Optimal Partition Trees

Timothy M. Chan*

February 12, 2010

Abstract

We revisit one of the most fundamental classes of data structure problems in computational geometry: range searching. Back in SoCG'92, Matoušek gave a partition tree method for d-dimensional simplex range searching achieving $O(n)$ space and $O(n^{1-1/d})$ query time. Although this method is generally believed to be optimal, it is complicated and requires $O(n^{1+\varepsilon})$ preprocessing time for any fixed $\varepsilon > 0$. An earlier method by Matoušek (SoCG'91) requires $O(n \log n)$ preprocessing time but $O(n^{1-1/d} \log^{O(1)} n)$ query time. We give a new method that achieves simultaneously $O(n \log n)$ preprocessing time, $O(n)$ space, and $O(n^{1-1/d})$ query time with high probability. Our method has several advantages:

- It is conceptually simpler than Matoušek's SoCG'92 method. Our partition trees satisfy many ideal properties (e.g., constant degree, optimal crossing number at almost all layers, and disjointness of the children's cells at each node).
- It leads to more efficient multilevel partition trees, which are important in many data structural applications (each level adds at most one logarithmic factor to the space and query bounds, better than in all previous methods).
- A similar improvement applies to a shallow version of partition trees, yielding $O(n \log n)$ time, $O(n)$ space, and $O(n^{1-1/\lfloor d/2 \rfloor})$ query time for halfspace range emptiness in even dimensions $d \geq 4$.

Numerous consequences follow (e.g., improved results for computing spanning trees with low crossing number, ray shooting among line segments, intersection searching, exact nearest neighbor search, linear programming queries, finding extreme points, ...).

1 Introduction

Data structures for range searching [4, 7, 39, 43] are among the most important and most often used results within the computational geometry literature—countless papers applied such results and techniques to obtain the best theoretical computational bounds for a wide variety of geometric problems. However, to this day, some questions remain even concerning the most basic version of nonorthogonal range searching, simplex range searching. The purpose of this paper is to provide a (hopefully) final resolution to some of these lingering questions, by nailing down the best precise upper bounds.

*Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada (tmchan@uwaterloo.ca). Work supported by NSERC.

History. Formally, in *simplex range searching*, we want to preprocess n (weighted) points in $\mathbb{R}^d$ so that we can quickly find (the sum of the weights of) all points inside a given query simplex. The dimension d is assumed to be a small constant. The problem not only is fundamental but has a remarkably rich history. We will confine the discussion of previous work to linear- or near-linear space data structures (otherwise, the number of results multiplies). The first published method was by Willard [49] in 1982, who gave a simple $O(n)$-space *partition tree* achieving $O(n^{0.792})$ query time in 2-d. This prompted researchers to look for data structures with the best exponent in the query time. Many subsequent results appeared; for example, in 2-d, Willard himself improved the exponent to about 0.774, which was further reduced to 0.695 by Edelsbrunner and Welzl [31]; in 3-d, F. Yao [51] obtained 0.936, Dobkin and Edelsbrunner [28] 0.916, Edelsbrunner and Huber [30] 0.909, and Yao *et al.* [52] 0.899; in higher dimensions, Yao and Yao [50] obtained $1 - \left\lceil \lg(2^d - 1) \right\rceil / d$; and so on. A significant advance was made in Haussler and Welzl's seminal paper [32], which introduced probabilistic techniques to computational geometry; they presented a partition tree with $O(n^{1-1/[d(d-1)+1]+\varepsilon})$ query time for any fixed $\varepsilon > 0$, greatly improving all results that came before, in all dimensions.

The right exponent turned out to be $1 - 1/d$. Four major papers described (near-)linear-space methods achieving the final near-$O(n^{1-1/d})$ query time:

1. Welzl (SoCG'88) and Chazelle and Welzl [24, 48] devised the key technique of *iterative reweighting*[1] to compute *spanning trees with low crossing number*. They obtained an elegant $O(n)$-space data structure where the answer to any query can be expressed using $O(n^{1-1/d} \log n)$ arithmetic (semigroup) operations on the weights (if subtractions on weights are allowed, the number of operations can be reduced to $O(n^{1-1/d}\alpha(n))$). Unfortunately, this combinatorial bound on the number of arithmetic operations does not bound the actual query time. Chazelle and Welzl obtained true "algorithmic" results only in 2-d, with $O(n)$ space and $O(\sqrt{n} \log n)$ query time, and in 3-d, with $O(n \log n)$ space and $O(n^{2/3} \log^2 n)$ query time. The preprocessing time of these data structures is also poor; in 2-d, one can get $O(n^{3/2} \log^{1/2} n)$ preprocessing time by a randomized algorithm [48], but improvement to a near-linear bound requires further ideas from subsequent papers. Despite these issues, their approach was a remarkable breakthrough and set up the right direction for subsequent methods to follow.

2. Next, Chazelle, Sharir, and Welzl (SoCG'90) [23] applied *cuttings* [21, 27] to get efficient data structures for simplex range searching. In the $O(n)$-space case, their method has query time $O(n^{1-1/d+\varepsilon})$ with $O(n^{1+\varepsilon})$ preprocessing time for any fixed $\varepsilon > 0$. Although this brings us closer to the ideal bounds in any dimension, this method seems inherently suboptimal by at least a logarithmic factor (the data structure requires multiple separate cuttings, so as to guarantee that for every query, at least one of the cuttings is "good").

3. Matoušek (SoCG'91) [35] combined the iterative reweighting technique with the use of cuttings to prove a new *simplicial partition theorem*. Applying the theorem in a recursive fashion gives an $O(n)$-space partition tree with $O(n^{1-1/d} \log^{O(1)} n)$ query time and $O(n \log n)$ preprocessing time.

[1] Also called the "multiplicative weights update method" in some circles. In computational geometry, other favorite examples of the technique include one of Clarkson's linear programming algorithm [26] and Brönnimann and Goodrich's geometric set cover algorithm [13].

(The same paper also gives an alternative $O(n)$-space structure with query time $O(n^{1-1/d}(\log\log n)^{O(1)})$—or better still $O(n^{1-1/d}2^{O(\log^* n)})$ if subtractions of weights are allowed—but the preprocessing time increases to $O(n^{1+\varepsilon})$. This result is subsumed by the next result, so will be ignored here.)

4. Finally, Matoušek (SoCG'92) [38] combined iterative reweighting with the *hierarchical cuttings* of Chazelle [19] to obtain an $O(n)$-space partition tree with $O(n^{1-1/d})$ query time. The preprocessing time is $O(n^{1+\varepsilon})$. This method is considerably more complicated than Matoušek's previous partition-theorem-based method.

Among data structures that are ordinary (unaugmented) partition trees, it is not difficult to show that the query cost must be $\Omega(n^{1-1/d})$ in the worst case [24]. More generally, Chazelle [18] proved a lower bound of $\Omega(\sqrt{n})$ in 2-d and $\Omega(n^{1-1/d}/\log n)$ in higher dimensions, for any simplex range searching data structure under the semigroup arithmetic model (see [22] for another lower bound); the $\log n$ factor in the denominator could likely be an artifact of the proof.

Matoušek's final method thus seems to achieve asymptotically optimal query time. However, the story is not as neatly resolved as one would like. The main issue is preprocessing time: Matoušek's final method is suboptimal by an n^{ε} factor. This explains why in actual algorithmic applications of range searching results, where we care about preprocessing and query times combined, researchers usually abandon his final method and resort to his earlier partition-theorem method. Unfortunately, this is the reason why in algorithmic applications, we frequently see appearances of extra $\log^{O(1)} n$ factors in time bounds, with unspecified numbers of log factors.

Even in 2-d, no ideal data structure with $O(n\log n)$ preprocessing time, $O(n)$ space, and $O(\sqrt{n})$ query time has been found.

New results. In this paper, we give a new method that rederives Matoušek's final result of $O(n)$ space and $O(n^{1-1/d})$ query time for simplex range searching. Our new method is much simpler and easier to explain than Matoušek's final method (it is a little more involved than the partition-theorem method, but not considerably so). The partition tree satisfies many desirable properties not achieved by Matoušek's and, for example, eliminates essentially all the disadvantages mentioned in the next-to-last paragraph of his paper [38] (we use a single tree and have complete control over the crossing number at almost all layers of the tree). Our tree also satisfies properties not achieved by the partition-theorem method [35] (we use a constant-degree tree and cells do not overlap). In the 2-d case, we can even make our partition tree a BSP tree, like Willard's original partition tree. These properties in themselves are already interesting, in the author's opinion, but the approach has further major implications:

- Our method immediately yields improved results on *multilevel* versions of partition trees [7, 39], which are needed in applications that demand more sophisticated kinds of queries. With our method, each level costs at most *one* more logarithmic factor in space and at most *one* more logarithmic factor in the query time—this behavior is exactly what we want and, for example, is analogous to what we see in the classical multilevel data structure, the range tree [7, 44], which can be viewed as a d-level 1-dimensional partition tree.

  In contrast, Matoušek's final method is not applicable at all, precisely because of the disadvantages mentioned. Matoušek's final paper did explore multilevel results in detail but had to switch to a different method based on Chazelle, Sharir, and Welzl's cutting approach [23],

now with hierarchical cuttings. This method costs *two* logarithmic factors in space and *one* logarithmic factor in the query time per level (see [38, Theorem 5.2]). Furthermore, the preprocessing time is huge. (An alternative version [38, Corollary 5.2] has $O(n^{1+\varepsilon})$ preprocessing time, but the space and query time increases by an unspecified number of extra logarithmic factors.) Matoušek's partition-theorem method in the multilevel setting is worse, giving extra $O(n^{\varepsilon})$ (or more precisely $2^{O(\sqrt{\log n})}$) factors in space and query time per level [35].

For concrete applications, consider the problems of preprocessing n simplices in $\mathbb{R}^d$ so that we can (i) report all simplices containing a query point, or (ii) report all simplices contained inside a query simplex. These problems can be solved by a $(d+1)$-level partition tree. The previous method from [38] requires $O(n \log^{2d} n)$ space and $O(n^{1-1/d} \log^d n)$ query time. Our result reduces the space bound to $O(n \log^d n)$.

We get improved data structures for many other problems, even some traditional 2-d problems, e.g., counting the number of intersections of line segments with a query line segment, and performing ray shooting among a collection of line segments. (It is especially pleasurable to improve a few old results from Pankaj Agarwal's Ph.D. thesis [2].)

- That our method is simpler allows us to examine the preprocessing time more carefully. We show that with some extra effort, a variant of our method indeed accomplishes what we set out to find: a simplex range searching structure with $O(n \log n)$ preprocessing time, $O(n)$ space, and $O(n^{1-1/d})$ query time w.h.p.[2] The approach works in the multilevel setting as well, with one more logarithmic factor in the preprocessing time per level.

