Fast Approximation Algorithms for Piercing Boxes by Points*

Pankaj K. Agarwal† Sariel Har-Peled‡ Rahul Raychaudhury§
Stavros Sintos¶

Abstract

Let $\mathcal{B} = \{\mathsf{b}_1, \ldots, \mathsf{b}_n\}$ be a set of n axis-aligned boxes in $\mathbb{R}^d$ where $d \geq 2$ is a constant. The *piercing problem* is to compute a smallest set of points $\mathcal{N} \subset \mathbb{R}^d$ that hits every box in $\mathcal{B}$, i.e., $\mathcal{N} \cap \mathsf{b}_i \neq \emptyset$, for $i = 1, \ldots, n$. The problem is known to be NP-Hard. Let $\mathfrak{p} := \mathfrak{p}(\mathcal{B})$, the *piercing number* be the minimum size of a piercing set of $\mathcal{B}$. We first present a randomized $O(\log\log \mathfrak{p})$-approximation algorithm with expected running time $O(n^{d/2}\operatorname{polylog}(n))$. Next, we show that the expected running time can be improved to near-linear using a sampling-based technique, if $\mathfrak{p} = O(n^{1/(d-1)})$. Specifically, in the plane, the improved running time is $O(n \log \mathfrak{p})$, assuming $\mathfrak{p} < n/\log^{\Omega(1)} n$. Finally, we study the dynamic version of the piercing problem where boxes can be inserted or deleted. For boxes in $\mathbb{R}^2$, we obtain a randomized $O(\log\log \mathfrak{p})$-approximation algorithm with $O(n^{1/2}\operatorname{polylog}(n))$ amortized expected update time for insertion or deletion of boxes. For squares in $\mathbb{R}^2$, the update time can be improved to $O(n^{1/3}\operatorname{polylog}(n))$.

Our algorithms are based on the multiplicative weight-update (MWU) method and require the construction of a weak ε-net for a point set with respect to boxes. A key idea of our work is to exploit the duality between the piercing set and independent set (for boxes) to speed up our MWU. We also present a simpler and slightly more efficient algorithm for constructing a weak ε-net than in [Ezr10], which is of independent interest. Our approach also yields a simpler algorithm for constructing (regular) ε-nets with respect to boxes for $d = 2, 3$.

1 Introduction

Problem statement A ***box*** is an axis-aligned box in $\mathbb{R}^d$ of the form $\prod_{i=1}^{d}[\alpha_i, \beta_i]$. A one dimensional box is an ***interval***, and a two dimensional box is a ***rectangle***. Let $\mathcal{B} = \{\mathsf{b}_1, \ldots, \mathsf{b}_n\}$ be a set of n boxes in $\mathbb{R}^d$. A subset $\mathcal{N} \subset \mathbb{R}^d$ is a *piercing set* of $\mathcal{B}$ if $\mathcal{N} \cap R \neq \emptyset$ for every box $\mathsf{b} \in \mathcal{B}$. The *piercing problem* asks to find a piercing set of $\mathcal{B}$ of the smallest size, which we denote by $\mathfrak{p} := \mathfrak{p}(\mathcal{B})$ and call the ***piercing number*** of $\mathcal{B}$. Although the piercing problem can be defined over arbitrary geometric objects such as disks and halfspaces, here we focus on boxes. The piercing problem is a fundamental problem in computational geometry and has applications in facility location, sensor networks, etc.

*A full version of this paper is available on the arXiv [AHRS23].

†Department of Computer Science, Duke University, Durham NC 27708. Work by Pankaj Agarwal was partially supported by NSF grants IIS-18-14493, CCF-20-07556, and CCF-22-23870.

‡Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL, 61801, USA; sariel@illinois.edu; http://sarielhp.org/. Work by Sariel Har-Peled was partially supported by a NSF AF award CCF-1907400 and CCF-2317241.

§Department of Computer Science, Duke University, Durham NC 27708.

¶Department of Computer Science, University of Illinois at Chicago, Chicago IL 60607.

The piercing problem is closely related to the classical *geometric hitting-set* problem: Given a (geometric) range space $\Sigma = (\mathcal{X}, \mathcal{R})$, where $\mathcal{X}$ is a (finite or infinite) set of points in $\mathbb{R}^d$ and $\mathcal{R} \subseteq 2^{\mathcal{X}}$ is a finite family of ranges, defined by simply-shaped regions such as rectangles, balls, hyperplanes etc. That is, each range in Σ is of the form $R \cap \mathcal{X}$, where $R \in \mathcal{R}$ is a geometric shape (strictly speaking, $\Sigma = (\mathcal{X}, \{R \cap \mathcal{X} \mid R \in \mathcal{R}\})$ but with a slight abuse of notation we will use $(\mathcal{X}, \mathcal{R})$ to denote the range space). A subset $H \subseteq \mathcal{X}$ is a *hitting set* of Σ if $H \cap R \neq \emptyset$ for all $R \in \mathcal{R}$. The hitting-set problem asks for computing a minimum-size hitting set of Σ. The piercing problem is a special case of the hitting-set problem in which $\Sigma = (\mathbb{R}^d, \mathcal{B})$. Instead of letting the set of points be the entire $\mathbb{R}^d$, we can choose the set of points to be the set of vertices in $\mathcal{A}(\mathcal{B})$, the arrangement of $\mathcal{B}$,[1] and the range space is now $\Sigma = \big(\mathcal{V}, \{\mathsf{b} \cap \mathcal{V} \mid \mathsf{b} \in \mathcal{B}\}\big)$, where $\mathcal{V} := \mathcal{V}(\mathcal{B})$ is the set of vertices in $\mathcal{A}(\mathcal{B})$. It is easily seen that $\mathcal{B}$ has a piercing set of size p if and only if Σ has a hitting set of size p. Hitting set for general range spaces was listed as one of the original NP-complete problems [Kar72]. Furthermore, the box piercing problem is NP-Complete even in 2D [FPT81], so our goal is to develop an efficient approximation algorithm for the piercing problem.

In many applications, especially those dealing with large data sets, simply a polynomial-time algorithm is not enough, and one desires an algorithm whose running time is near-linear in $|\mathcal{B}|$. In principle, the classical greedy algorithm can be applied to the range space $(\mathcal{V}, \mathcal{B})$, but $|\mathcal{V}| = O(n^d)$, so it will not lead to a fast algorithm. Intuitively, due to the unconstrained choice of points with which to pierce boxes of $\mathcal{B}$, the piercing problem seems easier than the geometric hitting-set problem and should admit faster solutions and better approximations. In this paper, we make progress towards this goal for a set of boxes in both static and dynamic settings.

Related work The well-known shifting technique by Hochbaum and Maass [HM85] can be used to obtain a PTAS when $\mathcal{B}$ comprises of unit-squares or near-equal-sized fat objects in any fixed dimension. Efrat *et al.* [EKNS97] designed an $O(1)$-approximation algorithm for a set of arbitrary "fat" objects that runs in near-linear time in 2d and 3d. Chan [Cha03] gave a separator-based PTAS for arbitrary sized fat objects, with running time $O(n^{\varepsilon^{-d}})$. Chan and Mahmood [CM05] later gave a PTAS for a set of boxes with arbitrary width but unit height. All of the above results consider a restricted setting of boxes. Surprisingly, little is known about the piercing problem for a set of arbitrary axis-aligned boxes in $\mathbb{R}^d$. By running a greedy algorithm or its variants based on a multiplicative weight update (MWU) method, an $O(\log p)$-approximation algorithm with running time roughly $O(n^d)$ can be obtained for the box-piercing problem in $\mathbb{R}^d$. Using the weak ε-net result by Ezra [Ezr10] (see also [AES09]), the approximation factor improves to $O(\log \log p)$. An interesting question is what is the smallest piercing set one can find in near-linear time. Nielsen [Nie00] presented an $O(\log^{d-1} p)$-approximation divide-and-conquer algorithm that runs in $O(n \log^{d-1} n)$ time. We are unaware of any near-linear time algorithm even with $O(\log p)$-approximation ratio for $d \geq 3$. We note that the piercing problem has also been studied in discrete and convex geometry, where the goal is to bound the size of the piercing set for a family of objects with certain properties. See e.g. [AK92, CSZ18].

We conclude this discussion by mentioning that there has been much work on the geometric hitting-set problem. For a range space $\Sigma = (\mathcal{X}, \mathcal{R})$ and weight function $\omega : \mathcal{X} \to \mathbb{R}_{\geq 0}$, a subset $\mathcal{N} \subseteq \mathcal{X}$ is an ε-net if for any $R \in \mathcal{R}$ with $\omega(R) \geq \varepsilon \omega(\mathcal{X})$, we have $R \cap \mathcal{N} \neq \emptyset$. The multiplicative weight update (MWU) method assigns a weight to each point so that every range in $\mathcal{R}$ becomes

[1] The arrangement of $\mathcal{B}$, denoted by $\mathcal{A}(\mathcal{B})$, is the partition of $\mathbb{R}^d$ into maximal connected cells so that all points within each cell are in the interior/boundary of the same set of rectangles. It is well known that $\mathcal{A}(\mathcal{B})$ has $O(n^d)$ complexity [Ede87].

"$1/e\rho$-heavy" and then one simply chooses a $1/e\rho$-net. Using the MWU method and results on ε-nets, Brönnimann and Goodrich [BG95] presented a polynomial-time $O(\log \rho)$-approximation algorithm for the hitting-set problem for range spaces with finite VC-dimension [VC71]. Later Agarwal and Pan [AP20] presented a more efficient implementation of the MWU method for geometric range spaces. Their approach led to $O((|X| + |\mathcal{R}|)\,\mathrm{polylog}(n))$ algorithm with $O(\log \rho)$-approximation for many cases including a set of rectangles in $\mathbb{R}^d$ (see also [BMR18, CH20]). But it does not lead to a near-linear time algorithm for the piercing problem because $|X| = O(n^d)$ in this case. In another line of work, polynomial-time approximation algorithms for hitting sets based on local search have also been proposed [MR10].

