problem immediately yields an $O(T(n)\log^d n)$-time algorithm for the original problem (in fact, the polylogarithmic factor disappears if $T(n)/n^{1+\delta}$ is increasing for some constant $\delta > 0$). Without loss of generality, assume that $o$ is the origin.

By mapping a box $B = (-x_1, x_1') \times \cdots \times (-x_d, x_d')$ (which has volume $(x_1 + x_1') \cdots (x_d + x_d')$) to the point $(x_1, x_1', \ldots, x_d, x_d')$ in $2d$ dimensions, and mapping each input point $p = (p_1, \ldots, p_d)$ to the orthant $(-p_1, \infty) \times (p_1, \infty) \times \cdots (-p_d, \infty) \times (p_d, \infty)$ in $2d$ dimensions (and changing $B_0$ appropriately), the problem reduces to the following variant of Problem 3, after doubling the dimension:

▶ **Problem 4.** *Define the function* $H_{\text{new-vol}}(x_1, \ldots, x_d) = (x_1 + x_2)(x_3 + x_4) \cdots (x_{d-1} + x_d)$ *for an even* $d$*. Given a set* $S$ *of* $n$ *orthants in* $\mathbb{R}^d$ *and a box* $B_0$*, find the maximum of* $H_{\text{new-vol}}$ *over* $B_0 - \bigcup S$.

The above objective function $H_{\text{new-vol}}$ is a bit more complicated than the one from Section 3, and so further ideas are needed…

### 4.1 Preliminaries

For a multigraph $G$ with vertex set $\{1, \ldots, d\}$ (without self-loops), define a *G-function* $H : \mathbb{R}^d \to \mathbb{R}$ to be a function of the form

$$H(x_1, \ldots, x_d) \;=\; \prod_{i=1}^{d} h_i(x_i) \cdot \prod_{e=ij \in G} (h_e'(x_i) + h_e''(x_j)),$$

where $h_i$, $h_e'$, and $h_e''$ are univariate step functions. The *complexity* of $H$ refers to the total complexity of these step functions.

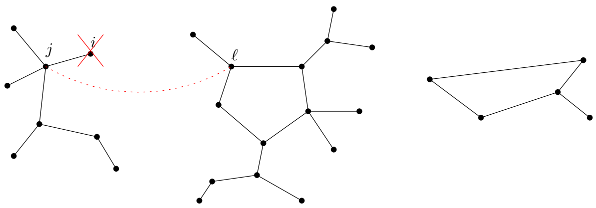


**Figure 4** After removing vertex $i$ and adding edge $j\ell$, the graph $G$ remains a pseudo-forest.

A *pseudo-forest* is a graph where each component is either a tree, or a tree plus an edge – in the latter case, the component is called a *1-tree* (and we allow the extra edge to be a duplicate of an edge in the tree).

▶ **Lemma 10.** *Let $H$ be a $G$-function $H$ with $O(n)$ complexity. Given a box $B_0$, we can compute the maximum of $H$ over $B_0$ in $\widetilde{O}(n)$ time if $G$ is a forest, or $\widetilde{O}(n^2)$ time if $G$ is a pseudo-forest, for any constant $d$.*

**Proof.** For the forest case: Pick a leaf $i$. Then $H$ is of the form $h(x_i) \cdot (h'(x_i) + h''(x_j)) \ \cdots$, where $h, h', h''$ are step functions and $x_i$ does not appear in "$\cdots$". Define $F(\xi) := \max_{x\in\mathbb{R}} h(x)\cdot(h'(x)+\xi)$. Then $F$ is the upper envelope of $O(n)$ linear functions in the single variable $\xi$, and can be constructed in $\widetilde{O}(n)$ time by the dual of a planar convex hull algorithm [17]. We can eliminate the variable $x_i$ by replacing the $h(x_i)\cdot(h'(x_i)+h''(x_j))$ factor with $F(h''(x_j))$ (which is a step function in $x_j$ with $O(n)$ complexity). As a result, $H$ becomes a $(G-\{i\})$-function in $d-1$ variables. After $d$ iterations, the problem becomes trivial.