  The improved preprocessing time should immediately lead to improved time bounds for a whole array of algorithmic applications.

- We can also obtain new bounds for the *halfspace range emptiness/reporting* problem. Here, we want to decide whether a query halfspace contains any data point, or report all points inside a query halfspace. Matoušek [36] described a *shallow* version of his partition theorem to obtain a data structure with $O(n \log n)$ preprocessing time, $O(n \log \log n)$ space, and $O(n^{1-1/\lfloor d/2 \rfloor} \log^{O(1)} n + k)$ query time for any $d \geq 4$, where k denotes the output size in the reporting case. (The space was reduced to $O(n)$ by Ramos [45] for even d; see also [1] for $d = 3$.) The same paper by Matoušek also gave an alternative data structure with $O(n^{1+\varepsilon})$ preprocessing time, $O(n)$ space, and $O(n^{1-1/\lfloor d/2 \rfloor} 2^{O(\log^* n)})$ query time for halfspace range emptiness for any $d \geq 4$. Although it is conceivable that Matoušek's final method for simplex range searching with hierarchical cuttings could similarly be adapted to halfspace range emptiness for even d, to the best of the author's knowledge, no one has attempted to find out, perhaps daunted by the amount of technical details in the paper [38]. It is generally believed that the exponent $1 - 1/\lfloor d/2 \rfloor$ is tight, although a matching lower bound is currently not known.

  With our simpler method, we can derive the following, likely optimal result for even dimensions without too much extra trouble: there is a data structure for halfspace range emptiness with $O(n \log n)$ preprocessing time, $O(n)$ space, $O(n^{1-1/\lfloor d/2 \rfloor})$ query time w.h.p. for any even $d \geq 4$. The same bounds hold for halfspace range reporting when the output size k is small $(o(n^{1-1/\lfloor d/2 \rfloor} / \log^{\omega(1)} n))$.

[2] Throughout the paper, the notation "w.h.p.(n)" (or "w.h.p." for short, if n is understood) means "with probability at least $1 - 1/n^{c_0}$" for an arbitrarily large constant c_0.

Halfspace range emptiness/reporting has its own long list of applications [37, 41, 15]. As a consequence, we get improved results for ray shooting and linear programming queries in intersections of halfspaces in even dimensions, exact nearest neighbor search in odd dimensions, and the computation of extreme points and convex hulls in even dimensions, to name just a few. (It is also nice to see improvements of results from the author's own Ph.D. thesis.)

Organization. The highlight of the paper is to be found in Section 3—in particular, the proof of Theorem 3.1—which describes the basic version of our new partition tree with $O(n)$ space and $O(n^{1-1/d})$ query time. This part is meant to be self-contained (except for the side remarks at the end), assuming only the background provided by Section 2. The rest of the paper—particularly Section 5 on getting $O(n \log n)$ preprocessing time, Section 6 on halfspace range searching, and Section 7 on applications—will get a bit more technical and incorporate additional known techniques, but mostly involves working out consequences of the basic new approach.

2 Background

Let P be the given set of n points in $\mathbb{R}^d$, which we assume to be in general position (by standard perturbation techniques). For our purposes, we define a *partition tree* T to be any tree where each leaf stores at most a constant number of points, each point of P is stored in exactly one leaf, and each node v stores a cell $\Delta(v)$ which encloses the subset $P(v)$ of all points stored at the leaves underneath v. Throughout the paper, we take "cell" to mean a constant-size polyhedron, or to be more specific, a simplex. In general, the degree of a node is allowed to be nonconstant and the cells of the children of a node are allowed to overlap (although in our new basic method, we can guarantee constant degree and no overlaps of the cells). At each node v, we do not store any auxiliary information about $P(v)$, other than the sum of the weights in $P(v)$. Therefore, a partition tree is by our definition always an $O(n)$-space data structure.

We can answer a query for a simplex q by proceeding recursively, starting at the root. When visiting a node, we only have to recurse in children whose cells cross the boundary ∂q, since each cell completely outside q can be ignored, and each cell completely inside q can be dealt with directly using the information stored at the node (or, in the case of the reporting problem, directly reporting all points underneath the node). We can identify the children cells crossing ∂q trivially in time linear in the number of children (the degree). The total query time is then upper-bounded by the expression

$$\sum_{v \in T:\ \Delta(v) \cap \partial q \neq \emptyset} \deg(v), \tag{1}$$

which we call the *query cost of* ∂q. (In the case of the reporting problem, we add an $O(k)$ term to the query time where k is the output size.) Since the boundary ∂q has $O(1)$ facets contained in $O(1)$ hyperplanes, it suffices to asymptotically bound the maximum query cost of an arbitrary hyperplane, which we refer to as the *query cost of the tree* T.

It is known [24] that there are point sets where the query cost is $\Omega(n^{1-1/d})$ for any partition tree under the above definitions. We want to devise a worst-case optimal method to construct partition trees matching this lower bound.

We begin by mentioning Matoušek's simplicial partition theorem [35], to set the context. (We will not use this theorem anywhere, but will prove something stronger in Section 3.)

Theorem 2.1 (Matoušek's Partition Theorem) *Let P be a set of n points in $\mathbb{R}^d$. Then for any t, we can partition P into t subsets P_i and find t cells Δ_i, with $\Delta_i \supset P_i$, such that each subset contains $\Theta(n/t)$ points and each hyperplane crosses at most $O(t^{1-1/d})$ cells.*

A recursive application of the above theorem immediately gives a partition tree where the query cost satisfies the recurrence $Q(n) \leq O(t^{1-1/d})Q(n/t)+O(t)$. If we set t to a sufficiently large constant, the recurrence has solution $Q(n) \leq O(n^{1-1/d+\varepsilon})$ for any constant $\varepsilon > 0$. If instead we set $t = n^\delta$ for a sufficiently small constant $\delta > 0$, the solution becomes $Q(n) \leq O(n^{1-1/d} \log^{O(1)} n)$. To avoid the extra n^ε or $\log^{O(1)} n$ factors, we need a more refined method, since each time we recurse using Theorem 2.1, the hidden constant in the crossing number bound "blows up".

Our new method is described in the next section, and is self-contained except for the use of two tools: a minor observation and an important lemma, both well-known and stated here without proofs.

First, we observe that in proving statements like Theorem 2.1, instead of considering an infinite number of possible hyperplanes, it suffices to work with a finite set of "test" hyperplanes. This type of observations was made in many of the previous papers [23, 24]; the particular version we need is a special case of one from [35]:

Observation 2.2 (Test Set) *Given a set P of n points in $\mathbb{R}^d$, we can construct a set H of $n^{O(1)}$ hyperplanes with the following property: For any collection of disjoint cells each containing at least one point of P, if the maximum number of cells crossed by a hyperplane in H is ℓ, then the maximum number of cells crossed by an arbitrary hyperplane is $O(\ell)$.*

For this version of the test-set observation, the proof is quite simple (briefly, we can just take H to be the $O(n^d)$ hyperplanes passing through d-tuples of points of P).

For the next lemma, we need a definition first. For a given set H of m hyperplanes, a $(1/r)$-*cutting* [20, 40, 43] is a collection of disjoint cells such that each cell Δ is crossed by at most m/r hyperplanes. Let $\mathcal{A}(H)$ denote the arrangement of H. We let H_Δ denote the subset of all hyperplanes of H that cross Δ.

Lemma 2.3 (Cutting Lemma) *Let H be a set of m hyperplanes and Δ be any simplex in $\mathbb{R}^d$. For any r, we can find a $(1/r)$-cutting of H into $O(r^d)$ disjoint cells whose union is Δ.*

If the number of vertices in $\mathcal{A}(H)$ inside Δ is X, then the number of cells can be reduced to $O(X(r/m)^d + r^{d-1})$.

The first part of the lemma is standard and was first shown by Chazelle and Friedman [21]. The X-sensitive variant is also known; e.g., see [12, 19]. The proof is by random sampling.[3]

To understand the intuitive significance of the cutting lemma, let H be a set of m test hyperplanes, $\Delta = \mathbb{R}^d$, and $r = t^{1/d}$. Then we get $O(t)$ cells each crossed by $O(m/t^{1/d})$ hyperplanes. The average

[3] (Very Rough) Proof Sketch: We draw a sample R of size r and take the cells of a *canonical triangulation* T of $\mathcal{A}(R)$ [25] restricted inside Δ. For a cell $\Delta \in T$ that is crossed by am/r hyperplanes with $a > 1$, we further subdivide the cell by taking a $(1/a)$-cutting of H_Δ using any weaker method with, say, $a^{O(1)}$ subcells; e.g., we can again use random sampling. To see why this procedure works, note that the size of the canonical triangulation is proportional to the size of $\mathcal{A}(R)$ restricted inside Δ, which is at most $O(X(r/m)^d + r^{d-1})$, since each vertex in $\mathcal{A}(H)$ shows up in $\mathcal{A}(R)$ with probability $O(r/m)^d$, and there are $O(r^{d-1})$ vertices in the intersection of $\mathcal{A}(R)$ with each $(d-1)$-dimensional boundary facet of Δ. On the other hand, analysis by Clarkson and Shor [27] or Chazelle and Friedman [21] tells us that the parameter a is $O(1)$ "on average" over all cells $\Delta \in T$.

number of points per cell is $O(n/t)$, and the average number of cells crossed by a test hyperplane is $O((t \cdot m/t^{1/d})/m) = O(t^{1-1/d})$. Thus, the cutting lemma "almost" implies the partition theorem. The challenge is to turn these average bounds into maximum bounds. For this purpose, Matoušek's proof of his partition theorem adopts an *iterative reweighting* strategy by Welzl [24, 48] (roughly, in each iteration, we apply the cutting lemma to a multiset of hyperplanes in H to find one good cell, then increase the multiplicities of the hyperplanes crossing the cell, and repeat).

3 The Basic Method

We now present our new method for simplex range searching with $O(n)$ space and $O(n^{1-1/d})$ query time. To keep the presentation simple, we defer discussion of preprocessing time to Section 5.

The key lies in the following theorem, where instead of constructing one partition of the point set from scratch as in Matoušek's partition theorem, we consider the problem of *refining* a given partition. As we will see, the main result will follow just by repeated applications of our theorem in a straightforward manner.