The MWU algorithm essentially solves and rounds the LP associated with the hitting-set problem, see [Har11, Chapter 6]. Thus, the approximability of the problem is strongly connected to the integrality gap of the LP. For the hitting-set problem of points with boxes for $d \leq 3$, Aronov *et al.* [AES09] showed a rounding scheme with integrality gap $O(\log \log \rho)$. Furthermore, Ezra [Ezr10] showed the same gap holds in higher dimensions if one is allowed to use any point to do the piercing. Surprisingly, Pach and Tardos [PT11] showed that this integrality gap is tight. While a better approximation than the integrality gap can be obtained in a few cases [CH12, MR10], these algorithms require a fundamentally different approach. Thus, a major open problem is to obtain an $O(1)$-approximation algorithm for the box-piercing problem.

Recently, Agarwal *et al.* [ACS+22] initiated a study of dynamic algorithms for geometric instances of set-cover and hitting-set. Here the focus is on maintaining an approximately optimal hitting-set (resp. set-cover) of a dynamically evolving instance, where in each step a new object may be added or deleted. They introduced fully dynamic sublinear time hitting-set algorithms for squares and intervals. These results were improved and generalized in [CHSX22, CH21]. Khan *et al.* [KLR+23] proposed a dynamic data structure for maintaining a $O(\mathrm{polylog}(n))$-factor approximation of the optimal hitting-set for boxes under restricted settings, but no algorithm with sublinear update time for the general setting is known.

Our results In this paper, we design an efficient $O(\log \log \rho)$-approximation algorithm for the box-piercing problem. Let $\mathcal{B}$ be a set of n boxes in $\mathbb{R}^d$. A naive way to get an $O(\log \log \rho)$-approximation is by using aforementioned MWU [AP20, BG95] based hitting-set algorithms on the range space $(\mathcal{V}, \mathcal{B})$, and use Ezra's [Ezr10] algorithm for computing a weak ε-net instead of computing a strong ε-net. While this naive approach gives the desired approximation, it runs in $\Omega(n^d)$ time. We present the following results.

(A) A new MWU and its fast implementation. We present two algorithms in Section 2. The first is essentially the one by Agarwal-Pan that computes a hitting set of the range space $(\mathcal{V}, \mathcal{B})$ of size $O(\rho \log \log \rho)$. We show that it can be implemented in $O(n^{(d+1)/2} \log^3 n)$ expected time (Theorem 2.1). To achieve the desired running time, we need a data structure to perform all the required operations on $\mathcal{V}$ without ever explicitly constructing it. We present such a data structure, which exploits the properties of the partitioning technique by Overmars and Yap [OY91]. For brevity, we include the details of this data structure in the full version of the paper [AHRS23].

Our main result is, however, a different MWU algorithm tailored for boxes with $O(n^{d/2} \log^{2d+3} n)$ expected running time (Theorem 2.2). It exploits the duality between the piercing-set problem and the independent-set problem, along with fast approximation algorithms for these two problems. The basic idea is to use the aforementioned approximation algorithm to find a large set of independent boxes among the light boxes identified by the MWU algorithm in a round. If the algorithm does not find such an independent set, then we can use a simple piercing-set algorithm to compute a desired piercing set. Otherwise we double the weight of the boxes in the

independent set. The idea of duality and approximation to speedup the MWU is critical to get a near linear running time in the plane. As far as we are aware, the idea of quickly rounding the dual problem and using it to speedup the primal algorithm, in the context of set-cover/hitting-set/piercing-set for rectangles, was not used before.

(B) Piercing, clustering, and multi-round algorithms. We show that the natural algorithm of first computing a piercing set $\mathcal{P}_0$ for a random sample of input boxes, and then piercing the input boxes that are not pierced by $\mathcal{P}_0$ leads to an efficient piercing algorithm. This algorithm can be extended to run an arbitrary number of rounds. For clustering this idea was described by Har-Peled [Har04] (but the idea is much older see e.g. [GRS98]). In particular, if the algorithm runs ζ rounds, then it needs to compute piercing sets ζ times for sets of boxes of size (roughly) $\mathfrak{p}^{1-1/\zeta} n^{1/\zeta}$. By picking ζ a sufficiently large constant, we obtain an $O(\zeta \log\log \mathfrak{p})$ approximation algorithm that runs in near-linear expected time, provided that $\mathfrak{p} = O(n^{1/(d-1)})$. See Theorem 3.1 and Corollary 3.1. For $d = 2$, this leads to the striking result that one can obtain an $O(\log\log \mathfrak{p})$-approximation algorithm with $O(n \log \mathfrak{p})$ expected running time provided that $\mathfrak{p} = O(n/\log^{15} n)$, see Corollary 3.2.

(C) Dynamic Algorithms for Piercing. We consider the piercing problem for a set $\mathcal{B}$ of boxes in $\mathbb{R}^2$ in the *dynamic* setting, i.e., at each step a new box is inserted into or deleted from $\mathcal{B}$. Our goal is to maintain an (approximately) optimal solution of the current set. We implement a dynamic version of the multi-round sampling based algorithm in Section 3, and attain a randomized Monte Carlo $O(\log\log \mathfrak{p})$-approximation algorithm with $O(n^{1/2} \operatorname{polylog}(n))$ amortized expected update time. The update time improves to $O(n^{1/3} \operatorname{polylog}(n))$ if $\mathcal{B}$ is a set of squares in $\mathbb{R}^2$. In principle, our approach extends to higher dimensions but currently we face a few technical hurdles in its efficient implementation (Section 4).

(D) New constructions of (weak) ε-nets. We present a simpler and more efficient algorithm for constructing a weak ε-net then the one in [Ezr10]. In particular, given a set $\mathcal{P}$ of n points in $\mathbb{R}^d$, a weight function $w : \mathcal{P} \to \mathbb{R}_{\geq 0}$, and a parameter $\varepsilon \in (0, 1)$, it computes a set $\mathcal{N} \subseteq \mathbb{R}^d$ of $O(\varepsilon^{-1} \log\log \varepsilon^{-1})$ points (not necessarily a subset of $\mathcal{P}$) in $O(n + \varepsilon^{-1} \log^{O(d^2)} \varepsilon^{-1})$ expected time such that for any box b with $\omega(\mathsf{b} \cap \mathcal{P}) \geq \varepsilon \omega(\mathcal{P})$, we have $\mathsf{b} \cap \mathcal{N} \neq \emptyset$. The running time can be improved to $O((n + m + \varepsilon^{-1}) \log^d \varepsilon^{-1})$ if we wish to guarantee above property for a given set of m boxes, which is the case in our setting. Our technique also gives an efficient algorithm for constructing an ε-net of size $O(\varepsilon^{-1} \log\log \varepsilon^{-1})$ for rectangles in $d = 2, 3$, which is somewhat simpler than the one in [AES09]. We include the details of the construction in the full version of the paper [AHRS23].

Building on the earlier work [AES09, Ezr10, PT11], our work brings the key insights of the results to the forefront. Our main new ingredient is using poly-logarithmic different grids to "capture" the distribution of the point sets. This idea appears in the work of Pach and Tardos [PT11] in the construction of the lower-bound, so it is not surprising that it is useful in the construction itself (i.e., upper-bound).

2 Piercing set via multiplicative weight update (MWU) algorithm

We describe two multiplicative-weight-update (MWU) based $O(\log\log \mathfrak{p})$-approximation algorithms for computing a piercing set for $\mathcal{B}$. Let $\mathcal{V} = \mathcal{V}(\mathcal{B})$ denote the set of vertices in the arrangement $\mathcal{A}(\mathcal{B})$. As already mentioned, there is always an optimal piercing set $\mathcal{N} \subseteq \mathcal{V}$ of $\mathcal{B}$. The basic idea is to reweight the points of $\mathcal{V}$ such that all boxes become *heavy*. Specifically, we compute a weight function $\omega : \mathcal{V} \to \mathbb{Z}^+$, such that for any box $\mathsf{b} \in \mathcal{B}$, we have $\omega(\mathsf{b} \cap \mathcal{V}) = \sum_{p \in \mathsf{b} \cap \mathcal{V}} \omega(p) \geq \omega(\mathcal{V})/2\mathfrak{p}e$.

Dim	preprocessing	Std. operations	**sample**
$d = 1$	$O(n \log n)$	$O(\log n)$	$O(\log n)$
$d = 2$	$O(n^{3/2} \log n)$	$O(\sqrt{n} \log n)$	$O(\log n)$
$d > 2$	$O(n^{(d+1)/2} \log n)$	$O(n^{(d-1)/2} \log n)$	$O(\log n)$

Figure 1: We describe a data structure for maintaining (implicitly) the vertices of an arrangement of boxes, under operations **weight**, **double**, **halve**, **insert**, **delete**, see Definition 2.1.

We then compute a weak $\frac{1}{2pe}$-net $\mathcal{N}$ of the range space $(\mathcal{V}, \mathcal{B})$ with respect to the above weight function. By definition $\mathcal{N}$ is a piercing set of $\mathcal{B}$. The two algorithms differ in how the weights are updated. Before describing the algorithms, we give the specifications of a data structure used by both algorithms.

For a multi-set $\mathcal{S} \subseteq \mathcal{B}$ of boxes, and a point $p \in \mathbb{R}^d$, let $\mathcal{S} \sqcap p = \{\mathsf{b} \in \mathcal{S} \mid p \in \mathsf{b}\}$ be the multi-set of all boxes in $\mathcal{S}$ containing p, let

$$w_{\mathcal{S}}(p) := 2^{|\mathcal{S} \sqcap p|} \tag{2.1}$$

be the ***doubling weight*** of p. For a finite set of points $X \subseteq \mathbb{R}^d$, let $w_{\mathcal{S}}(X) := \sum_{x \in X} w_{\mathcal{S}}(p)$.