For the pseudo-forest case: We may assume the graph is connected, since we can maximize the parts of $H$ corresponding to different components separately. Pick a vertex $i$ that belongs to the unique cycle of $G$ (if exists). Then $G-\{i\}$ is a forest. By trying out all $O(n)$ different settings of $x_i$ (breakpoints of the step functions), the problem reduces to $O(n)$ instances of the forest case. ◀

▶ **Lemma 11.** *Let $H$ be a $G$-function with $O(n)$ complexity, where $G$ is a pseudo-forest. Given a set $S$ of $n$ boxes in $\mathbb{R}^d$ and a box $B_0$, we can compute the maximum of $H$ over $B_0 - \bigcup S$ in $\widetilde{O}(n^{d/2+1})$ time for any constant $d$.*

**Proof.** Apply Lemma 5 to the $O(n)$ $(d-2)$-flats through the boundaries of the orthants, together with the $O(n)$ $(d-1)$-flats $x_j = a$ for all breakpoints $a$ of the step functions appearing in $H$. This yields a partition of $B_0$ into cells.

Consider a cell $\Delta$. The number of $(d-2)$-flats intersecting $\Delta$ is $O(n/r^2)$, which can be made 0 by setting $r := \Theta(\sqrt{n})$. So, inside the cell $\Delta$, we see only 1-sided orthants, and their union simplifies to the complement of a box. In addition, the number of $(d-1)$-flats intersecting $\Delta$ is $O(n/r) = O(\sqrt{n})$; in other words, the breakpoints of the step functions in $H$ relevant to the cell $\Delta$ is $O(\sqrt{n})$. We can thus apply Lemma 10 to maximize $H$ over the cell $\Delta$ in $\widetilde{O}((\sqrt{n})^2)$ time. As the number of cells is $O(r^d) = O(n^{d/2})$, the total running time is $\widetilde{O}(n^{d/2}\cdot(\sqrt{n})^2)$. ◀

## 4.2 Algorithm

We now modify the proof of Theorem 7 to solve Problem 4 for the 2-sided orthant case:

▶ **Theorem 12.** *In the case when all input orthants are 2-sided, Problem 4 can be solved in $\widetilde{O}(n^{d/4+1})$ time for any constant even $d$.*

**Proof.** We maintain a $G$-function $H$. Initially, $H(x_1,\ldots,x_d) = (\sigma(x_1)+\sigma(x_2))(\sigma(x_3)+\sigma(x_4))\cdots(\sigma(x_{d-1})+\sigma(x_d))$, with $G$ being a matching with $d/2$ edges, where $\sigma(x)$ denotes the successor of $x$ among all $O(n)$ input coordinate values. We call an index $i$ *free* if $x_i$ appears exactly once in $H$ and is "unaltered" (i.e., $H$ is of the form $(\sigma(x_i)+h(x_\ell))\cdots$ where $x_i$ does not appear in "$\cdots$"). All indices are initially free. We maintain the following invariants: at any time, (i) $G$ is a pseudo-forest with at most $d/2$ edges, and (ii) for each component $T$ of $G$ which is a tree (not a 1-tree), $T$ has at least two free leaves.

In each iteration, we pick a free leaf $i$ in some component $T$ of $G$ which is a tree. As before, we rewrite the expression $E$ as a disjunction of $O(1)$ subexpressions, where in each subexpression, only two occurrences of $x_i$ remain – in a predicate of the form $[x_i \le f(x_j)]$, and another predicate of the form $[x_i \ge g(x_k)]$.