Theorem 3.1 (Partition Refinement Theorem) *Let P be a set of at most n points and H be a set of m hyperplanes in $\mathbb{R}^d$. Suppose we are given t disjoint cells covering P, such that each cell contains at most $2n/t$ points of P and each hyperplane in H crosses at most ℓ cells. Then for any b, we can subdivide every cell into $O(b)$ disjoint subcells, such that each subcell contains at most $n/(bt)$ points of P, and each hyperplane in H crosses at most the following total number of subcells:*

$$O((bt)^{1-1/d} + b^{1-1/(d-1)}\ell + b \log t \log m). \tag{2}$$

Note that the first term matches the bound from Matoušek's partition theorem for a partition into $O(bt)$ parts. In the second term, it is crucial that the coefficient $b^{1-1/(d-1)}$ is asymptotically smaller than $b^{1-1/d}$; this will ensure that repeated applications of the theorem do not cause a constant-factor blow-up, if we pick b to be sufficiently large. The third term will not matter much in the end, and is purposely not written in the tightest manner. The proof of the theorem (like Matoušek's) uses the cutting lemma repeatedly in conjunction with an iterated reweighting strategy as detailed below. The basic algorithm is quite elegant, in the author's opinion:

Proof: We maintain a multiset $\widehat{H}$, initially containing C copies of each hyperplane in H. The value of C is not important (for conceptual purposes, we can set C to be a sufficiently large power of b, which will ensure that future multiplicities are always integers). The size $|\widehat{H}|$ of a multiset H always refers to the sum of the multiplicities (the "weights") of the elements. Let $X_\Delta(\widehat{H})$ denote the number of vertices inside Δ defined by the hyperplanes in $\widehat{H}$ (the multiplicity of a vertex is the product of the multiplicities of its defining hyperplanes).

The algorithm. Let $\Delta_t, \ldots, \Delta_1$ be the given cells in a random order. For $i = t, \ldots, 1$ do:

1. Subdivide Δ_i into disjoint subcells by building a $(1/r)$-cutting of $\widehat{H}_{\Delta_i}$ inside Δ_i with

$$r \;:=\; \min\left\{ |\widehat{H}_{\Delta_i}| \left(\frac{b}{X_{\Delta_i}(\widehat{H})} \right)^{1/d},\; b^{1/(d-1)} \right\}.$$

By Lemma 2.3, the number of subcells is $O(X_{\Delta_i}(\widehat{H})(r/|\widehat{H}_{\Delta_i}|)^d + r^{d-1}) = O(b)$.

2. Further subdivide each subcell, e.g., by using vertical cuts, to ensure that each subcell contains at most $n/(bt)$ points. At most $O(b)$ extra cuts are sufficient, so the number of subcells inside Δ_i remains $O(b)$.

3. For each hyperplane h, multiply the multiplicity of h in $\widehat{H}$ by $(1+1/b)^{\lambda_i(h)}$ where $\lambda_i(h)$ denotes the number of subcells inside Δ_i crossed by h.

Analysis. For a fixed hyperplane $h \in H$, let $\ell_i(h)$ be the number of cells in $\{\Delta_i, \ldots, \Delta_1\}$ crossed by h. Since $\{\Delta_i, \ldots, \Delta_1\}$ is a random subset of size i from a set of size t, we have $\ell_i(h) \le O(\ell i/t + \log(mt))$ w.h.p.(mt) by a Chernoff bound (a version for sampling without replacement, as noted in Appendix A). So, defining $\ell_i := \max_{h\in H} \ell_i(h)$, we have $\ell_i \le O(\ell i/t + \log(mt))$ w.h.p.(t).

In step 3, $\sum_{h\in\widehat{H}} \lambda_i(h) \le O(b|\widehat{H}_{\Delta_i}|/r)$, since each of the $O(b)$ subcells is crossed by at most $|\widehat{H}_{\Delta_i}|/r$ hyperplanes of $\widehat{H}_{\Delta_i}$. As $\lambda_i(h) \le O(b)$, step 3 increases $|\widehat{H}|$ by

$$\sum_{h\in\widehat{H}} [(1+1/b)^{\lambda_i(h)} - 1] \;\le\; O\left(\sum_{h\in\widehat{H}} \lambda_i(h)/b\right) \;\le\; O(|\widehat{H}_{\Delta_i}|/r) \;=\; O(\alpha_i)|\widehat{H}|$$

$$\text{where}\;\; \alpha_i \;:=\; \frac{|\widehat{H}_{\Delta_i}|}{r|\widehat{H}|} \;=\; O\left(\frac{1}{|\widehat{H}|}\cdot\left(\frac{X_{\Delta_i}(\widehat{H})}{b}\right)^{1/d} + \frac{|\widehat{H}_{\Delta_i}|}{b^{1/(d-1)}|\widehat{H}|}\right).$$

Note that $\sum_{j=1}^{i} |\widehat{H}_{\Delta_j}| \le |\widehat{H}|\ell_i$ and $\sum_{j=1}^{n} X_{\Delta_j}(\widehat{H}) \le |\widehat{H}|^d$ by disjointness of the cells. Conditioned to a fixed choice of $\Delta_t, \ldots, \Delta_{i+1}$, since Δ_i is a random element among the remaining i cells, we have $E[|\widehat{H}_{\Delta_i}|] \le |\widehat{H}|\ell_i/i$ and $E[X_{\Delta_i}(\widehat{H})] \le |\widehat{H}|^d/i$, and in particular, $E[X_{\Delta_i}(\widehat{H})^{1/d}] \le |\widehat{H}|/i^{1/d}$, so

$$E[\alpha_i] \;\le\; O\left(\frac{1}{(bi)^{1/d}} + \frac{\ell_i}{b^{1/(d-1)}i}\right).$$

Unconditionally, since $E[\ell_i] \le O(\ell i/t + \log(mt))$,

$$E[\alpha_i] \;\le\; O\left(\frac{1}{(bi)^{1/d}} + \frac{\ell}{b^{1/(d-1)}t} + \frac{\log(mt)}{b^{1/(d-1)}i}\right).$$

At the end, $|\widehat{H}| \;\le\; Cm\prod_{i=1}^{t}(1+O(\alpha_i)) \;\le\; Cm\,\exp\left(O\left(\sum_{i=1}^{t}\alpha_i\right)\right)$, where

$$E\left[\sum_{i=1}^{t}\alpha_i\right] \;\le\; O\left(\frac{t^{1-1/d}}{b^{1/d}} + \frac{\ell}{b^{1/(d-1)}} + \frac{\log(mt)\log t}{b^{1/(d-1)}}\right).$$

Let $\lambda(h)$ be the total number of subcells crossed by h. Since the final multiplicity of h in $\widehat{H}$ is equal to $C(1+1/b)^{\lambda(h)} \le |\widehat{H}|$, it follows that $\max_{h\in H}\lambda(h) \le \log_{1+1/b}(|\widehat{H}|/C) \le O(b\log(|\widehat{H}|/C))$, which has expected value at most $O(b\log m + (bt)^{1-1/d} + b^{1-1/(d-1)}\ell + b^{1-1/(d-1)}\log(mt)\log t)$, which is upper-bounded by (2). □

Theorem 3.2 *Given n points in $\mathbb{R}^d$, we can build a partition tree with query cost $O(n^{1-1/d})$.*

Proof: In what follows, summations involving the variable t are over all t that are powers of 2. Let H be the test set from Observation 2.2 of size $m \le n^{O(1)}$. Initially, our partition tree consists of just a root cell containing all n points.

The algorithm. For $t = 2, 4, 8, \ldots$ do:

Let Π_t be the set of all current leaf cells Δ such that the number of points inside Δ is between n/t and $2n/t$; the number of such cells is at most t. Apply Theorem 3.1 to Π_t to get a set Π'_t of subcells where the number of points inside each subcell is at most $n/(bt)$. Make the $O(b)$ subcells of each cell $\Delta \in \Pi_t$ the children of Δ in our partition tree (Δ is no longer a leaf but its children now are).

Analysis. Let $\ell(t)$ denote the maximum number of cells of Π_t crossed by an arbitrary hyperplane. Let $\ell'(t)$ denote the maximum number of cells of Π'_t crossed by a hyperplane in H, which is bounded by (2). The maximum number of cells of Π_u crossed by a hyperplane in H is at most $\sum_{t:\ bt \le u} \ell'(t)$. By Observation 2.2, $\ell(u)$ is asymptotically bounded by the same expression. We obtain the following recurrence:

$$\ell(u) \ \le \ O\left(\sum_{t:\ bt\le u} [(bt)^{1-1/d} + b^{1-1/(d-1)}\ell(t) + b\log t \log n]\right). \tag{3}$$

Fix an arbitrarily small constant $\delta > 0$. Set b to be a sufficiently large constant depending on δ. It is a straightforward exercise to verify by induction that

$$\ell(u) \ \le \ c[u^{1-1/d} + u^{1-1/(d-1)+\delta}\log n]$$

for some constant c depending on δ and b. Note that the first term dominates when u exceeds $\log^{c'} n$ for some constant c', so we can write $\ell(u) \le O(u^{1-1/d} + \log^{O(1)} n)$. (For the induction proof, it is helpful to note that sums like $\sum_t (bt)^{1-1/d}$ form geometric series and are asymptotically bounded by their last term, as summations are over t's that are powers of 2.)

We conclude that the query cost is at most $O\left(\sum_{t\le n} b\,\ell(t)\right) \ \le \ O\left(\sum_{t\le n}[t^{1-1/d} + \log^{O(1)} n]\right) \ = \ O(n^{1-1/d})$. □

Remarks. To readers familar with the proof of Matoušek's partition theorem [35], we point out some differences with our proof of the partition refinement theorem. First, to ensure $\sum_j X_{\Delta_j}(\widehat{H}) \le |\widehat{H}|^d$, we need disjointness of cells, which forces us to choose multiple subcells from a cutting instead of one subcell per iteration. This in turn forces us to use a different multiplier $(1 + 1/b)$ instead of doubling. Unlike Theorem 2.1, Theorem 3.1 guarantees an upper but not a lower bound on size of each subset (certain versions of the test set observation, e.g., Observation 5.1, require such a lower bound), though this is not an issue in our proof of Theorem 3.2.

To those familiar with Matoušek's final method based on hierarchical cuttings [38], we note that our method share certain common ideas. Both apply iterative reweighting more "globally" to refine multiple cells simultaneously, as a way to avoid constant-factor blow-up in the partition-theorem method; and hierarchical cuttings originate from the same X-sensitive version of the cutting lemma. However, Matoušek's method uses a far more complicated weight/multiplicity function, because it tries to deal with different layers of the tree all at once; in contrast, our partition refinement theorem works with one layer at a time and leads to a simpler, more modular design. Matoušek's method also requires a special root of $O(n^{1/d}\log n)$ degree (whose children cells may overlap), and does not guarantee optimality for $\ell(t)$ except at the bottommost layer, whereas our method ensures optimality for almost all layers (except near the very top for very small t, which is not important). This will make a difference in the next section and in subsequent applications to multilevel data structures.