Definition 2.1. (Implicit arrangement data structure) *Let $\mathcal{B}$ be a set of axis-aligned boxes in $\mathbb{R}^d$ (known in advance). Let $\mathcal{C} \subseteq \mathcal{B}$ be a set of* **active** *boxes (initially empty) and let $\mathcal{V}(\mathcal{C})$ be the set of vertices of the arrangement $\mathcal{A}(\mathcal{C})$. Let $\mathcal{S} \subseteq \mathcal{B}$ be a multi-set of* **update** *boxes (initially empty). $\mathcal{S}$ induces a weight function on the vertices in $\mathcal{V}(\mathcal{C})$ using Eq. (2.1). Here, we require a data structure that supports the following operations:*

(I) **weight**(b)*: given a box b, compute $w_{\mathcal{S}}(\mathcal{V}(\mathcal{C}) \cap \mathsf{b})$.*

(II) **double**(b)*: given a box $\mathsf{b} \in \mathcal{B}$, adds a copy of b to the multi-set of update boxes $\mathcal{S}$.*

(III) **halve**(b)*: given a box $\mathsf{b} \in \mathcal{B}$, removes a copy of b from the multi-set of update boxes $\mathcal{S}$.*

(IV) **sample***: returns a random point $p \in \mathcal{V}(\mathcal{C})$, with probability $w_{\mathcal{S}}(p)/w_{\mathcal{S}}(\mathcal{V}(\mathcal{C}))$.*

(V) **insert**(b)*: inserts b into the set of active boxes $\mathcal{C}$.*

(VI) **delete**(b)*: removes b from the set of active boxes $\mathcal{C}$.*

Figure 1 summarizes the performance of the data structure. We defer the implementation of the data structure to the full version [AHRS23]. Omitting the details, we obtain the following lemma:

Lemma 2.1. *Let $\mathcal{B}$ be a set of n axis-aligned rectangles in $\mathbb{R}^d$. A data structure can be constructed in $O(n^{(d+1)/2} \log n)$ time that supports every operation specified in Definition 2.1 in $O(n^{(d-1)/2} \log n)$ time except for* **sample***, which takes only $O(\log n)$ time.*

Note that the **halve** and **delete** operations are not used by the basic algorithm.

2.1 Basic MWU algorithm The algorithm is a small variant of the Agarwal-Pan algorithm [AP20]. The algorithm performs an exponential search on a value k, and it stops when $k \geq p$ and

$k \leq 2\mathfrak{p}$. Specifically, in the ith stage, k is set to 2^i. We next describe such a stage for a fixed value of k (and i).

In the beginning of the stage, $\omega(p) = 1$ for all $p \in \mathcal{V}$. Let $\varepsilon = \frac{2}{3k}$. A box b is ***ε-light*** if $\omega(\mathcal{V} \cap \mathsf{b}) < \varepsilon\omega(\mathcal{V})$, and ***$\varepsilon$-heavy*** otherwise. The algorithm proceeds in rounds. In each round, it scans all the boxes in $\mathcal{B}$ trying to find an ε-light box. If an ε-light box b is found, the algorithm performs a ***doubling*** operation on its points. That is, it doubles the weight of each of the points in $\mathcal{V} \cap \mathsf{b}$. The algorithm performs the doubling on b repeatedly until b becomes ε-heavy (in relation to the updated weight $\omega(\mathcal{V})$). The algorithm then resumes the scan for ε-light boxes (over the remaining boxes). Importantly, the algorithm never revisits a box during a round (thus a box that becomes heavy during a round might become light again in that round). Let $\ell = \lfloor 1/\varepsilon \rfloor \leq 2k$. If ℓ doubling operations are performed in a round, the algorithm aborts the round, and proceeds to the next round. If a round was completed without ℓ weight-doubling being performed, the algorithm computes a weak (ε/e)-net $\mathcal{N}$ of $(\mathcal{V}, \mathcal{B})$ using the algorithm described in Section 6 of [AHRS23].

Finally, if the number of rounds exceeds $\tau = c \ln(|\mathcal{V}|/k)$ at any stage, where $c > 0$ is a suitably large constant, the algorithm decides the guess for k as too small, doubles the value of k, and continues to the next stage, till success.

In order to implement the above algorithm, we use the implicit arrangement data structure from Definition 2.1. At the beginning of each stage, we build an instance of the data structure on $\mathcal{B}$ and add all the boxes to the active set using the **insert** operation. The doubling step with a box b is performed using **double**(b) and the ε-lightness of a box is tested using the **weight**(b) operation.

Analysis The following lemma proves the correctness of the algorithm.

LEMMA 2.2. ([AP20]) *If $k \in [\mathfrak{p}/2, \mathfrak{p}]$, then the algorithm returns a piercing set for $\mathcal{B}$, of size $O(\mathfrak{p} \log\log \mathfrak{p})$, where $\mathfrak{p} = \mathfrak{p}(\mathcal{B})$ is the size of the optimal piercing set of $\mathcal{B}$.*

Proof. This claim is well known [AP20, Lemma 3.1], but we include a proof for the sake of completeness.

Initially, the weight W_0 of all the vertices of $\mathcal{V}$ is $m = |\mathcal{V}| = O(n^d)$. Let W_i be the weight of $\mathcal{V}$ after the ith doubling operation, and observe that

$$\mathfrak{p} 2^{\lfloor i/\mathfrak{p} \rfloor} \leq W_i \leq (1+\varepsilon)^i W_0 \leq m \exp(\varepsilon i).$$

The lower bound follows as every doubling operation must double the weight of one of the $\mathfrak{p}$ points in the optimal piercing set (and to minimize this quantity, this happens in a round robin fashion). The upper bound follows readily from the ε-lightness of the box being doubled. Taking $i = t\mathfrak{p}$, and taking the log of both sides (in base 2), we have

$$\log \mathfrak{p} + t \leq \log m + \varepsilon t \mathfrak{p} \log e \leq \log m + \frac{2}{3k} t\mathfrak{p} \cdot 1.45 \leq \log m + 0.97 t \frac{\mathfrak{p}}{k}.$$

Assuming $k \geq \mathfrak{p}$, this readily implies that $t = O(\log \frac{m}{k})$. Namely, the algorithm performs at most $\tau = O(k \log(m/k))$ doubling operations in a stage, if the guess $k \geq \mathfrak{p}$.

Since every round performs exactly $\ell = \lfloor 1/\varepsilon \rfloor = \lfloor 3k/2 \rfloor \geq k$ doubling operations in each round, except the last one, it follows that the algorithm performs at most $\tau/k = O(\log \frac{m}{k})$ rounds.

So consider the last round. Assume the weight of $\mathcal{V}$ at the start of the round was W. At the end of the round, the total weight of $\mathcal{V}$ is at most $(1+\varepsilon)^\ell W \leq \exp(\varepsilon \lfloor 1/\varepsilon \rfloor) W \leq eW$. Every box during this round, must have been heavy (at least for a little while), which implies that it had weight $\geq \varepsilon W$, as weights only increase during the algorithm execution. This implies that all the

boxes are ε/e-heavy at the end of the round. Thus, the weak ε/e-net the algorithm computes must pierce all the boxes of $\mathcal{B}$. ∎

The above lemma shows that the algorithm succeeds if it stops, and it must stop in a stage if k is sufficiently large.

THEOREM 2.1. *Let $\mathcal{B}$ be set of n axis-aligned boxes in $\mathbb{R}^d$, for some fixed $d \geq 2$, and let $\mathfrak{p} = \mathfrak{p}(\mathcal{B})$ be the piercing number of $\mathcal{B}$. The above algorithm computes, in $O(n^{(d+1)/2} \log^3 n)$ expected time, a piercing set of $\mathcal{B}$ of size $O(\mathfrak{p} \log\log \mathfrak{p})$.*

Proof. The above implies that if the algorithm outputs a piercing set, then it is of the desired size (it might be that the algorithm stops at an earlier stage than expected with a guess of k that is smaller than $\mathfrak{p}$). Thus, it must be that if the guess for the value of k is too small, then the algorithm double weights in vain, and performs too many rounds (i.e., their number exceeds τ), and the algorithm continues to the next stage.

Let $h = \lceil \log \mathfrak{p} \rceil$ and $m = |\mathcal{V}|$. Overall, the algorithm performs at most

$$\sum_{i=1}^{h} O(2^i \log m) = O(\mathfrak{p} \log m) = O(\mathfrak{p} d \log n) \tag{2.2}$$

doubling operations (i.e., **double**). Every round requires n **weight** operations. There are $O(\log m)$ rounds in each stage, and there are h stages. We conclude that the algorithm performs

$$O(n \log m \log \mathfrak{p}) = O(n \log^2 n)$$

weight operations. Thus, the overall running time (including preprocessing and activating all boxes), is

$$O(n^{(d+1)/2} \log n + n \cdot n^{(d-1)/2} \log n \cdot \log^2 n) = O(n^{(d+1)/2} \log^3 n).$$

Finally, the algorithm in [AHRS23] for computing a weak ε-net first chooses a random sample $\mathcal{Q}$ of $\mathcal{V}$ of size $O(\varepsilon^{-1} \log \varepsilon^{-1})$ and then computes a weak $\frac{\varepsilon}{2}$-net of $(\mathcal{Q}, \mathcal{B})$. Using **sample**, $\mathcal{Q}$ can be computed in $O(|\mathcal{Q}| \log n)$ time. Combining this with Lemma 6.12 in [AHRS23], we conclude that a ε/e-net of $(\mathcal{V}, \mathcal{B})$ of size $O(\mathfrak{p} \log\log \mathfrak{p})$ can be computed in $O\left((n + \varepsilon^{-1}) \log^d n\right)$ time, which is dominated by the data structure's overall running time. ∎