We branch off to maximize $H$ for each of these subexpressions separately. In such a subexpression, to eliminate the variable $x_i$ while maximizing $H$, we replace the two predicates $[x_i \le f(x_j)]$ and $[x_i \ge g(x_k)]$ with $[f(x_j) \ge g(x_k)]$, and replace $x_i$ with $f(x_j)$ in $H$ (since $x_i$ is free). Now, $i$ and $j$ are not free. Also, in the graph $G$, the unique edge $i\ell$ incident to $i$ is replaced by $j\ell$ (unless $j=\ell$). If $j$ is in the same component $T$ as $i$, then $T$ becomes a 1-tree; otherwise, two components are merged and the new component is either a tree with at least two free leaves, or a 1-tree. (See Figure 4.) So, the invariants are maintained.

We stop a branch when there are no free indices left. At the end, we get $O(1)$ subproblems, where in each subproblem, all components are 1-trees, and so the number of nodes is exactly equal to the number of edges, implying that the dimension is $d' \le d/2$. Now we can apply Lemma 11 to solve these subproblems in $\widetilde{O}(n^{d'/2+1})$ time. ◀

▶ **Corollary 13.** *Problem 4 can be solved in $\widetilde{O}(n^{(5d+4)/12})$ time for any constant even $d$.*

**Proof.** Following the proof of Corollary 8 but using Theorem 12 instead of Theorem 7 gives running time $\widetilde{O}(n^{d/3}\cdot(n^{1/3})^{d/4+1})$. ◀

Applying the above corollary in $2d$ dimensions, we finally obtain:

▶ **Corollary 14.** *Given $n$ points in $\mathbb{R}^d$ and a box $B_0$, we can compute the maximum-volume empty box inside $B_0$ in $\widetilde{O}(n^{(5d+2)/6})$ time for any constant $d$.*

## 5 Remarks

**On the 2D algorithm.** The $2^{O(\log^* n)}$ factor can be analyzed more precisely (an upper bound of $3^{\log^* n}$ can be shown with minor changes to the algorithm). A question remains whether the extra factor could be further lowered to inverse-Ackermann, or eliminated completely.

The previous algorithm by Aggarwal and Suri [3] used matrix searching techniques, namely, for finding row minima in certain types of partial Monge matrices. We are able to bypass such subroutines because we have focused our effort on solving the *decision problem* (due to the author's randomized optimization technique [9]). Generally, the row minima problem is equivalent to the computation of lower envelopes of pseudo-rays and pseudo-segments, not necessarily of constant complexity [12]. However, to solve the decision problem, we only need lower envelopes of pseudo-rays and pseudo-segments of constant complexity (formed by hyperbolas), for which there are simpler direct methods, as we have noted in Lemma 1. (Incidentally, the proof we gave for reducing Lemma 1(b) to (a) is essentially equivalent to Aggarwal and Klawe's reduction of row minima in double-staircase to staircase matrices [2]; a similar idea has also been used in dynamic data structures with "FIFO updates" [13].)

On the other hand, it should be possible to modify our approach to get improved *deterministic* algorithms for 2D largest empty rectangle, by solving the optimization problem directly and using known matrix searching subroutines [24], though details are more involved and the running time seems slightly worse than in our randomized algorithm.

It is theoretically possible to devise an optimal algorithm for Problem 1 without knowing the true complexity of the algorithm, since by a constant number of rounds of recursion in our method, the problem is reduced to subproblems of very small size (say, $\log\log\log\log n$), for which we can afford to explicitly build an optimal decision tree (this type of trick appeared before in the literature [25, 28]).

**On the higher-dimensional algorithms.** Our approach in higher dimensions works for maximizing the perimeter (sum of edge lengths) of the box as well. In fact, the algorithm for the simpler, largest empty *anchored* box problem should suffice here after doubling the dimension, since the required objective function here is $H_{\text{perim}}(x_1, \ldots, x_d) = x_1 + \cdots + x_d$, which is "similar" to $H_{\text{vol}}(x_1, \ldots, x_d) = x_1 \cdots x_d$.