Random ordering is convenient but is not the most crucial part of the proof of Theorem 3.1: We can easily deterministically find a good cell Δ_i in each iteration if we do not care about the bound on ℓ_i. The naive bound $\ell_i \leq \ell$ can still lead to a weaker version of Theorem 3.1 with an extra logarithmic factor in the second term of (2), and Theorem 3.2 can still be derived but with some extra effort. Alternatively, we can derandomize (hint: consider the potential function $\sum_{h \in \widehat{H}} 2^{c\ell_i(h)t/i}$).

In the 2-d case, our partition tree can be made into a BSP tree, since the cuttings produced by taking canonical triangulations of the arrangements of samples are easily realizable as binary plane partitions (our algorithm computes such cuttings for $r \leq b^{O(1)}$ bounded by a constant).

4 Additional Properties

We point out some implications that follow from our method. Define the *order-γ query cost* of ∂q to be $\sum_{v \in T:\ \Delta(v) \cap \partial q = \emptyset} \sum_{w \text{ child of } v} |P(w)|^\gamma$. Define the order-$\gamma$ query cost of T to be the maximum order-γ query cost of an arbitrary hyperplane. Our previous definition coincides with the case $\gamma = 0$. This extended definition is relevant in multilevel data structures where secondary data structures for $P(v)$ are stored at each node v (see Section 7 for more details). We have the following new result:

Theorem 4.1 *Given n points in $\mathbb{R}^d$, we can build a partition tree with*

(i) *height $O(\log n)$ and order-$(1-1/d)$ query cost $O(n^{1-1/d} \log n)$;*

(ii) *height $O(\log \log n)$ and order-γ query cost $O(n^{1-1/d})$ for any fixed $\gamma < 1 - 1/d$.*

Proof:

(i) The same partition tree in the proof of Theorem 3.2 has order-$(1-1/d)$ query cost at most

$$O\left(\sum_{t \leq n} b \left(\frac{n}{bt}\right)^{1-1/d} \ell(t)\right) \;\leq\; O\left(\sum_{t \leq n} (n/t)^{1-1/d} \cdot [t^{1-1/d} + t^{1-1/(d-1)+\delta} \log n]\right)$$
$$= \; O(n^{1-1/d} \log n + n^{1-1/d} \log n) \;=\; O(n^{1-1/d} \log n).$$

(ii) To lower the height, we modify our partition tree T. Fix a sufficiently small constant $\varepsilon > 0$. We examine each node v of T in a top-down order, and whenever a child w of a node v has $|P(w)| > |P(v)|^{1-\varepsilon}$, we remove w by making w's children v's children. In the new tree T', every child w of a node v has $|P(w)| \leq |P(v)|^{1-\varepsilon}$, so the height is $O(\log \log n)$. For each child w of v in T', w's parent in T has size at least $|P(v)|^{1-\varepsilon}$, so the degree of v in T' is at most $b|P(v)|^\varepsilon \leq O((n/t)^\varepsilon)$ if $|P(v)| = \Theta(n/t)$. The order-γ query cost is at most

$$O\left(\sum_{t \leq n} (n/t)^\varepsilon \left((n/t)^{1-\varepsilon}\right)^\gamma \ell(t)\right) \;\leq\; O\left(\sum_{t \leq n} (n/t)^{\gamma+\varepsilon} \cdot [t^{1-1/d} + \log^{O(1)} n]\right)$$
$$= \; O(n^{1-1/d} + n^{\gamma+\varepsilon} \log^{O(1)} n) \;=\; O(n^{1-1/d}).$$

□

We have omitted the case $\gamma > 1 - 1/d$, because a simple recursive application of Matoušek's partition theorem already gives optimal order-γ query cost $O(n^\gamma)$ with height $O(\log \log n)$ in this case.

Next, define a *B-partial partition tree* for a point set P to be the same as in our earlier definition of a partition tree, except that a leaf now may contain up to B points. Define the *query cost* in the same manner, as in (1). Note the query cost of a partial partition tree upper-bounds the number of internal and leaf cells crossed by a hyperplane, but does not account for the cost of any auxiliary data structures we plan to store at the leaves. The following theorem is useful in applications that need space/time tradeoffs, where we can switch to a larger-space data structure with small query time at the leaves (see Section 7 for more details). The theorem is also self-evidently useful in the external memory model (hence, the choice of name "B"): we get improved external-memory range searching data structures [5] immediately as a byproduct.

Theorem 4.2 *Given n points in $\mathbb{R}^d$ and $B < n/\log^{\omega(1)} n$, we can build a B-partial partition tree with size $O(n/B)$ and query cost $O(n/B)^{1-1/d}$.*

Proof: Stop the algorithm in the proof of Theorem 3.2 when t reaches n/B. As a result, we get $O(bn/B)$ leaf cells, each containing at most B points. The maximum number of internal node cells crossed by an arbitrary hyperplane is at most $O(\sum_{t \le n/B} \ell(t)) = O(n/B)^{1-1/d}$. The maximum number of leaf cells crossed by an arbitrary hyperplane is at most b times this number. Recall that the tree has degree $O(b)$, which is a constant. □

5 Preprocessing Time

In this section, we examine the issue of preprocessing time. Obtaining polynomial time is easy with our method, and it might be possible to lower the bound to $O(n^{1+\varepsilon})$ by using recursion, as was done in Matoušek's final method [38]. Instead of recursion, we show how to directly speed up the algorithm in Section 3. For now, we aim for $O(n \log^{O(1)} n)$ preprocessing time; later, we show how to improve it to $O(n \log n)$.

First, we need a better version of Observation 2.2 with a smaller number of test hyperplanes. The following observation was shown by Matoušek [35] (see also [43]).

Observation 5.1 (Test Set) *Given a set P of n points and any t, we can construct a set H_t of $O(t \log^{O(1)} N)$ test hyperplanes, in $O(t \log^{O(1)} N)$ time, satisfying the following property w.h.p.(N) for any $N \ge n$: For any collection of disjoint cells each containing at least n/t points of P, if the maximum number of cells crossed by a hyperplane in H_t is ℓ, then the maximum number of cells crossed by an arbitrary hyperplane is at most $O(\ell + t^{1-1/d})$.*

(Roughly interpreted, the proof involves drawing a random sample of $t^{1/d} \log N$ points and taking the $O(t \log^d N)$ hyperplanes through d-tuples of points in the sample; the time bound is clearly as stated. The $\log N$ factors can be removed with more work, using cuttings, but will not matter.)

In the proof of Theorem 3.2, we can take H to be the union of the test sets H_t from Observation 5.1 over all t (powers of 2); the total size is $m = O(n \log^{O(1)} N)$. This causes an extra term $O(u^{1-1/d})$ in the recurrence (3), but the solution is unaffected.

Regarding Lemma 2.3, we can use known cutting algorithms [21, 35, 19] to get running time $O(mr^{O(1)})$ (expected, or even worst-case).

To get the best time bound for Theorem 3.1, we need to modify the algorithm. The details become more involved, but there are two new main ideas:

- in step 1, we work with a sparser subset of $\widehat{H}$, by considering a random sample of $\widehat{H}$;
- more crucially, in step 3, we update the multiplicities less frequently, by considering a random sample of the subcells.

To prepare for the proof of the theorem below, we make two definitions, the first of which is standard: Given a (multi)set S, let a *p-sample* be a subset R generated by taking each (occurrence of an) element of S, and independently choosing to put the element in R with probability p. (Note that we can generate a p-sample in time linear in the output size rather than the size of S, by using a sequence of exponentially distributed random variables rather than 0-1 random variables.)

For a list $S = \langle s_1, \ldots, s_K \rangle$ and a sublist R of S, we say that R is a *generalized p-sample of S* if the events $E_i = \{s_i \in R\}$ are all independent. Note the subtle difference with the earlier definition: here, the list S (the elements s_i and the size K) is allowed to be random; it is acceptable for s_i to be dependent on $E_1, \ldots, E_{i-1}$, as long as we decide to choose whether to put s_i to R independently of $E_1, \ldots, E_{i-1}$ and s_i.

Properties enjoyed by standard random samples may not necessarily hold for generalized p-samples. However, the Chernoff bound is still clearly applicable to show that $|R \cap \{s_1, \ldots, s_k\}| \leq O(pk + \log N)$ and $k \leq O(1/p)(|R \cap \{s_1, \ldots, s_k\}| + \log N)$ for all $k \leq N$ w.h.p.(N). In particular, assuming $K \leq N$, we have $|R| \leq O(p|S| + \log N)$ and $|S| \leq O(1/p)(|R| + \log N)$ w.h.p.(N).

We now present a near-linear-time algorithm for the partition refinement theorem:

Theorem 5.2 *In Theorem 3.1, given the subset of points inside each cell, we can construct a subdivision into subcells, together with the subset of points inside each subcell, in time*

$$O(bn + b^{O(1)}(m+t)\log^{O(1)} N),$$

such that the stated properties are satisfied, except that the crossing number bound is now $O((bt)^{1-1/d} + b^{1-1/(d-1)}\ell + b \log t \log N)$ w.h.p.(N) for any $N \geq mt$.

Proof: *The algorithm.* Let $\Delta_t, \ldots, \Delta_1$ be the given cells in a random order. For $i = t, \ldots, 1$ do:

0. Let q be a power of 2 (with a negative integer exponent) closest to $\min\{\frac{(bi)^{1/d}\log N}{|\widehat{H}|}, \frac{b^{1/(d-1)}t\log N}{|\widehat{H}|\ell}, \frac{b^{1/(d-1)}i}{|\widehat{H}|}\}$. If q is different from its value in the previous iteration, then create a new q-sample $\widehat{R}$ of $\widehat{H}$.

1. Subdivide Δ_i into disjoint subcells by building a $(1/r)$-cutting of $\widehat{R}_{\Delta_i}$ inside Δ_i with

$$r \;:=\; \min\left\{|\widehat{R}_{\Delta_i}|\left(\frac{b}{X_{\Delta_i}(\widehat{R})}\right)^{1/d},\; b^{1/(d-1)}\right\}.$$

   By Lemma 2.3, the number of subcells is $O(b)$.

2. Further subdivide each subcell by using extra cuts to ensure that each subcell contains at most $n/(bt)$ points. The number of subcells inside Δ_i remains $O(b)$.