2.2 An improved MWU algorithm We now present an alternative algorithm for piercing that exploits LP duality. Operationally, the improved algorithm differs from the basic one in a few ways. The algorithm does not use the implicit arrangement data structure to maintain the weights on $\mathcal{V}$ directly. Moreover, instead of doubling the weight of light boxes one at a time, the algorithm performs "batch doubling", i.e., it doubles the weights of a collection of boxes at the same time. The algorithm also does not compute the weights of boxes exactly. Instead, it approximates the weights using a suitable sample. This sample is periodically recomputed by a process we call "batch sampling" that uses the implicit arrangement data structure in $\mathbb{R}^{(d-1)}$. Before describing the improved algorithm, we explore the duality between piercing sets and independent sets, and discuss the details of batch sampling.

$$\begin{aligned} \mathfrak{p}^*(\mathcal{B}) = \min & \sum_{v \in \mathcal{V}} x_v \\ \forall \mathsf{b} \in \mathcal{B} \qquad & \sum_{v \in \mathsf{b} \cap \mathcal{V}} x_v \geq 1 \\ \forall v \in \mathcal{V} \qquad & x_v \geq 0. \end{aligned}$$

The LP for the fractional piercing number.

$$\begin{aligned} \mathfrak{i}^*(\mathcal{B}) = \max & \sum_{\mathsf{b} \in \mathcal{B}} y_{\mathsf{b}} \\ \forall v \in \mathcal{V} \qquad & \sum_{\mathsf{b} \in \mathcal{B} : v \in \mathsf{b}} y_{\mathsf{b}} \leq 1 \\ \forall \mathsf{b} \in \mathcal{B} \qquad & y_{\mathsf{b}} \geq 0. \end{aligned}$$

The dual LP for the fractional independence number.

Figure 2: LPs for the piercing and Independence problems. By duality, we have $\mathfrak{p}^*(\mathcal{B}) = \mathfrak{i}^*(\mathcal{B})$.

LP duality, piercing and independence numbers Let $\mathcal{B}$ be a set of boxes in $\mathbb{R}^d$, and let $\mathcal{V}$ be the set of vertices of the arrangement $\mathcal{A}(\mathcal{B})$. A subset $\mathcal{X} \subseteq \mathcal{B}$ is an ***independent set*** if no pair of boxes of $\mathcal{X}$ intersect. The ***independence number*** of $\mathcal{B}$, denoted by $\mathfrak{i}(\mathcal{B})$, is the size of the largest independent set $\mathcal{X} \subseteq \mathcal{B}$. Observe that $\mathfrak{i}(\mathcal{B}) \leq \mathfrak{p}(\mathcal{B})$, as each box in an independent set must be pierced, but no point can pierce more than one box in the independent set. Let $\mathfrak{p}^* = \mathfrak{p}^*(\mathcal{B})$ denote the fractional piercing number of $\mathcal{B}$, i.e., this is the minimum piercing number of $\mathcal{B}$ when solving the associated LP, see Figure 2. The dual LP, is the fractional independence LP – it computes (fractionally) a maximum independent set of boxes in $\mathcal{B}$. The value of this LP $\mathfrak{i}^* := \mathfrak{i}^*(\mathcal{B})$ is the ***fractional Independence number*** of $\mathcal{B}$. By LP duality $\mathfrak{p}^* = \mathfrak{i}^*$, see Figure 2. It is known, and also implied by Lemma 2.2, that $\mathfrak{p}^* \leq \mathfrak{p} \leq c_1 \mathfrak{p}^* \log\log \mathfrak{p}^*$, for some constant $c_1 > 0$. This implies that $\mathfrak{p}^* \geq \mathfrak{p}/(c_1 \log\log \mathfrak{p})$.

We need the following standard algorithms for computing independent set and piercing set of boxes. Better results are known [CC09, GKM+22, Mit21], but they are unnecessary for our purposes.

LEMMA 2.3. ([AvKS97]) *Let $\mathcal{B}$ be a set of n boxes in $\mathbb{R}^d$. An independent subset of boxes of $\mathcal{B}$ of size $\Omega(\mathfrak{i}^*/\log^{d-1} n)$ can be computed in $O(n \log^{d-1} n)$ time, where $\mathfrak{i}^* = \mathfrak{i}^*(\mathcal{B})$ is the fractional independence number of $\mathcal{B}$.*

Proof. This algorithm is well known and we sketch it here for completeness – it works by induction on the dimension. For $d = 1$, a greedy algorithm picking the first interval with the minima right endpoint computes the optimal independent set — the running time of this algorithm is $O(n)$ after sorting. It is straightforward to verify that one can assume that the LP solution in this case is integral, and thus $\mathfrak{i}^* = \mathfrak{i}(\mathcal{B})$.

For $d = 2$, compute the vertical line such that its x-coordinate is the median of the x-coordinates of the endpoints of the boxes. Compute the optimal independent set of rectangles intersecting this line (which is just the one dimensional problem), and now recurse on the two subsets of rectangles that do not intersect the median line. This results in a partition of the set of rectangles $\mathcal{B}$ into $O(\log n)$ sets, such that for each set, we have the optimal (fractional) independent set. Clearly, one of them has to be of size $\Omega(\mathfrak{i}^*/\log n)$.

For $d > 2$, the same algorithm works by approximating the optimal solution along the median of the first coordinate (i.e., $d-1$ subproblem), and then recursively on the two subproblems. The bounds stated readily follows. ∎

The piercing problem can be solved exactly by the greedy algorithm in one dimension (i.e., add the leftmost right endpoint of an input interval to the piercing set, remove the intervals that

intersect it, and repeat). For higher dimensions, one can perform the same standard divide and conquer approach, used in Lemma 2.3, and get the following.

LEMMA 2.4. ([NIE00]) *Let $\mathcal{B}$ be a set of n boxes in $\mathbb{R}^d$ for $d \geq 2$. A piercing set $\mathcal{P}$ of $\mathcal{B}$ of size $O(p^* \log^{d-1} n)$ can be computed in $O(n \log^{d-1} n)$ time.*

COROLLARY 2.1. *Let $\mathcal{B}$ be a set of n boxes in $\mathbb{R}^d$ for $d \geq 2$. Then the following two sets can be computed in $O(n \log^{d-1} n)$ time:*

(i) An independent set $\mathcal{B}' \subseteq \mathcal{B}$ of $\mathcal{B}$ of size $\Omega\left(\frac{p}{\log^{d-1} n \log\log n}\right)$, and

(ii) a piercing set $\mathcal{P} \subseteq \mathbb{R}^d$ of $\mathcal{B}$ of size $O(p \log^{d-1} n)$.

Proof. By Lemma 2.3, an independent set $\mathcal{B}'$ of size $\Omega\left(\frac{i^*}{\log^{d-1} n}\right)$ can be computed in $O(n \log^{d-1} n)$ time, and by Lemma 2.4, a piercing set $\mathcal{P}$ of size $O(p \log^{d-1} n)$ can be computed in $O(n \log^{d-1} n)$ time. By Lemma 2.2, we have

$$|\mathcal{B}'| = \Omega\left(\frac{i^*}{\log^{d-1} n}\right) = \Omega\left(\frac{p}{\log^{d-1} n \log\log n}\right),$$

∎

Batch sampling For the purposes of our improved algorithm, we need to support the following "batch sampling" operation. Given a set $\mathcal{B}$ of n boxes in $\mathbb{R}^d$, a multiset (i.e., list) $\mathcal{S} \subseteq \mathcal{B}$ of boxes such that $|\mathcal{S}| = O(n \log n)$, and a parameter $r > 0$ such that $r = O(n \log n)$, the task at hand is to compute a random subset $\mathcal{R}$ of r vertices of $\mathcal{A}(\mathcal{B})$, where each vertex $v \in \mathcal{A}(\mathcal{B})$ is sampled independently with probability $w_{\mathcal{S}}(p)/w_{\mathcal{S}}(\mathcal{V})$, see Eq. (2.1). This procedure can be implemented using the data structure described in Definition 2.1, but one can do better, by sweeping along the x_d-axis and constructing the data structure in one lower dimension, as follows.

Let $\mathcal{B}_\downarrow$ be the collection of $(d-1)$-dimensional projections of the boxes in $\mathcal{B}$, to the hyperplane $x_d = 0$. Let $\mathcal{V}'$ be the set of vertices in $\mathcal{A}(\mathcal{B}_\downarrow)$, the arrangement of $\mathcal{B}_\downarrow$ in $\mathbb{R}^{d-1}$. Let $\mathcal{F}$ be the set of x_d-coordinates of all the vertices of the boxes of $\mathcal{B}$.

Build $\mathcal{D}$, an instance of the $(d-1)$-dimensional data structure of Definition 2.1 on $\mathcal{B}_\downarrow$. Importantly, initially all the boxes of $\mathcal{B}_\downarrow$ are inactive. Observe that $\mathcal{V} \subseteq \mathcal{V}' \times \mathcal{F}$. Assume $\mathcal{F}$ is sorted and the ith point (in sorted order) is denoted by e_i. We perform two space-sweeps along the x_d-axis. In step i of a sweep, $\mathcal{D}$ represents the point-set $\mathcal{J}_i := (\mathcal{V}' \times \{e_i\}) \cap \mathcal{V}$ lying on a $\mathbb{R}^{(d-1)}$-dimensional subspace, and we compute $\alpha(i)$, the total weight of vertices in $\mathcal{J}_i$

At any step i of the first sweep, for every box $\mathsf{b} \in \mathcal{B}$ that starts (resp. ends) at e_i call **insert**(b) (resp. **delete**(b)). This activates/deactivates all the vertices on the boundary on b. Similarly, for every $\mathsf{b} \in \mathcal{S}$ that begins (resp. ends) at e_i, use **double**(b) (resp. **halve**(b)) operation on $\mathcal{D}$ to double (resp. halve) the weight of the points in $\mathcal{J}_i \cap \mathsf{b}$. Next, use the **weight** operation on $\mathcal{D}$ to get the weight of all the points in $\mathcal{J}_i$ and store it in $\alpha[i]$. At the end of the first sweep, independently sample r numbers from the set $\{1, ..., 2n\}$ where the integer j is sampled with probability $\alpha(j)/\sum_{i=1}^{2n} \alpha(i)$. Let $\delta(j)$ denote the number of times j was sampled.