For the largest empty anchored box problem, the $\widetilde{O}(n^{5d/12})$ time bound can be further improved to $\widetilde{O}(n^{(7d+6)/18})$, by building on the graph-theoretic ideas from Section 4, as we show in the full paper. Still further improvements of the exponent is likely possible, by working with $G$-functions for *hypergraphs* $G$, not just graphs, though improvement on the fraction 7/18 appears very tiny and requires $d$ to be a very large constant, and the algorithm becomes more complicated. For the largest empty box problem, we currently don't know how to improve the fraction 5/6, even using hypergraphs. It remains a fascinating question what the best fraction $\beta$ is for which the problem could be solved in $O(n^{\beta d + o(d)})$ time.

On the conditional lower bound side, another relevant question is whether Problem 3 or 4 remain $W[1]$-hard with respect to the parameter $d$ in the special case of 2-sided orthants.

## References


1 Pankaj K. Agarwal and Jeff Erickson. Geometric range searching and its relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, pages 1–56. AMS Press, 1999. URL: `http://jeffe.cs.illinois.edu/pubs/survey.html`.

2 Alok Aggarwal and Maria M. Klawe. Applications of generalized matrix searching to geometric algorithms. *Discret. Appl. Math.*, 27(1-2):3–23, 1990. `doi:10.1016/0166-218X(90)90124-U`.

3 Alok Aggarwal and Subhash Suri. Fast algorithms for computing the largest empty rectangle. In *Proc. 3rd Symposium on Computational Geometry (SoCG)*, pages 278–290, 1987. `doi:10.1145/41958.41988`.

4 Jonathan Backer and J. Mark Keil. The mono- and bichromatic empty rectangle and square problems in all dimensions. In *Proc. 9th Latin American Theoretical Informatics Symposium (LATIN)*, volume 6034 of *Lecture Notes in Computer Science*, pages 14–25. Springer, 2010. `doi:10.1007/978-3-642-12200-2_3`.

5 Jérémy Barbay, Timothy M. Chan, Gonzalo Navarro, and Pablo Pérez-Lantero. Maximum-weight planar boxes in $O(n^2)$ time (and better). *Inf. Process. Lett.*, 114(8):437–445, 2014. `doi:10.1016/j.ipl.2014.03.007`.

6 Michael A. Bender and Martin Farach-Colton. The LCA problem revisited. In Gaston H. Gonnet, Daniel Panario, and Alfredo Viola, editors, *Proc. 4th Latin American Symposium on Theoretical Informatics (LATIN)*, volume 1776, pages 88–94, 2000. `doi:10.1007/10719839_9`.

7 Jean-Daniel Boissonnat, Micha Sharir, Boaz Tagansky, and Mariette Yvinec. Voronoi diagrams in higher dimensions under certain polyhedral distance functions. *Discret. Comput. Geom.*, 19(4):485–519, 1998. `doi:10.1007/PL00009366`.

8 Karl Bringmann. An improved algorithm for Klee's measure problem on fat boxes. *Comput. Geom.*, 45(5-6):225–233, 2012. Preliminary version in SoCG'10. doi:10.1016/j.comgeo.2011.12.001.

9 Timothy M. Chan. Geometric applications of a randomized optimization technique. *Discret. Comput. Geom.*, 22(4):547–567, 1999. doi:10.1007/PL00009478.

10 Timothy M. Chan. A (slightly) faster algorithm for Klee's measure problem. *Comput. Geom.*, 43(3):243–250, 2010. Preliminary version in SoCG'08. doi:10.1016/j.comgeo.2009.01.007.

11 Timothy M. Chan. Klee's measure problem made easy. In *Proc. 54th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 410–419, 2013. doi:10.1109/FOCS.2013.51.

12 Timothy M. Chan. (Near-)linear-time randomized algorithms for row minima in Monge partial matrices and related problems. In *Proc. 32nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1465–1482, 2021. doi:10.1137/1.9781611976465.88.