3. Take a p-sample ρ_i of the subcells of Δ_i with $p := \min\{\frac{b^{1/d}\log N}{t^{1-1/d}}, \frac{b^{1/(d-1)}\log N}{\ell}, \frac{b^{1/(d-1)}}{\log t}, 1\}$. If $\rho_i \neq \emptyset$, then

(a) for each $h \in H_{\Delta_i}$, add new copies of h to $\widehat{H}$ so that the multiplicity of h in $\widehat{H}$ gets multiplied by $(1+1/b)^{|\rho_i(h)|}$, where $\rho_i(h)$ denotes the set of subcells in ρ_i crossed by h;

(b) insert a q-sample of the newly added hyperplanes of $\widehat{H}$ to $\widehat{R}$.

Analysis of the crossing number. Let $\widehat{R}^{(q)}$ denote the set $\widehat{R}$ in the duration when the power of 2 closest to $\min\{\frac{(bi)^{1/d}\log N}{|\widehat{H}|},\ \frac{b^{1/(d-1)}t\log N}{|\widehat{H}|\ell},\ \frac{b^{1/(d-1)}i}{|\widehat{H}|}\}$ is equal to q. For any fixed q, observe that $\widehat{R}^{(q)}$ is a generalized q-sample of $\widehat{H}$ if we view the multiset $\widehat{H}$ as a list where the hyperplanes are listed in order they are added to $\widehat{H}$. Similarly, for any fixed simplex σ, $\widehat{R}^{(q)}_\sigma$ is a generalized q-sample of $\widehat{H}_\sigma$. Call two simplices σ_1 and σ_2 equivalent if $H_{\sigma_1} = H_{\sigma_2}$. The number of equivalence classes is polynomially bounded in N. Since $|\widehat{H}|$ is loosely upper-bounded by $Cm(1+1/b)^{O(bt)}$ at all times, the total number of different q's encountered is polynomially bounded in N. By the Chernoff bound, we can thus guarantee that the following holds at all times for all simplices σ and all q w.h.p.(N):

$$|\widehat{H}_\sigma| \le O(1/q)(|\widehat{R}^{(q)}_\sigma| + \log N) \qquad \text{and} \qquad |\widehat{R}| \le O(q|\widehat{H}| + \log N).$$

After step 1, w.h.p., for each subcell σ, we have $|\widehat{H}_\sigma| \le O(1/q)(|\widehat{R}_\sigma| + \log N) \le O(1/q)(|\widehat{R}_{\Delta_i}|/r + \log N)$, implying that $\sum_{h\in\widehat{H}} |\rho_i(h)| \le |\rho_i| \cdot O(1/q)(|\widehat{R}_{\Delta_i}|/r + \log N)$. W.h.p., step 3(a) increases $|\widehat{H}|$ by the following amount:

$$\begin{aligned}
\sum_{h\in\widehat{H}} [(1+1/b)^{|\rho_i(h)|} - 1] \ &\le\ O\left(\sum_{h\in\widehat{H}} |\rho_i(h)|/b\right) \ \le\ \frac{|\rho_i|}{bq}\cdot O(|\widehat{R}_{\Delta_i}|/r + \log N) \\
&\le\ \frac{|\rho_i|}{bq}\cdot O(\alpha_i|\widehat{R}| + \log N) \quad \text{where} \ \ \alpha_i \ :=\ \frac{|\widehat{R}_{\Delta_i}|}{r|\widehat{R}|} \ =\ O\left(\frac{1}{|\widehat{R}|}\cdot\left(\frac{X_{\Delta_i}(\widehat{R})}{b}\right)^{1/d} + \frac{|\widehat{R}_{\Delta_i}|}{b^{1/(d-1)}|\widehat{R}|}\right) \\
&\le\ \frac{|\rho_i|}{bq}\cdot O(\alpha_i q|\widehat{H}| + \log N) \ \le\ O(\beta_i)|\widehat{H}| \quad \text{where} \ \ \beta_i \ :=\ \frac{|\rho_i|}{b}\left(\alpha_i + \frac{\log N}{q|\widehat{H}|}\right).
\end{aligned}$$

Let ℓ_i be as before. Conditioned to a fixed choice of $\Delta_t, \ldots, \Delta_{i+1}$ and $\widehat{R}$, we have $E[|\widehat{R}_{\Delta_i}|] \le |\widehat{R}|\ell_i/i$ and $E[X_{\Delta_i}(\widehat{R})] \le |\widehat{R}|^d/i$, so $E[\alpha_i] \ \le\ O(1/(bi)^{1/d} + \ell_i/(b^{1/(d-1)}i))$. Furthermore, $E[|\rho_i|] \le O(bp)$ conditioned to any fixed value of α_i, so $E[\beta_i] \le p\cdot O((1/(bi)^{1/d} + \ell_i/(b^{1/(d-1)}i) + (\log N)/(q|\widehat{H}|))$. Unconditionally, since $E[\ell_i] \le O(\ell i/t + \log N)$,

$$E[\beta_i] \ \le\ p\cdot O\left(\frac{1}{(bi)^{1/d}} + \frac{\ell}{b^{1/(d-1)}t} + \frac{\log N}{b^{1/(d-1)}i} + \frac{\log N}{q|\widehat{H}|}\right).$$

The last term disappears for our choice of q.

At the end, w.h.p., we can upper-bound $|\widehat{H}|$ by $Cm\prod_{i=1}^t(1+O(\beta_i)) \le Cm\,\exp\left(O\left(\sum_{i=1}^t \beta_i\right)\right)$, where

$$E\left[\sum_{i=1}^t \beta_i\right] \ \le\ p\cdot O\left(\frac{t^{1-1/d}}{b^{1/d}} + \frac{\ell}{b^{1/(d-1)}} + \frac{\log N\log t}{b^{1/(d-1)}}\right) \ \le\ O(\log N)$$

for our choice of p. By Markov's inequality, with probability $\Omega(1)$, we have $\sum_{i=1}^t \beta_i \le\ O(\log N)$ and $|\widehat{H}| \le CN^{O(1)}$. If not, we declare failure.

Let $\rho(h) = \bigcup_i \rho_i(h)$. Since the multiplicity of h in $\widehat{H}$ is $C(1+1/b)^{|\rho(h)|} \le |\widehat{H}|$, it follows that $\max_{h\in H}|\rho(h)| \le \log_{1+1/b}(|\widehat{H}|/C) \le O(b\log N)$ if algorithm does not fail. Observe that $\rho(h)$ is a

generalized p-sample of the list $\lambda(h)$ of all subcells crossed by h, where the subcells are listed in the order they are created. Thus, w.h.p., we have $\max_{h\in H} |\lambda(h)| \leq O((1/p)(|\rho(h)| + \log N)) \leq O((b/p)\log N) \leq O((bt)^{1-1/d} + b^{1-1/(d-1)}\ell + b^{1-1/(d-1)} \log t \log N + b \log N)$ if the algorithm does not fail.

Analysis of running time. Conditioned to a fixed choice of $\Delta_t, \ldots, \Delta_{i+1}$ and $\widehat{R}$, we have $E[|\widehat{R}_{\Delta_i}|] \leq |\widehat{R}|\ell_i/i$. Unconditionally, $E[|\widehat{R}_{\Delta_i}|] \leq (q|\widehat{H}| + \log N) \cdot O(\ell i/t + \log N)/i \leq (b \log N)^{O(1)}$ for our choice of q. Assuming that $\widehat{R}_{\Delta_i}$ is available, step 1 takes time $|\widehat{R}_{\Delta_i}|r^{O(1)}$, which has expected value $(b \log N)^{O(1)}$. We don't need to compute $X_{\Delta_i}(\widehat{R})$, but rather we try different values of r (powers of 2) until we find a cutting with the right number of subcells. Thus, the expected cost over all iterations is $O(t)(b \log N)^{O(1)}$. By Markov's inequality, with probability $\Omega(1)$, this bound holds; if not, we declare failure.

In step 2, we can check the number of points per subcell by assigning points to subcells in $O(b)$ time per point, for a total cost of $O(bn)$. We can actually make the cost sublinear in n—namely, $O(t)(b \log N)^{O(1)}$—by replacing P by a $\frac{bt \log N}{n}$-sample Q of P. W.h.p., $|Q| = O(bt \log N)$, and for any simplex σ, $|Q \cap \sigma| \leq |Q|/(bt)$ implies $|P \cap \sigma| \leq O(n/(bt))$ by a Chernoff bound.

For step 3(a), we first need to compute the set H_{Δ_i}, i.e., report all hyperplanes of H intersecting a query cell Δ_i. In the dual, H becomes an m-point set and Δ_i becomes a constant-size polyhedron, and this subproblem reduces to simplex range searching. We can use existing data structures, e.g., [35], to get $O(m \log m)$ preprocessing time and $O(m^{1-1/d} \log^{O(1)} m + |H_{\Delta_i}|)$ query time. Since $\sum_i |H_{\Delta_i}| \leq m\ell$ and the probability that $\rho_i \neq \emptyset$ is at most $O(bp)$, the total expected query time is $bp \cdot O(tm^{1-1/d} \log^{O(1)} m + m\ell) \leq O(t^{1/d}m^{1-1/d} + m)(b \log N)^{O(1)} \leq O(t + m)(b \log N)^{O(1)}$. Again, by Markov's inequality, with probability $\Omega(1)$, this bound holds; if not, we declare failure. We can compute $|\rho_i(h)|$ for each $h \in H_{\Delta_i}$ naively in time $O(b)$, which can be absorbed in the $b^{O(1)}$ factor.

Finally, we account for the total cost of all operations done to $\widehat{R}^{(q)}$: initialization in step 0, insertions in step 3(b), and computing $\widehat{R}^{(q)}_{\Delta_i}$ in step 1. Fix q. We continue inserting to $\widehat{R}^{(q)}$ only if $q \leq 2\frac{(bt)^{1/d} \log N}{|\widehat{H}|}$. Thus, w.h.p., at all times, $|\widehat{R}^{(q)}| \leq O(q|\widehat{H}| + \log N) \leq O(t^{1/d})(b \log N)^{O(1)}$. Computing $\widehat{R}^{(q)}_{\Delta_i}$ reduces to simplex range searching again. This time, we use an existing data structure that has larger space but supports faster querying, with preprocessing time and total insertion time $O(|\widehat{R}^{(q)}|^d \log^{O(1)} |\widehat{R}^{(q)}|) = O(t)(b \log N)^{O(1)}$ w.h.p., and query time $O(\log^{O(1)} |\widehat{R}^{(q)}| + |\widehat{R}^{(q)}_{\Delta_i}|)$. (Insertions are supported by standard amortization techniques [11].) Since $|\widehat{H}| \leq CN^{O(1)}$, the total number of different q's is $O(\log N)$, which can be absorbed in the $\log^{O(1)} N$ factor.