Next, reset $\mathcal{D}$ and begin the second sweep. In step i of the second sweep, perform the same operations except replace the weight-computing step with taking $\delta(i)$ independent samples from $\mathcal{J}_i$ using the **sample** operation on $\mathcal{D}$.

Observe that at the end of the first sweep $\alpha(i)/\sum_{j=1}^{2n}\alpha(j)$ is the total probability mass of all the points in $\mathcal{J}_i$. Using this fact, it is easy to verify that at the end of the second sweep the sampled points correspond to the desired subset $\mathcal{R}$.

Initializing $\mathcal{D}$ takes $O(n^{d/2}\log n)$ time, and performing each step of the sweep takes

$$O(n^{(d-2)/2}\log n)$$

time. Each sweep handles $O(|\mathcal{S}|) = O(n\log n)$ events, hence, $\mathcal{R}$ can be computed in $O(n^{d/2}\log^2 n)$ time. We thus obtain the following:

LEMMA 2.5. *Let $\mathcal{B}$ be a set of n boxes in $\mathbb{R}^d$, and let $\mathcal{S}$ be a multiset of boxes such that $|\mathcal{S}| = O(n\log n)$. Let ω be the doubling weight function induced by $\mathcal{S}$ over the vertices $\mathcal{V}$ of $\mathcal{A}(\mathcal{B})$, see Eq. (2.1). Given a parameter $r > 0$ such that $r = O(n\log n)$, a random subset $\mathcal{R} \subset \mathcal{V}$ of size r, where each vertex of $\mathcal{V}$ is sampled with probability proportional to its weight, can be computed in $O(n^{d/2}\log^2 n)$ time.*

The new MWU algorithm As in the previous algorithm, the algorithm performs an exponential search on k until $k \leq 2p$. For a particular guess k, the new algorithm also works in rounds. The difference is how each round is implemented. Instead of doubling the weight of a light box as soon as we find one, we proceed as follows. Let $\omega : \mathcal{V} \to \mathbb{R}_{\geq 0}$ be the weight function at the beginning of the current round. Observe that for any point $p \in \mathcal{V}$, $\omega(p) = w_{\mathcal{S}}(p)$ where $\mathcal{S}$ is the multi-set of boxes that doubled their weights so far, see Eq. (2.1). At the beginning of each round, we process $\mathcal{B}$ and $\mathcal{S}$ to generate a random subset $\mathcal{R} \subset \mathcal{V}$ of size $O(k\log n)$, such that for a box $\mathsf{b} \in \mathcal{B}$ one can determine whether it is light, i.e., $\omega(\mathsf{b}) \leq \omega(\mathcal{V})/4k$ by checking if $|\mathsf{b} \cap \mathcal{R}| \leq |\mathcal{R}|/4k$. By processing $\mathcal{R}$ into an orthogonal range-counting data structure [Aga04], we can compute $|\mathsf{b} \cap \mathcal{R}|$, in $O(\log^{d-1} n)$ time, for any $\mathsf{b} \in \mathcal{B}$, by querying the data structure with b. By repeating this for all boxes of $\mathcal{B}$, we compute a set $\mathcal{L} \subseteq \mathcal{B}$ of light boxes.

Next, we compute an independent set of boxes $\mathcal{I} \subseteq \mathcal{L}$ using Lemma 2.3. There are several possibilities:

- If $|\mathcal{I}| \geq 2k$, then the guess for k is too small. We double the value of k, and restart the process.
- If $|\mathcal{I}| < c_1 k/\log^{2d-1} n$, then by Corollary 2.1, a piercing set $\mathcal{P}$ of $\mathcal{L}$ of $O(k)$ points can be computed in $O(n\log^{d-1} n)$ time. The remaining boxes of $\mathcal{H} = \mathcal{B} \setminus \mathcal{L}$ are (say) $1/4.01k$-heavy. By applying Lemma 6.12 in [AHRS23] to $\mathcal{H}$ and $\mathcal{R}$, we compute a weak $1/4.01k$-net $\mathcal{N}$ for these boxes of size $O(k\log\log k)$ in $O(n\log^d n)$ expected time. The set $\mathcal{P} \cup \mathcal{N}$ is the desired piercing set of $\mathcal{B}$ of size $O(p\log\log p)$.
- If $|\mathcal{I}| \geq c_1 k/\log^{2d-1} n$, then we update $\mathcal{S}$, the multi-set of boxes doubled so far, to $\mathcal{S} = \mathcal{S} \cup \mathcal{I}$. The algorithm now continues to the next round.

REMARK 2.1. *It is interesting to observe that the batch doubling operation in the above algorithm corresponds to merely updating the list $\mathcal{S}$. The actual work associated with the doubling is in regenerating the sample $\mathcal{R}$ at the beginning of the next round. This is done by batch sampling as described below.*

Computing the random sample $\mathcal{R}$ in each round Let $\omega : \mathcal{V} \to \mathbb{R}_{\geq 0}$ be the weight function in the beginning of a particular round. Our goal is to compute a sample $\mathcal{R} \subset \mathcal{V}$, such that we can

check if a box $\mathsf{b} \in \mathcal{B}$ is light by checking if $|\mathsf{b} \cap \mathcal{R}| \leq |\mathcal{R}|/4k$. Recall that $\mathcal{S}$ denotes the list of boxes for which the MWU algorithm had doubled the weights. Moreover, recall that the weight of a vertex $v \in \mathcal{V}$ is $2^{|\mathcal{S} \sqcap v|}$, where $|\mathcal{S} \sqcap v|$ denotes the number of boxes in $\mathcal{S}$ that contain v. We use the concept of relative approximations.

DEFINITION 2.2. *Let $\Sigma = (\mathcal{X}, \mathcal{R})$ be a finite range space, and $\mathcal{R} \subset \mathcal{X}$, and let $\omega : \mathcal{X} \to \mathbb{R}_{\geq 0}$ be a weight function. For a range $\mathbf{r} \in \mathcal{R}$, let $\overline{m}(\mathbf{r}) = \omega(\mathbf{r} \cap \mathsf{X})/\omega(\mathsf{X})$ and $\overline{s}(\mathbf{r}) = |\mathbf{r} \cap \mathcal{R}|/|\mathcal{R}|$. Then, $\mathcal{R}$ is a* **relative (p, ε)-approximation** *of Σ if for each $\mathbf{r} \in \mathcal{R}$ we have:*

(i) If $\overline{m}(\mathbf{r}) \geq p$, then $(1-\varepsilon)\overline{m}(\mathbf{r}) \leq \overline{s}(\mathbf{r}) \leq (1+\varepsilon)\overline{m}(\mathbf{r})$.

(ii) If $\overline{m}(\mathbf{r}) \leq p$, then $\overline{s}(\mathbf{r}) \leq (1+\varepsilon)p$.

It is known [HS11, Har11] that if the VC-dimension of Σ is d, then a random sample $\mathcal{R}$ of size $O\big(\frac{1}{\varepsilon^2 p}\big[d \log \frac{1}{p} + \log \frac{1}{\varphi}\big]\big)$, where each point is chosen with probability proportional to its weight, is a relative (p, ε)-approximation with probability $\geq 1 - \varphi$. In view of this result, we can detect light boxes of $\mathcal{B}$ (with respect to $\mathcal{S}$) as follows. Set $\varepsilon = 0.01$, $p = \frac{\varepsilon}{20k}$, $\varphi = \frac{1}{n^{O(1)}}$. We chose a random subset $\mathcal{R} \subseteq \mathcal{V}$ of $r = O(\frac{k}{\varepsilon^2} \log n)$ points. Then we have the following property for each box $\mathsf{b} \in \mathcal{B}$.

(I) If $\overline{m}(\mathsf{b}) = \omega(\mathsf{b} \cap \mathcal{V})/\omega(\mathcal{B}) \geq (\varepsilon/20k)$ then $0.99\overline{m}(\mathsf{b}) \leq \overline{s}(\mathsf{b}) \leq 1.01\overline{m}(\mathsf{b})$.

(II) If $\overline{m}(\mathsf{b}) \leq \varepsilon/20k$ then $\overline{s}(\mathsf{b}) \leq 1.01/20k$.

Therefore to check if a box is b is light, it suffices to check $|\mathsf{b} \cap \mathcal{R}|$. As for computing the sample $\mathcal{R}$, observe that this is exactly what batch sampling is designed for, see Lemma 2.5.

Analysis The correctness of the new MWU follows from the previous one. We now analyze the running time. Each round except the last round doubles the weight of $\Omega(k/\log^{2d} n)$ boxes. Since the total number of weight-doubling operations performed by the algorithm is $O(p \log n)$, see [AP20], the algorithm stops within $O(\log^{2d} n)$ rounds. In a particular round, the algorithm uses batch sampling to recompute the random subset $\mathcal{R}$ which it uses to identify the set $\mathcal{L} \subseteq \mathcal{B}$ of light boxes. Lemma 2.5 shows that the cost of batch sampling is bounded by $O(n^{d/2} \log^2 n)$ time. As discussed above, once $\mathcal{R}$ is computed, $\mathcal{L}$ can be identified in $O(n \log^{d-1} n)$ time. Finding an independent set $\mathcal{I} \subseteq \mathcal{L}$ using Lemma 2.3 also takes $O(n \log^{d-1} n)$ time. Updating the multi-set $\mathcal{S}$ of boxes whose weights have been doubled so far also takes $O(n)$ time. The algorithm computes a piercing set and a weak net only in the last round. Together, they take $O(n \log^d n)$ time. The running time for these steps is dominated by the the time for the batch sampling in each round. Considering the need to do an exponential search for p, we obtain the following:

THEOREM 2.2. *Let $\mathcal{B}$ be set of n axis-aligned boxes in $\mathbb{R}^d$, for $d \geq 2$, and let $p = p(\mathcal{B})$ be the piercing number of $\mathcal{B}$. A piercing set of $\mathcal{B}$ of size $O(p \log \log p)$ can be computed in $O(n^{d/2} \log^{2d+3} n)$ expected time.*

3 Multi-round piercing algorithm

Let $\mathcal{B}$ be a set of (closed) boxes in $\mathbb{R}^d$, and let $\mathcal{V} = \mathcal{V}(\mathcal{B})$ be the set of vertices of the arrangement $\mathcal{A}(\mathcal{B})$. When considering a piercing set for $\mathcal{B}$, one can restrict the selection of piercing points to points of $\mathcal{V}$. Two point sets $\mathcal{Q}, \mathcal{Q}'$ are ***equivalent*** for $\mathcal{B}$ if for all faces (of all dimensions) f of the arrangement of $\mathcal{B}$, we have that $|\mathcal{Q} \cap \mathsf{f}| = |\mathcal{Q}' \cap \mathsf{f}|$. Since $|\mathcal{V}| = O(n^d)$, it follows that the number of non-equivalent (i.e., distinct) piercing sets of size $\leq t$ is bounded by $O(n^{dt})$.