13 Timothy M. Chan, John Hershberger, and Simon Pratt. Two approaches to building time-windowed geometric data structures. *Algorithmica*, 81(9):3519–3533, 2019. doi:10.1007/s00453-019-00588-3.

14 Timothy M. Chan and Patrick Lee. On constant factors in comparison-based geometric algorithms and data structures. *Discret. Comput. Geom.*, 53(3):489–513, 2015. doi:10.1007/s00454-015-9677-y.

15 Bernard Chazelle, Robert L. (Scot) Drysdale III, and D. T. Lee. Computing the largest empty rectangle. *SIAM J. Comput.*, 15(1):300–315, 1986. doi:10.1137/0215022.

16 Karen L. Daniels, Victor J. Milenkovic, and Dan Roth. Finding the largest area axis-parallel rectangle in a polygon. *Comput. Geom.*, 7:125–148, 1997. doi:10.1016/0925-7721(95)00041-0.

17 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008. URL: https://www.worldcat.org/oclc/227584184.

18 Adrian Dumitrescu and Minghui Jiang. On the largest empty axis-parallel box amidst $n$ points. *Algorithmica*, 66(2):225–248, 2013. doi:10.1007/s00453-012-9635-5.

19 Adrian Dumitrescu and Minghui Jiang. Perfect vector sets, properly overlapping partitions, and largest empty box. *CoRR*, abs/1608.06874, 2016. arXiv:1608.06874.

20 Adrian Dumitrescu and Minghui Jiang. On the number of maximum empty boxes amidst $n$ points. *Discret. Comput. Geom.*, 59(3):742–756, 2018. Preliminary version in SoCG'16. doi:10.1007/s00454-017-9871-1.

21 Adrian Dumitrescu, Joseph S. B. Mitchell, and Micha Sharir. Binary space partitions for axis-parallel segments, rectangles, and hyperrectangles. *Discret. Comput. Geom.*, 31(2):207–227, 2004. doi:10.1007/s00454-003-0729-3.

22 Harold N. Gabow, Jon Louis Bentley, and Robert Endre Tarjan. Scaling and related techniques for geometry problems. In *Proc. 16th ACM Symposium on Theory of Computing (STOC)*, pages 135–143, 1984. doi:10.1145/800057.808675.

23 Panos Giannopoulos, Christian Knauer, Magnus Wahlström, and Daniel Werner. Hardness of discrepancy computation and epsilon-net verification in high dimension. *J. Complex.*, 28(2):162–176, 2012. doi:10.1016/j.jco.2011.09.001.

24 Maria M. Klawe and Daniel J. Kleitman. An almost linear time algorithm for generalized matrix searching. *SIAM J. Discret. Math.*, 3(1):81–97, 1990. doi:10.1137/0403009.

25 Lawrence L. Larmore. An optimal algorithm with unknown time complexity for convex matrix searching. *Inf. Process. Lett.*, 36(3):147–151, 1990. doi:10.1016/0020-0190(90)90084-B.

26 Amnon Naamad, D. T. Lee, and Wen-Lian Hsu. On the maximum empty rectangle problem. *Discret. Appl. Math.*, 8(3):267–277, 1984. doi:10.1016/0166-218X(84)90124-0.

27 Mark H. Overmars and Chee-Keng Yap. New upper bounds in Klee's measure problem. *SIAM J. Comput.*, 20(6):1034–1045, 1991. doi:10.1137/0220065.

28 Seth Pettie and Vijaya Ramachandran. An optimal minimum spanning tree algorithm. *J. ACM*, 49(1):16–34, 2002. doi:10.1145/505241.505243.

**29** Micha Sharir and Pankaj K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications.* Cambridge University Press, 1995.

**30** Mario Ullrich and Jan Vybíral. An upper bound on the minimal dispersion. *J. Complex.*, 45:120–126, 2018. doi:10.1016/j.jco.2017.11.003.

**31** Jean Vuillemin. A unifying look at data structures. *Commun. ACM*, 23(4):229–239, 1980. doi:10.1145/358841.358852.