To summarize, our algorithm succeeds with probability $\Omega(1)$, and w.h.p. the stated bound holds if the algorithm does not fail. We can re-run the algorithm until success, with $O(\log N)$ number of trials w.h.p. The running time remains $O(m + t)(b \log N)^{O(1)}$. At the end, we can assign all the input points to subcells in $O(bn)$ additional time (this term can actually be reduced to $O(n \log b)$ with point location data structures). □

Theorem 5.3 *We can build data structures in $O(n \log n)$ time achieving the bounds stated in Theorems 3.2, 4.1, and 4.2 w.h.p.(n).*

Proof: Using Theorem 5.3, the algorithm in the proof of Theorem 3.2 can immediately be implemented in $O(n \log^{O(1)} n)$ time w.h.p.(n). To reduce the time bound further, we "bootstrap" as follows.

Build a B_0-partial partition tree with $O(n/B_0)$ size and $O(n/B_0)^{1-1/d}$ query cost, as in the proof of Theorem 4.2, with $B_0 = \log^c n$ for a sufficiently large constant c. Here, we take H to be the union of H_t for all $t \le n/B_0$, of total size $m = O((n/B_0)\log^{O(1)} n)$, and the time bound drops to $O(\sum_{t \le n/B_0}[n + (n/B_0)\log^{O(1)} n]) = O(n \log n)$ for our choice of B_0. Using the preceding partition tree method with $P(B_0) = O(B_0 \log^{O(1)} B_0)$ and $Q(B_0) = O(B_0^{1-1/d})$ to handle the subproblems at the leaf cells, we then get a new partition tree with preprocessing time $O(n/B_0)P(B_0) + O(n \log n) = O(n \log n)$ and query time $O(n/B_0)^{1-1/d}Q(B_0) = O(n^{1-1/d})$.

The query bound is expected but can be made to hold w.h.p.(n): For a fixed query hyperplane, we are bounding a sum Z of independent random variables lying between 0 and $O(B_0)$ with $E[Z] \le O(n^{1-1/d})$; by a Chernoff bound (see Appendix A), w.h.p.(n), we have $Z \le O(n^{1-1/d} + B_0 \log n) = O(n^{1-1/d})$. Observe that every cell in our construction are defined by a constant number of input points, so the number of possible cells, and hence the number of combinatorially different query hyperplanes, is polynomially bounded in n. Thus, the maximum query cost is indeed $O(n^{1-1/d})$ w.h.p.(n).

The same approach also works for Theorems 4.1 and 4.2. □

Note that the above result is Las Vegas, since the random choices made in the preprocessing algorithm only affect the query time, not the correctness of the query algorithm.

6 Shallow Version

In this section, we modify our method to solve the halfspace range reporting problem in an even dimension $d \ge 4$. It suffices to consider upper halfspaces, i.e., we want to report all points above a query hyperplane. We say that a hyperplane h is *k-shallow* if the number of points of P above h is at most k.

It can be checked that Observation 2.2 remains true for k-shallow hyperplanes, i.e., with the same test set H, for every k, if the maximum number of cells crossed by a k-shallow hyperplane in H is ℓ, then the maximum number of cells crossed by an arbitrary k-shallow hyperplane is at most $O(\ell)$.

Matoušek [36] gave a "shallow version" of the cutting lemma. Actually, a weaker version of the lemma suffices to derive a "shallow version" of the partition theorem e.g., as noted in [43] (the original shallow cutting lemma wants to cover all points in the $(\le k)$-level, whereas it suffices to cover a large number of points at low levels). We use this weaker cutting lemma, which has a simpler proof, because it makes an X-sensitive variant of the lemma easier to verify. Our version also has a nice new feature: all our cells are *vertical*, i.e., they contain $(0, \ldots, 0, -\infty)$.

Stating the right analog of "X" requires more care. Define $X_\Delta(H, p)$ to be the expected number of vertices of the lower envelope of a p-sample of H inside Δ. Define $\overline{X}_\Delta(H, p) = \sum_{p' \le p} X_\Delta(H, p)$ where the sum is over powers p' of 2. Note that although $X_\Delta(\cdot, \cdot)$ may not be monotone increasing in p, $\overline{X}_\Delta(\cdot, \cdot)$ is, so it is more convenient to work with the latter quantity.

In order to state how many points ought to be covered, define $\mu_\Delta(H)$ (the "measure" of H) to be the total number of pairs (p, h) with $p \in P \cap \Delta$ and $h \in H$ such that p is above h.

Lemma 6.1 (Shallow Cutting Lemma, weak version) *Let H be a set of m hyperplanes, P be a set of points, and Δ be a simplex in $\mathbb{R}^d$. For any r, we can find an $O(1/r)$-cutting of H into $O(r^{\lfloor d/2 \rfloor})$ disjoint vertical cells inside Δ that cover all but $O(\mu_\Delta(H)r/m)$ points of P.*

In terms of $\overline{X}(\cdot, \cdot)$, the number of cells can be reduced to $O(\overline{X}_\Delta(H, r/m) + r^{\lfloor (d-1)/2 \rfloor})$.

Proof: Draw an (r/m)-sample R of H. Take the canonical triangulation T of the lower envelope of R restricted inside Δ. The expected number of cells is $O(\overline{X}_\Delta(H, r/m) + r^{\lfloor (d-1)/2 \rfloor})$, since there are $O(r^{\lfloor (d-1)/2 \rfloor})$ vertices in the lower envelope inside each $(d-1)$-dimensional boundary facet of Δ. For each cell $\Delta \in T$ with $|H_\Delta| = am/r$ $(a > 1)$, further subdivide Δ as follows: pick a $(1/(2a))$-approximation A of H_Δ with $|A| = O(a^2 \log a)$ (ε-*approximations* [32, 40, 43] can be found by random sampling); return any triangulation of the $(\leq |A|/a)$-level L of A into $|A|^{O(1)} \leq a^{O(1)}$ vertical subcells intersected with Δ. The total number of subcells is $\sum_{\Delta \in T} (r|H_\Delta|/m)^{O(1)}$, which has expected value $O(\overline{X}_\Delta(H, r/m) + r^{\lfloor (d-1)/2 \rfloor})$ by Clarkson and Shor's or Chazelle and Friedman's technique [21, 27].

Each vertex of L has level at most $O(|H_\Delta|/a) \leq O(m/a)$ in H, so each subcell is indeed crossed by at most $O(m/r)$ hyperplanes (since a hyperplane crossing the subcell must lie below one of the d vertices of the subcell). Inside a cell $\Delta \in T$, any uncovered point has level at least $|A|/a$ in A and thus at least $\Omega(|H_\Delta|/a) \geq \Omega(m/r)$ in H. So, the number of uncovered points in T is at most $O(\mu_\Delta(H)r/m)$. On the other hand, the number of uncovered points outside T is at most $\mu_\Delta(R)$, which has expected value $\mu_\Delta(H)r/m$. By Markov's inequality, the bounds on the number of cells and uncovered points hold simultaneously with probability $\Omega(1)$. □

Equipped with the shallow cutting lemma, we can adopt the same approach from Section 3 to solve the halfspace range reporting problem, with d replaced by $\lfloor d/2 \rfloor$ in the exponents. The assumption that d is even is critical, to ensure that $\lfloor (d-1)/2 \rfloor$ is strictly less than $\lfloor d/2 \rfloor$.

Theorem 6.2 (Shallow Partition Refinement Theorem) *Let P be a set of at most n points and H be a set of m k-shallow hyperplanes in $\mathbb{R}^d$ with $d > 2$. Suppose we are given t disjoint vertical cells covering P, such that each cell contains at most $2n/t$ points of P and each hyperplane in H crosses at most ℓ cells. Then for any b, we can find $O(b)$ disjoint vertical subcells inside every cell, such that all but $O(bt)^{1/\lfloor d/2 \rfloor}k)$ points are covered by the subcells, each subcell contains at most $n/(bt)$ points of P, and each hyperplane in H crosses at most the following number of subcells:*

$$O((bt)^{1-1/\lfloor d/2 \rfloor} + b^{1-1/\lfloor (d-1)/2 \rfloor}\ell + b \log t \log m).$$

Proof: We apply the same algorithm as in the proof of Theorem 3.1, except that in step 1 we apply Lemma 6.1 to $\widehat{H}_{\Delta_i}$ with

$$r \;\approx\; \min\left\{(bi)^{1/\lfloor d/2 \rfloor}\frac{|\widehat{H}_{\Delta_i}|}{a^*|\widehat{H}|},\;\; b^{1/\lfloor (d-1)/2 \rfloor}\right\}$$

where $a^* \geq 1$ is the smallest value guaranteeing that $\overline{X}_\Delta(\widehat{H}, r/|\widehat{H}_\Delta|) \leq b$ and thus the number of subcells in the cutting is $O(b)$. (As a technicality, make $r/|\widehat{H}_\Delta|$ a power of 2.)

Note that $\sum_{j=1}^n \overline{X}_{\Delta_j}(\widehat{H}, p) \leq O(\sum_{p' \leq p}(p'|\widehat{H}|)^{\lfloor d/2 \rfloor}) = O((p|\widehat{H}|)^{\lfloor d/2 \rfloor})$, where the sum is over powers p' of 2. Conditioned to a fixed choice of $\Delta_t, \ldots, \Delta_{i+1}$, we then have $E[\overline{X}_{\Delta_i}(\widehat{H}, p)] \leq O((p|\widehat{H}|)^{\lfloor d/2 \rfloor}/i)$, which is $O(b/a^{\lfloor d/2 \rfloor})$ for $p \leq (bi)^{1/\lfloor d/2 \rfloor}/(a|\widehat{H}|)$. By Markov's inequality, the probability that $\overline{X}_{\Delta_i}(\widehat{H}, p)$ exceeds b is at most $O(1/a^{1/\lfloor d/2 \rfloor})$. So $E[a^*] \leq \sum_a O(1/a^{\lfloor d/2 \rfloor}) \cdot a = O(1)$, where the sum is over powers a of 2. Now, $\alpha_i := |\widehat{H}_{\Delta_i}|/(r|\widehat{H}|) \leq O(a^*/(bi)^{1/\lfloor d/2 \rfloor} + b^{1/\lfloor (d-1)/2 \rfloor}|\widehat{H}_{\Delta_i}|/|\widehat{H}|)$, so the previous analysis for $E[\alpha_i]$ and the expected crossing number carries through, with d and $d-1$ replaced by $\lfloor d/2 \rfloor$ and $\lfloor (d-1)/2 \rfloor$.