The key insight is that a piercing set for a sufficiently large, but not too large, sample is a piercing set for almost all the boxes.

LEMMA 3.1. *Let $\mathcal{B}$ be a set of n boxes in $\mathbb{R}^d$, $\delta \in (0,1)$ and $t > 0$ be parameters, and let $\mathcal{S} \subseteq \mathcal{B}$ be a random sample of size $O(\frac{dt}{\delta} \log n)$. If $\mathcal{S}$ can be pierced by a set $\mathcal{Q}$ of t points, then at most δn boxes of $\mathcal{B}$ are not pierced by $\mathcal{Q}$, and this holds with probability $\geq 1 - 1/n^{O(d)}$.*

Proof. Let $\mathcal{F}$ be the collection of all piercing sets of $\mathcal{B}$ with at most t points, where no two sets in $\mathcal{F}$ are equivalent for $\mathcal{B}$. Let $\mathcal{F}' = \left\{ X \in \mathcal{F} \;\middle|\; |\mathcal{B} \setminus X| \geq \delta n \right\}$ be all the "bad" piercing set in $\mathcal{F}$ that fail to stab $\geq \delta n$ boxes of $\mathcal{B}$, where $\mathcal{B} \setminus X = \{\mathsf{b} \in \mathcal{B} \mid \mathsf{b} \cap X = \emptyset\}$. By the above, we have that $m = |\mathcal{F}'| \leq O(n^{dt})$. Fix a "bad" set $X \in \mathcal{F}'$. Let $u = c\frac{dt}{\delta} \ln n$ be the size of $|\mathcal{S}|$, where c is a sufficiently large constant. The probability that the set X is a piercing set for the sample $\mathcal{S}$ is at most

$$\psi = (1-\delta)^u \leq \exp(-\delta u) = \exp(-c \cdot d \cdot t \cdot \ln n) = \frac{1}{n^{cdt}}.$$

In particular, by the union bound, the probability that any set of $\mathcal{F}'$ will be a valid piercing set for $\mathcal{S}$ is at most $|\mathcal{F}'|\,\psi < 1/n^{O(d)}$, for c sufficiently large. ∎

3.1 A piercing algorithm via sampling Lemma 3.1 suggests a natural algorithm for piercing – pick a random sample from $\mathcal{B}$, compute (or approximate) a piercing set for it, compute the boxes this piercing set misses, and repeat the process for several rounds. In the last round, hopefully, the number of remaining boxes is sufficient small than one can apply the piercing approximation algorithm directly to $\mathcal{B}$. We thus get the following.

LEMMA 3.2. *Let $\mathcal{B}$ be a set of n boxes in $\mathbb{R}^d$, and let $\zeta > 1$ be a parameter. Furthermore, assume that we are given an algorithm* **Pierce** *that for m boxes, can compute a $O(\log\log \mathfrak{p})$ approximate to their piercing set in $T_{\mathbf{P}}(m)$ time, where $\mathfrak{p}$ is the size of the optimal piercing set for the given set. Then, one can compute a piercing set for $\mathcal{B}$ of size $O(\zeta \mathfrak{p} \log\log \mathfrak{p})$ in expected time*

$$O\Big(\zeta T_{\mathbf{P}}\big(\mathfrak{p}^{1-1/\zeta} n^{1/\zeta} \log n\big) + n \log^{d-1} \mathfrak{p}\Big).$$

Proof. We assume that we have a number k such that $\mathfrak{p} \leq k \leq 2\mathfrak{p}$, where $\mathfrak{p}$ is the size of the optimal piercing set for $\mathcal{B}$. To this end, one can perform an exponential search for this value, and it is easy to verify that this would not effect the running time of the algorithm.

The algorithm performs ζ rounds. Let $\mathcal{B}_0 = \mathcal{B}$. Let $\delta = (k/n)^{1/\zeta}$. In the ith round, for $i = 1, \ldots, \zeta - 1$, we pick a random sample $\mathcal{S}_i$ from $\mathcal{B}_{i-1}$ of size

$$m = O\Big(dk\delta^{-1} \log n\Big) = O\big(dk^{1-1/\zeta} n^{1/\zeta} \log n\big) = O\big(\mathfrak{p}^{1-1/\zeta} n^{1/\zeta} \log n\big).$$

In the ζth round, we set $\mathcal{S}_\zeta = \mathcal{B}_{\zeta-1}$. Now, we approximate the optimal piercing set for $\mathcal{S}_i$, by calling $\mathcal{Q}_i \leftarrow$ **Pierce**$(\mathcal{S}_i)$. If the piercing set $\mathcal{Q}_i$ is too large – that is, $|\mathcal{Q}_i| \gg k \log\log k$, then the guess for k is too small, and the algorithm restarts with a larger guess for k. Otherwise, the algorithm builds a range tree for $\mathcal{Q}_i$, and streams the boxes of $\mathcal{B}_{i-1}$ through the range tree, to compute the set $\mathcal{B}_i = \mathcal{B}_{i-1} \setminus \mathcal{Q}_i$, the boxes in $\mathcal{B}_{i-1}$ not pierced by $\mathcal{Q}_i$. By Lemma 3.1, we have $|\mathcal{B}_i| \leq \delta\,|\mathcal{B}_{i-1}| \leq \delta^i n$ with high probability. If $|\mathcal{B}_i| > \delta\,|\mathcal{B}_{i-1}|$, we repeat round i, so assume $|\mathcal{B}_i| \leq \delta\,|\mathcal{B}_{i-1}|$. In particular $|\mathcal{S}_\zeta| = |\mathcal{B}_{\zeta-1}| \leq \delta^{\zeta-1} n \leq \mathfrak{p}^{1-1/\zeta} n^{1/\zeta}$. Therefore, the total time spent by **Pierce**(.) in ζ rounds is $O(\zeta T_{\mathbf{P}}(\mathfrak{p}^{1-1/\zeta} n^{1/\zeta} \log n))$. Finally, during the first $\zeta - 1$ iterations, computing $\mathcal{B}_i$ takes

$$\sum_{i=1}^{\zeta-1} O(|\mathcal{B}_{i-1}| \log^{d-1} |\mathcal{Q}_i|) = \sum_{i=1}^{\zeta-1} O(\delta^i n \log^{d-1}(\mathfrak{p} \log\log \mathfrak{p})) = O(n \log^{d-1} \mathfrak{p})$$

time. Clearly, $\cup_i \mathcal{Q}_i$ is the desired piercing set. ∎

We can use the algorithm of Theorem 2.2, for the piercing algorithm. For this choice,

$$T_{\mathbf{P}}(m) = O(m^{d/2} \log^{2d+3} m).$$

We then get an approximation algorithm with running time

$$O\left(\zeta\left(\mathfrak{p}^{1-1/\zeta} n^{1/\zeta}\right)^{d/2} \operatorname{polylog}(n) + n \log^{d-1} \mathfrak{p}\right).$$

We thus get our second main result.

THEOREM 3.1. *Let $\mathcal{B}$ be a set of n axis-aligned boxes in $\mathbb{R}^d$, for $d \geq 2$, and let $\zeta > 0$ be an integer. A piercing set of $\mathcal{B}$ of size $O(\zeta \mathfrak{p} \log\log \mathfrak{p})$ can be computed in*

$$O\left(\zeta \mathfrak{p}^{d/2-d/2\zeta} n^{d/2\zeta} \operatorname{polylog}(n) + n \log^{d-1} \mathfrak{p}\right).$$

expected time, where $\mathfrak{p} = \mathfrak{p}(\mathcal{B})$ is the size of the optimal piercing set.

The above algorithm provides a trade-off between the approximation factor and the running time. It readily leads to a near linear time algorithm if the piercing set is sufficiently small. For example, by choosing $\zeta = d$, we obtain the following:

COROLLARY 3.1. *Let $\mathcal{B}$ be a set of n axis-aligned boxes in $\mathbb{R}^d$ for some fixed $d \geq 2$, and assume $\mathfrak{p}(\mathcal{B}) = O(n^{1/(d-1)})$. Then, a piercing set of $\mathcal{B}$ of size $O(d\mathfrak{p} \log\log \mathfrak{p})$ can be computed in $O(n \operatorname{polylog}(n))$ expected time.*

If the piercing set is slightly sublinear, the above leads to an approximation algorithm with running time $O(n \log n)$.