Next, we bound the number of points not covered by the subcells. Note that $\sum_{j=1}^n \mu_{\Delta_j}(\widehat{H}) \leq O(|\widehat{H}|k)$. Conditioned to a fixed choice of $\Delta_t, \ldots, \Delta_{i+1}$, we have $E[\mu_{\Delta_i}(\widehat{H})] \leq O(|\widehat{H}|k/i)$, so the expected number of uncovered points in Δ_i is $O(|\widehat{H}|k/i \cdot r/|\widehat{H}_{\Delta_i}|) \leq O(b^{1/\lfloor d/2 \rfloor}k/i^{1-1/\lfloor d/2 \rfloor})$. Summing

over $i = 1, \ldots, t$ gives the desired $O((bt)^{1/\lfloor d/2 \rfloor} k)$ bound. By Markov's inequality, the bounds on crossing number and the number of uncovered points simultaneously hold with probability $\Omega(1)$. □

Theorem 6.3 *Given n points in $\mathbb{R}^d$ for an even $d \geq 4$, we can build a partition tree that has query cost $O(n^{1-1/\lfloor d/2 \rfloor} + k \log^{O(1)} n)$ for any k-shallow query hyperplane.*

Proof: Let $k_t := n/(ct^{1/\lfloor d/2 \rfloor} \log n)$ for a constant c. We follow the algorithm in the proof of Theorem 3.2, where Π'_t is now constructed from applying Theorem 6.2 to Π_t and the k_{bt}-shallow hyperplanes in the test set H from Observation 2.2. We remove the points not covered. The total number of points removed is at most $O(\sum_t (bt)^{1/\lfloor d/2 \rfloor} k_{bt}) = O(n/c)$, which can be made at most $n/2$ for a sufficiently large c. We ignore the removed points for the time being.

In the analysis, we redefine $\ell(t)$ to be the maximum number of cells of Π_t crossed by an arbitrary k_t-shallow hyperplane, and $\ell'(t)$ to be the maximum number of cells of Π'_t crossed by a k_{bt}-shallow hyperplane in H. The same recurrence holds, with d and $d-1$ replaced by $\lfloor d/2 \rfloor$ and $\lfloor (d-1)/2 \rfloor$, and we get $\ell(u) \leq O(u^{1-1/\lfloor d/2 \rfloor} + \log^{O(1)} n)$.

More generally, define $\ell(t, n, k)$ to be the maximum number of cells of Π_t crossed by a k-shallow hyperplane h. We now show that

$$\ell(u, n, k) \;\leq\; O(u^{1-1/\lfloor d/2 \rfloor} + (1 + ku/n) \log^{O(1)} n).$$

We may assume $k > k_u$. Let $t_* \leq u$ be such that $k \approx k_{t_*}$. There are $O(\sum_{t \leq t_*} \ell(t))$ cells crossed by h containing at least n/t_* points. For each cell Δ crossed by h which contains less than n/t_* points but whose parent contains at least n/t_* points, the cell Δ can contain at most $O((n/t_*)/(n/u)) = O(u/t_*)$ cells of Π_u. Thus, $\ell(u, n, k) \leq O(b \sum_{t \leq t_*} \ell(t) \cdot u/t_*) = O((t_*^{1-1/\lfloor d/2 \rfloor} + \log^{O(1)} n) \cdot (u/t_*)) = O((uk/n) \log^{O(1)} n)$.

The at most $n/2$ removed points can be handled recursively. We can join the $O(\log n)$ partition trees into one by creating a new root of degree $O(\log n)$. Define $\overline{\ell}(u, n, k)$ to be the maximum overall number of cells containing between n/u and $2n/u$ points that are crossed by a k-shallow hyperplane. Then $\overline{\ell}(u, n, k) \;\leq\; \ell(u, n, k) + \ell(u/2, n/2, k) + \cdots \;\leq\; O(u^{1-1/\lfloor d/2 \rfloor} + (1 + ku/n) \log^{O(1)} n)$.

We conclude that the query cost is at most $O\left(\sum_{t \leq n} b\overline{\ell}(t, n, k)\right) \;\leq\; O(n^{1-1/\lfloor d/2 \rfloor} + k \log^{O(1)} n)$. □

Theorem 6.4 *Given n points in $\mathbb{R}^d$ for an even $d \geq 4$ and $B < n/\log^{\omega(1)} n$, we can build a B-partial partition tree with size $O(n/B)$ and query cost $O((n/B)^{1-1/\lfloor d/2 \rfloor} + (k/B) \log^{O(1)} n)$ for any k-shallow query hyperplane.*

Proof: The proof is as in that of Theorem 4.2, where the query cost is now $O\left(\sum_{t \leq n/B} \overline{\ell}(t, n, k)\right) \leq O((n/B)^{1-1/\lfloor d/2 \rfloor} + (1 + k/B) \log^{O(1)} n)$. □

Remarks. Because of the recursive handling of removed points, we loose the property that the children cells are disjoint, but only at the root.

Our method above is more direct than some of halfspace range reporting methods from Matoušek's paper [36], in that we do not need auxiliary data structures. We bound the overall crossing number of a k-shallow hyperplane directly as a function of n and k.

We have purposely been sloppy about the second $O(k \log^{O(1)} n)$ term in Theorem 6.3, since our main interest is in the case when k is small. (For large $k > n^{1-1/\lfloor d/2 \rfloor} \log^{\omega(1)} n$, we already know how

to get $O(k)$ time [36].) By bootstrapping, it seems possible to reduce to polylogarithmic factor to the j-th iterated logarithm $\log^{(j)} n$ for any constant j.

It is not difficult to construct point sets where the maximum query cost of an arbitrary $n^{1-1/\lfloor d/2 \rfloor}$-shallow hyperplane is $\Omega(n^{1-1/\lfloor d/2 \rfloor})$ for any partition tree.

Preprocessing time. We now examine the preprocessing time of this method. First we need a more economical shallow version of the test set observation. This time, we cannot quote Matoušek's paper [36] directly, since he did not state a sufficiently general version of the observation (he only considered (n/t)-shallow hyperplanes rather than near-$(n/t^{1/\lfloor d/2 \rfloor})$-shallow hyperplanes, so the test set size became larger). We thus include a proof of the observation we need (as it turns out, the dependence on t disappears in the case of vertical cells):

Observation 6.5 (Shallow Test Set) *Given a set P of n points in $\mathbb{R}^d$ and any $k \geq \log N$, we can construct a set H_k of $O((n/k)^{\lfloor d/2 \rfloor} \log^{O(1)} N)$ $O(k)$-shallow hyperplanes, in $O((n/k)^{\lfloor d/2 \rfloor} \log^{O(1)} N)$ time, satisfying the following property w.h.p.(N) for any $N \geq n$: For any collection of vertical cells, if the maximum number of cells crossed by a hyperplane in H is ℓ, then the maximum number of cells crossed by an arbitrary k-shallow hyperplane is $O(\ell)$.*

Proof: In what follows, the dual of an object q is denoted by q^*. Draw a random sample R of P of size $(n/k)\log N$ and return the set H_k of all hyperplanes through d-tuples of points in R that are $(c \log N)$-shallow with respect to R for a constant c. Note that in the dual, H_k^* corresponds to the vertices of the $(\leq c \log N)$-level L of R^* and thus has size $O(|\widehat{R}|^{\lfloor d/2 \rfloor} \log^{O(1)} N) = O((n/k)^{\lfloor d/2 \rfloor} \log^{O(1)} N)$ [27].

To prove correctness, first note that every hyperplane in H_k is $O(k)$-shallow with respect to P w.h.p.(N) by a Chernoff bound (the number of combinatorially different hyperplanes is polynomially bounded). Let h be a k-shallow hyperplane with respect to P. Then by a Chernoff bound, w.h.p.(N), it is $(c \log N)$-shallow with respect to R for a sufficiently large c, i.e., the point h^* is in L. Thus, h^* lies below a facet Δ^* of L, defined by d vertices $h_1^*, \ldots, h_d^* \in H_k^*$. Suppose h crosses a vertical cell σ. Then some point $q \in \sigma$ lies above h, i.e., the hyperplane q^* lies below h^*. Then q^* must lie below one of the points $h_1^*, \ldots, h_d^*$, i.e., q must lie above one of the hyperplanes $h_1, \ldots, h_d \in H_k$. So, σ is crossed by one of $h_1, \ldots, h_d$. □

In the proof of Theorem 6.3, we can take H to be the union of the the test sets H_{k_t} over all t; the total size is $m = O(\sum_t (n/k_t)^{\lfloor d/2 \rfloor} \log^{O(1)} N) = O(\sum_t t \log^{O(1)} N) = O(n \log^{O(1)} N)$.

Lemma 6.1 requires $O(mr^{O(1)})$ time; we can repeat for $O(\log N)$ trials and return best result found to ensure correctness w.h.p.(N).

We can then proceed as in the proof of Theorem 5.2 to get a near-linear-time algorithm for Theorem 6.2. Since cells are vertical, the computation of H_{Δ_i} and R_{Δ_i} now reduces to halfspace range reporting in the dual (reporting hyperplanes crossing a vertical cell reduces to reporting hyperplanes lying below each of the d vertices of the cell).

It can be checked that the bootstrapping step in the proof of Theorem 5.3 carries through as well, yielding the final result:

Theorem 6.6 *We can build a partition tree in $O(n \log n)$ time achieving the bound stated in Theorems 6.3 and 6.4 w.h.p.(n).*

7 Some Applications

Although we dare not go through all papers in the literature that have used simplex or halfspace range searching data structures as subroutines, it is illustrative to at least briefly mention a few sample applications.

Spanning trees with low crossing number. A series of papers [2, 9, 29, 34, 48] addressed the construction of spanning trees such that the maximum number of edges crossed by a hyperplane is small. This problem has direct applications to a number of geometric problems [3, 48]. We obtain the first $O(n \log n)$-time algorithm that attains asymptotically optimal worst-case crossing number. The result follows directly from Theorems 3.2 and 5.3: inside each cell in the partition tree, we just create a constant ($O(b)$) number of edges to connect the subsets at the children.