COROLLARY 3.2. *Let $\mathcal{B}$ be a set of n axis-aligned rectangles in $\mathbb{R}^2$ for some fixed $d \geq 2$, and assume that it can be pierced by $\mathfrak{p} = O(n/\log^{15} n)$ points. Then, a piercing set of $\mathcal{B}$ of size $O(\mathfrak{p} \log\log \mathfrak{p})$ can be computed in $O(n \log \mathfrak{p})$ expected time.*

Proof. Pick a random sample $\mathcal{S} \subseteq \mathcal{B}$ of size $O(n/\log^7 n)$. The algorithm of Theorem 2.2 yields in $O(n)$ time a piercing set $\mathcal{Q}$ for $\mathcal{S}$, of size $u = O(\mathfrak{p} \log\log \mathfrak{p})$. Preprocess $\mathcal{Q}$ for orthogonal range emptiness queries – this takes $O(\mathfrak{p} \log^2 \mathfrak{p})$ time, and one can decide if a rectangle is not pierced by $\mathcal{Q}$ in $O(\log \mathfrak{p})$ time. Lemma 3.1 implies that at most δn rectangles unpierced by $\mathcal{Q}$, where

$$\delta = \frac{\mathfrak{p} \log^8 n}{n}.$$

Namely, the unhit set has size $\delta n = O(n/\log^7 n)$. Running time algorithm of Theorem 2.2 on this set of rectangles, takes $O(n)$ time, and yields a second piercing set $\mathcal{Q}'$ of size $O(\mathfrak{p} \log\log \mathfrak{p})$. Combining the two sets results in the desired piercing set. ∎

4 Dynamic Algorithm for piercing

We present a data structure for maintaining a near-optimal piercing set for a set $\mathcal{B}$ of boxes in $\mathbb{R}^2$ as boxes are inserted into or deleted from $\mathcal{B}$. By adapting the multi-round sampling based algorithm described in Section 3, we obtain a Monte Carlo algorithm that maintains a piercing set of $O(\mathfrak{p} \log\log \mathfrak{p})$ size with high probability and that can update the piercing set in $O^*(n^{1/2})$ amortized expected time per update. (The $O^*()$ notation hides polylogarithmic factors). The update time can be improved to $O^*(n^{1/3})$ if $\mathcal{B}$ is a set of squares in $\mathbb{R}^2$.

Overview of the Algorithm We observe that the size of the optimal piercing set changes by at most one when a box is inserted or deleted. We periodically reconstruct the piercing set using a faster implementation of the multi-round sampling based algorithm in Section 3, as described below. More precisely, if s is the size of the piercing set computed during the previous reconstruction, then we reconstruct the piercing set after $\lceil s/2 \rceil$ updates. To expedite the reconstruction, we maintain $\mathcal{B}$ in a data structure as follows. We map a box $\mathsf{b} = [a_1, a_2] \times [b_1, b_2]$ to the point $\mathsf{b}^* = (a_1, b_1, a_2, b_2)$ in $\mathbb{R}^4$, and let $\mathcal{B}^*$ be the resulting set of points in $\mathbb{R}^4$. We store $\mathcal{B}^*$ into a 4-dimensional dynamic range tree T, which is a 4-level tree. Each node v of T is associated with a *canonical subset* $\mathcal{B}^*_v \subseteq \mathcal{B}^*$ of points. Let $\mathcal{B}_v$ be the set of boxes corresponding to $\mathcal{B}^*_v$. For a box $\square$ in $\mathbb{R}^4$, $\square \cap \mathcal{B}^*$ can be represented as the union of $O(\log^4 n)$ canonical subsets, and they can be computed in $O(\log^4 n)$ time. The size of T is $O(n \log^4 n)$, and it can be updated in $O(\log^4 n)$ amortized time per insertion/deletion of point. See [dBCvKO08].

Between two consecutive reconstructions, we use a lazy approach to update the piercing set, as follows: Let $\mathcal{P}$ be the current piercing set. When a new box b is inserted, we insert it into T. If $\mathcal{P} \cap \mathsf{b} = \emptyset$, we choose an arbitrary point p inside b and add p to $\mathcal{P}$. When we delete a box b, we simply delete b^* from T but do not update $\mathcal{P}$. If $\lceil s/2 \rceil$ updates have been performed since the last reconstruction, we discard the current $\mathcal{P}$ and compute a new piercing set as described below.

We show below that a piercing set of size $s := O(\boldsymbol{\rho} \log\log \boldsymbol{\rho})$ of $\mathcal{B}$ can be constructed in

$$O^*\left((\boldsymbol{\rho} n)^{1/2} + \min\{\boldsymbol{\rho}^2, n\}\right)$$

expected time, where $\boldsymbol{\rho}$ is the size of the optimal piercing set of $\mathcal{B}$. This implies the amortized expected update time is $O^*\left((n/\boldsymbol{\rho})^{1/2} + \min\{\boldsymbol{\rho}, n/\boldsymbol{\rho}\}\right)$, including the time spent in updating T. The second term is bounded by $n^{1/2}$, so the amortized expected update time is $O^*(n^{1/2})$.

Reconstruction algorithm Here is how we construct the piercing set of boxes in $\mathbb{R}^2$. Let $\mathcal{B}$ be the current set of boxes. We follow the algorithm in Theorem 3.1 and set the number of rounds to 2. More precisely, perform an exponential search on the value of k, the guess for the size of the optimal piercing set, every time we reconstruct the piercing set. For a fixed k, the reconstruction algorithm consists of the following steps:

(I) Choose a random sample $\mathcal{B}_1$ of $\mathcal{B}$ of size $r = c_1(kn)^{1/2}$, where c_1 is a suitable constant.
(II) Construct a piercing set $\mathcal{P}_1$ of $\mathcal{B}_1$ of size $s = O(k \log\log k)$ in $O^*(r)$ time using the algorithm in Section 2.2.
(III) Compute $\mathcal{B}_2 \subseteq \mathcal{B}$, the subset of boxes that are not pierced by $\mathcal{P}_1$. If $|\mathcal{B}_2| > c_2(kn)^{1/2}$, where c_2 is a suitable constant, we return to Step 1. As described below, this step can be computed in $O^*(\min\{k^2, n\} + (kn)^{1/2})$ time.
(IV) Compute a piercing set $\mathcal{P}_2$ of $\mathcal{B}_2$, again using the algorithm in Section 2.2.
(V) Return $\mathcal{P}_1 \cup \mathcal{P}_2$.

The expected running time of this algorithm is $O^*(\min\{k^2, n\} + (kn)^{1/2})$, as desired.

Computing $\mathcal{B}_2$ We now describe how to compute $\mathcal{B}_2$ efficiently using T. If $\boldsymbol{\rho} \geq n^{1/2}$, then we simply preprocess $\mathcal{P}$ into a 2-dimensional range tree in $O(s \log s)$ time. By querying with each box in $\mathcal{B}$, we can compute $\mathcal{B}_2$ in $O(n \log n)$ time [dBCvKO08]. The total time spent is $O(n \log n)$. So assume $\boldsymbol{\rho} < n^{1/2}$. For a point $p = (x_p, y_p) \in \mathbb{R}^2$, let $Q_p \subset \mathbb{R}^4$ be the orthant $Q_p = \{(x_1, x_2, x_3, x_4) \mid x_1 \leq x_p, x_2 \leq y_p, x_3 \geq x_p, x_4 \geq y_p\}$. Then, a box $\mathsf{b} \subset \mathbb{R}^2$ contains p if and only if $\mathcal{B}^* \in Q_p$. Therefore, an input box $\mathsf{b} \in \mathcal{B}$ is not pierced by $\mathcal{P}_1$ if $\mathsf{b}^* \notin \bigcup_{p \in \mathcal{P}} Q_p$. Let $\mathcal{K} = \mathbb{R}^4 \setminus \bigcup_{p \in \mathcal{P}} Q_p$. It is well known that the complexity of $\mathcal{K}$ is $O(s^2)$. Furthermore, $\mathcal{K}$ can be

partitioned into $O(s^2)$ boxes with pairwise-disjoint interiors, as follows.

Let $\mathcal{Q} = \{Q_p \mid p \in \mathcal{P}\}$, and let $\hat{\mathcal{P}} = \{(x_p, y_p, x_p, y_p) \mid p \in \mathcal{P}\}$ be their corners. We sort $\hat{\mathcal{P}}$ by the x_4-coordinates of its points. Let Δ be an x_4-interval between the x_4-coordinates of two consecutive points of $\hat{\mathcal{P}}$. For any value $a \in \Delta$, the cross-section $\mathcal{Q}_a$ of $\mathcal{Q}$ with the hyperplane $h_a : x_4 = a$ is a collection of s 3-dimensional octants, and $\mathcal{K}_a := \mathcal{K} \cap h_a$ is the complement of the union of $\mathcal{Q}_a$. Furthermore, the cross-section $\mathcal{K}_a$ remains the same for any value of $a \in \Delta$. It is well known that the complexity of $\mathcal{K}_a$ is $O(s)$, and that it can be partitioned into a set $\mathcal{R}_a$ of 3-dimensional boxes in $O^*(s)$ time. Hence, we can partition $\mathcal{K}$ inside the slab $\mathbb{R}^3 \times \Delta$ by the set $\mathcal{R}_\Delta = \{R \times \Delta \mid R \in \mathcal{R}_a\}$. By repeating this procedure for all x_4-intervals between two consecutive points of $\hat{\mathcal{P}}$, we partition $\mathcal{K}$ into a family $\mathcal{R}$ of $O(s^2)$ boxes.

Next, we query T with each box $R \in \mathcal{R}$. The query procedure returns a set V_R of $O(\log^4 n)$ nodes of T such that $\mathcal{B}^* \cap R = \bigcup_{v \in V_R} \mathcal{B}^*_v$. We thus obtain a set V of $O(s^2 \log^4 n)$ nodes of T such that $\mathcal{B}^*_2 = \bigcup_{v \in V} \mathcal{B}^*_v$. If $\sum_{v \in V} |\mathcal{B}^*_v| \leq c_2(kn)^{1/2}$, we return $\bigcup_{v \in V} \mathcal{B}_v$ as $\mathcal{B}_2$. Otherwise, we return NULL. The total time spent by this procedure is $O^*(\min\{n, k^2\} + (kn)^{1/2})$. Putting everything together we obtain the following.

THEOREM 4.1. *A set $\mathcal{B}$ of n boxes in $\mathbb{R}^2$ can be stored in a data structure so that a piercing set of $\mathcal{B}$ of size $O(\mathfrak{p} \log\log \mathfrak{p})$ can be maintained with high probability under insertion and deletion of boxes with amortized expected time $O(n^{1/2}\,\mathrm{polylog}(n))$ per insertion or deletion; $\mathfrak{p}$ is the piercing number of $\mathcal{B}$.*

Dynamic algorithm for squares in 2D If we have squares instead of boxes, then the reconstruction time reduces to $O^*(\mathfrak{p}^{2/3} n^{1/3})$, which leads to an amortized update time of $O^*((n/\mathfrak{p})^{1/3}) = O^*(n^{1/3})$. We proceed in a similar manner as before. There are two differences. First, we now choose a random sample of size $O(k^{2/3} n^{1/3})$, and the algorithm works in three rounds. After the first round, we have a piercing set $\mathcal{P}_1$ of size $O(\mathfrak{p} \log\log \mathfrak{p})$, and we need to represent the set of squares not pierced by $\mathcal{P}_1$ as $O^*(\mathfrak{p})$ canonical subsets, so that we can choose a random sample $\mathcal{B}_2$ from this subset of squares. After the second round, we have a piercing set $\mathcal{P}_2$ of $\mathcal{B}_2$ of size $O(\mathfrak{p} \log\log \mathfrak{p})$. Finally, we find the subset $\mathcal{B}_3 \subseteq \mathcal{B}$ of squares not pierced by $\mathcal{P}_1 \cup \mathcal{P}_2$ and compute a piercing set of $\mathcal{B}$. It suffices to describe how we compactly represent the set $\mathcal{B}_2$.

We map a square, which is centered at a point c and of radius (half side length) a, to the point $\mathsf{b}^* = (c, a) \in \mathbb{R}^3$. A point $p = (x_p, y_p) \in \mathbb{R}^2$ is now mapped to the cone $C_p = \{(x, y, z) \in \mathbb{R}^3 \mid \|(x, y) - p\|_\infty \leq z,\ z \geq 0\}$ with the square cross-section and p its apex; C_p is the graph of the L_∞-distance function from p. It is easily seen that $p \in \mathsf{b}$ if and only if $\mathsf{b}^* \in C_p$. Hence, a box $\mathsf{b} \in \mathcal{B}_2$ if it lies below all the cones of $C = \{C_p \mid p \in \mathcal{P}_1\}$. Using a 3D orthogonal range-searching data structure, we can compute $\mathcal{B}_2$ as the union of $O^*(k)$ canonical subsets. We omit the details from here and obtain the following.

THEOREM 4.2. *A set $\mathcal{B}$ of n squares in $\mathbb{R}^2$ can be stored in the data structure described above so that a piercing set of $\mathcal{B}$ of size $O(\mathfrak{p} \log\log \mathfrak{p})$ can be maintained with high probability under insertion and deletion of boxes in $O(n^{1/3}\,\mathrm{polylog}(n))$ amortized expected time per insertion or deletion.*

References

[ACS+22] Pankaj K. Agarwal, Hsien-Chih Chang, Subhash Suri, Allen Xiao, and Jie Xue. Dynamic geometric set cover and hitting set. *ACM Trans. Algorithms*, 18(4):40:1–40:37, 2022.

[AES09] Boris Aronov, Esther Ezra, and Micha Sharir. Small-size epsilon-nets for axis-parallel rectangles and boxes. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 639–648. ACM, 2009.

[Aga04] Pankaj K. Agarwal. Range searching. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 36, pages 809–838. CRC Press LLC, Boca Raton, FL, USA, 2nd edition, 2004.

[AHRS23] Pankaj K. Agarwal, Sariel Har-Peled, Rahul Raychaudhury, and Stavros Sintos. Fast approximation algorithms for piercing boxes by points, 2023. https://arxiv.org/abs/2311.02050.

[AK92] Noga Alon and Daniel J. Kleitman. Piercing convex sets. In David Avis, editor, *Proc. 8th Annu. Sympos. Comput. Geom.* (SoCG), pages 157–160. ACM, 1992.

[AP20] Pankaj K. Agarwal and Jiangwei Pan. Near-linear algorithms for geometric hitting sets and set covers. *Discrete Comput. Geom.*, 63(2):460–482, 2020.

[AvKS97] Pankaj K. Agarwal, Marc J. van Kreveld, and Subhash Suri. Label placement by maximum independent set in rectangles. In *Proceedings of the 9th Canadian Conference on Computational Geometry, Kingston, Ontario, Canada, August 11-14, 1997*, 1997.

[BG95] Hervé Brönnimann and Michael T. Goodrich. Almost optimal set covers in finite vc-dimension. *Discrete Comput. Geom.*, 14(4):463–479, 1995.

[BMR18] Norbert Bus, Nabil H. Mustafa, and Saurabh Ray. Practical and efficient algorithms for the geometric hitting set problem. *Discret. Appl. Math.*, 240:25–32, 2018.

[CC09] Parinya Chalermsook and Julia Chuzhoy. Maximum independent set of rectangles. In Claire Mathieu, editor, *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 892–901. SIAM, 2009.

[CH12] Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete Comput. Geom.*, 48:373–392, 2012.

[CH20] Timothy M. Chan and Qizheng He. Faster approximation algorithms for geometric set cover. In Sergio Cabello and Danny Z. Chen, editors, *Proc. 36th Int. Annu. Sympos. Comput. Geom.* (SoCG), volume 164 of *LIPIcs*, pages 27:1–27:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[CH21] Timothy M. Chan and Qizheng He. More dynamic data structures for geometric set cover with sublinear update time. In *37th International Symposium on Computational Geometry, SoCG 2021, June 7-11, 2021, Buffalo, NY, USA (Virtual Conference)*, volume 189 of *LIPIcs*, pages 25:1–25:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[Cha03] Timothy M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *J. Algorithms*, 46(2):178–189, 2003.

[CHSX22] Timothy M. Chan, Qizheng He, Subhash Suri, and Jie Xue. Dynamic geometric set cover, revisited. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 3496–3528. SIAM, 2022.

[CM05] Timothy M. Chan and Abdullah-Al Mahmood. Approximating the piercing number for unit-height rectangles. In *Proc. 17th Canad. Conf. Comput. Geom.* (CCCG), pages 15–18, 2005.

[CSZ18] Maria Chudnovsky, Sophie Spirkl, and Shira Zerbib. Piercing axis-parallel boxes. *Electron. J. Comb.*, 25(1):1, 2018.

[dBCvKO08] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications*. Springer, 3rd edition, 2008.

[Ede87] Herbert Edelsbrunner. *Algorithms in combinatorial geometry*, volume 10. Springer Science & Business Media, 1987.

[EKNS97] Alon Efrat, Matthew J. Katz, Frank Nielsen, and Micha Sharir. Dynamic data structures for

fat objects and their applications. In Frank K. H. A. Dehne, Andrew Rau-Chaplin, Jörg-Rüdiger Sack, and Roberto Tamassia, editors, *Algorithms and Data Structures, 5th International Workshop, WADS '97, Halifax, Nova Scotia, Canada, August 6-8, 1997, Proceedings*, volume 1272 of *Lect. Notes in Comp. Sci.*, pages 297–306. Springer, 1997.

[Ezr10] Esther Ezra. A note about weak epsilon-nets for axis-parallel boxes in d-space. *Inf. Process. Lett.*, 110(18-19):835–840, 2010.

[FPT81] Robert J. Fowler, Mike Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are np-complete. *Inf. Process. Lett.*, 12(3):133–137, 1981.

[GKM+22] Waldo Gálvez, Arindam Khan, Mathieu Mari, Tobias Mömke, Madhusudhan Reddy Pittu, and Andreas Wiese. A 3-approximation algorithm for maximum independent set of rectangles. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 894–905. SIAM, 2022.

[GRS98] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: an efficient clustering algorithm for large databases. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 73–84. ACM Press, 1998.

[Har04] Sariel Har-Peled. Clustering motion. *Discrete Comput. Geom.*, 31(4):545–565, 2004.

[Har11] Sariel Har-Peled. *Geometric Approximation Algorithms*, volume 173 of *Math. Surveys & Monographs*. Amer. Math. Soc., Boston, MA, USA, 2011.

[HM85] Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. Assoc. Comput. Mach.*, 32(1):130–136, 1985.

[HS11] Sariel Har-Peled and Micha Sharir. Relative (p, ε)-approximations in geometry. *Discrete Comput. Geom.*, 45(3):462–496, 2011.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.

[KLR+23] Arindam Khan, Aditya Lonkar, Saladi Rahul, Aditya Subramanian, and Andreas Wiese. Online and dynamic algorithms for geometric set cover and hitting set. In *39th International Symposium on Computational Geometry, SoCG 2023, June 12-15, 2023, Dallas, Texas, USA*, volume 258 of *LIPIcs*, pages 46:1–46:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[Mit21] Joseph S. B. Mitchell. Approximating maximum independent set for rectangles in the plane. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 339–350. IEEE, 2021.

[MR10] Nabil H. Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete Comput. Geom.*, 44(4):883–895, 2010.

[Nie00] Frank Nielsen. Fast stabbing of boxes in high dimensions. *Theor. Comput. Sci.*, 246(1-2):53–72, 2000.

[OY91] Mark H. Overmars and Chee-Keng Yap. New upper bounds in klee's measure problem. *SIAM J. Comput.*, 20(6):1034–1045, 1991.

[PT11] János Pach and Gábor Tardos. Tight lower bounds for the size of epsilon-nets. In *Proceedings of the twenty-seventh annual symposium on Computational geometry*, pages 458–463, 2011.

[VC71] Vladimir N. Vapnik and Alexey Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.