Corollary 7.1 *Given n points in $\mathbb{R}^d$, there is an $O(n \log n)$-time Monte Carlo algorithm to compute a spanning tree with the property that any hyperplane crosses at most $O(n^{1-1/d})$ tree edges w.h.p.(n). For $d = 2$, we can ensure that the output tree is planar.*

Multilevel data structures. Many query problems are solved using multilevel partition trees, where we build secondary data structures for each *canonical subset* and answer a query by identify relevant canonical subsets and querying their corresponding secondary structures. The corollary below encapsulates the main property of our partition tree for multilevel applications, and is a direct consequence of Theorems 4.1 and 5.3:

Corollary 7.2 *Given n points in $\mathbb{R}^d$,*

(i) *we can form $O(n)$ canonical subsets of total size $O(n \log n)$ in $O(n \log n)$ time, such that the subset of all points inside any query simplex can be reported as a union of disjoint canonical subsets C_i with $\sum_i |C_i|^{1-1/d} \leq O(n^{1-1/d} \log n)$ in time $O(n^{1-1/d} \log n)$ w.h.p.(n);*

(ii) *for any fixed $\gamma < 1 - 1/d$, we can form $O(n)$ canonical subsets of total size $O(n \log \log n)$ in $O(n \log n)$ time, such that the subset of all points inside any query simplex can be reported as a union of disjoint canonical subsets C_i with $\sum_i |C_i|^{\gamma} \leq O(n^{1-1/d})$ in time $O(n^{1-1/d})$ w.h.p.(n).*

For example, we can get the following results:

Corollary 7.3 *Given n line segments in the plane, there is a data structure with*

(a) *$O(n \log^2 n)$ preprocessing time and $O(n \log n)$ space such that we can report the k intersections with a query line in $O(\sqrt{n} \log n + k)$ expected time;*[4]

(b) *$O(n \log^3 n)$ preprocessing time and $O(n \log^2 n)$ space such that we can report the k intersections with a query line segment in $O(\sqrt{n} \log^2 n + k)$ expected time;*

(c) *$O(n \log^3 n)$ preprocessing time and $O(n \log^2 n)$ space such that we can find the first intersection by a query ray in $O(\sqrt{n} \log^2 n)$ expected time.*

[4]For simplicity, we state expected time bounds rather than high-probability bounds in this section, although some can be made high-probability bounds with more care. In the expected bounds, we assume that the query objects are independent of the random choices made by the preprocessing algorithm.

Specifically, (a) follows by applying Corollary 7.2(i) twice; (b) follows by applying Corollary 7.2(i) once more; and (c) follows from (b) by a simple randomized reduction (e.g., [14]). For (c), the previous result by Agarwal [3] had $O(n\alpha(n)\log^4 n)$ space and $O(\sqrt{n\alpha(n)}\log^2 n)$ query time (for nonintersecting segments, his bounds of $O(n\log^3 n)$ space and $O(\sqrt{n}\log^2 n)$ query time are still worse than ours).

For another example, we get the following result by applying Corollary 7.2(i) $d+1$ times:

Corollary 7.4 *Given n simplices in $\mathbb{R}^d$, there is a data structure with $O(n\log^{d+1} n)$ preprocessing time and $O(n\log^d n)$ space, such that we can find all simplices containing a query point in $O(n^{1-1/d}\log^d n)$ expected time; and we can find all simplices contained inside a query simplex in $O(n^{1-1/d}\log^d n)$ expected time.*

In contrast, previous methods [35, 38] had $n^{O(1)}$ preprocessing time, $O(n\log^{2d} n)$ space, and $O(n^{1-1/d}\log^d n)$ query time; or $O(n^{1+\varepsilon})$ preprocessing time, $O(n\log^{O(1)} n)$ space, and $O(n^{1-1/d}\log^{O(1)} n)$ query time; or $O(n2^{O(\sqrt{\log n})})$ preprocessing time and space, and $O(n^{1-1/d}2^{O(\sqrt{\log n})})$ query time.

Other problems can be solved in a similar way, e.g., intersection searching among simplices in $\mathbb{R}^d$.

Shallow applications. Halfspace range emptiness dualizes to *membership* queries for an intersection $\mathcal{P}$ of n halfspaces: decide whether a query point lies in $\mathcal{P}$. The problem is a special case of *ray shooting* in a convex polytope: find the intersection of a query ray with $\mathcal{P}$. In turn, ray shooting queries are special cases of j-dimensional *linear programming queries*: find a point in $\mathcal{P}$ that is extreme along a query direction and lies inside a query j-flat. The author [14] (see also [46]) has given a randomized reduction of linear programming queries to halfspace range reporting queries in the case when the output size k is small ($O(\log n)$). This reduction does not increase the asymptotic time and space bounds w.h.p. if the query time bound grows faster than n^{ε} for some fixed $\varepsilon > 0$. *Exact nearest neighbor search* (finding the closest point in a given point set to a query point under the Euclidean metric) reduces to ray shooting in one higher dimension by the standard lifting map. Theorems 6.3 and 6.6 thus imply:

Corollary 7.5 *For $d \geq 4$ even, there are data structures for halfspace range emptiness queries for n points in $\mathbb{R}^d$, for ray shooting and linear programming queries inside the intersection of n halfspaces in $\mathbb{R}^d$, and exact nearest neighbor queries of n points in $\mathbb{R}^{d-1}$, with $O(n\log n)$ preprocessing time, $O(n)$ space, and $O(n^{1-1/\lfloor d/2\rfloor})$ expected query time.*

Trade-offs. Thus far, we have focused exclusively on linear-space data structures. By combining with a large-space data structure with logarithmic query time, one can usually obtain a continuous trade-off between space and query time. Specifically, such a trade-off for halfspace range counting, for example, can be obtained through the following corollary:

Corollary 7.6 *Suppose there is a d-dimensional halfspace range counting data structure for point sets of size at most B with $P(B)$ preprocessing time, $S(B)$ space, and $Q(B)$ (expected) query time. Then there is a d-dimensional halfspace range counting data structure for point sets of size at most n with*

(i) $O(n/B)P(B) + O(n \log n)$ *preprocessing time,* $O(n/B)S(B) + O(n)$ *space, and* $O(n/B)^{1-1/d}Q(B) + O(n/B)^{1-1/d}$ *expected query time, assuming* $B < n/\log^{\omega(1)} n$.

(ii) $O(n/B)^d P(B) + (n/B)^d B)$ *preprocessing time,* $O(n/B)^d S(B) + O((n/B)^d B)$ *space, and* $Q(B) + O(\log(n/B))$ *(expected) query time.*

Proof: Part (i) is an immediate consequence of Theorems 4.2 and 5.3. Part (ii) is known and follows from a direct application of hierarchical cuttings [19] (in particular, see [38, Theorem 5.1]). □

As space/query-time tradeoffs are already known [38], we look instead at preprocessing-time/query-time trade-offs, which are more important in algorithmic applications and where we get new results. We explain how such a trade-off can be obtained through the above corollary. Interesting, iterated logarithm arises:

Corollary 7.7 *There is a d-dimensional halfspace range counting data structure with* $O(m2^{O(\log^* n)})$ *preprocessing time and* $O((n/m^{1/d})2^{O(\log^* n)})$ *expected query time for any given* $n \log n \le m \le n^d/\log^d n$.

Proof: We first consider the extreme cases when m is $\Theta(n \log n)$ or $\Theta(n^d/\log^d n)$. Assume inductively that there is a data structure for problem size n_i with preprocessing and query time

$$P(n_i) \le c_i n_i \log n_i, \qquad Q(n_i) \le c_i n_i^{1-1/d}/\log^{1/d} n_i.$$

Set n_{i+1} so that $\log n_{i+1} = n_i^{1-1/d}/\log^{1/d} n_i$. Apply Corollary 7.6(ii) to get a data structure for problem size n_{i+1}, with preprocessing and query time

$$\begin{aligned} P(n_{i+1}) &\le O((n_{i+1}/n_i)^d c_i n_i \log n_i) = O(c_i n_{i+1}^d/\log^d n_{i+1}), \\ Q(n_{i+1}) &\le c_i n_i^{1-1/d}/\log^{1/d} n_i + O(\log n_{i+1}) = O(\log n_{i+1}). \end{aligned}$$

Next, set n_{i+2} so that $\log n_{i+2} = n_{i+1}^{d-1}/\log^d n_{i+1}$. Apply Corollary 7.6(i) to get a data structure for problem size n_{i+2}, with preprocessing and query time

$$\begin{aligned} P(n_{i+2}) &\le O((n_{i+2}/n_{i+1}) c_i n_{i+1}^d/\log^d n_{i+1} + n_{i+2} \log n_{i+2}) = O(c_i n_{i+2} \log n_{i+2}), \\ Q(n_{i+2}) &\le O((n_{i+2}/n_{i+1})^{1-1/d} c_i \log n_{i+1}) = O(c_i n_{i+2}^{1-1/d}/\log^{1/d} n_{i+2}). \end{aligned}$$

Thus, the induction carries through with $c_{i+2} \le O(1)c_i$, implying $c_i \le 2^{O(i)}$. Since $n_i \le \log^{O(1)} n_{i+1}$, the number of iterations to reach problem size n is $O(\log^* n)$, so we conclude that there is a data structure with preprocessing time $O(2^{O(\log^* n)} n \log n)$ and query time $O(2^{O(\log^* n)} n^{1-1/d}/\log^{1/d} n)$.

To obtain the full trade-off, we use one final application of Corollary 7.6(ii), choosing B so that $B^{d-1}/\log B = n^d/m$, to get preprocessing time $O(2^{O(\log^* n)}(n/B)^d B \log B) = O(2^{O(\log^* n)} m)$ and query time $O(2^{O(\log^* n)} B^{1-1/d}/\log^{1/d} B + \log n) = O(2^{O(\log^* n)} n/m^{1/d})$. □

The above result is reminiscent of Matoušek's $O(n^{4/3}2^{O(\log^* n)})$-time algorithm [38] for *Hopcroft's problem* in 2-d: given n points and n lines in the plane, find a point-line incidence (or similarly count the number of point-above-line pairs). Matoušek's solution required only cuttings and managed

to avoid partition trees because the problem is off-line, i.e., the queries are known in advance. Corollary 7.7 can be viewed as an extension to on-line queries.

(Incidentally, we pick the example of halfspace range counting in the above, because for simplex range counting, the current best large-space data structure needed in Corollary 7.7(ii) has extra logarithmic factors [38], so our new method can eliminate some but not all logarithmic factors.)

We can obtain similar results for halfspace range emptiness:

Corollary 7.8 *Let $d \geq 4$ be even. Suppose there is a d-dimensional halfspace range emptiness data structure for point sets of size at most B with $P(B)$ (expected) preprocessing time, $S(B)$ (expected) space, and $Q(B)$ (expected) query time. Then there is a d-dimensional halfspace range emptiness data structure for point sets of at most size n with*

(i) *$O(n/B)P(B)+O(n\log n)$ (expected) preprocessing time, $O(n/B)S(B)+O(n)$ (expected) space, and $O(n/B)^{1-1/\lfloor d/2\rfloor}Q(B)+O(n/B)^{1-1/\lfloor d/2\rfloor}$ expected query time, assuming $B < n/\log^{\omega(1)} n$.*

(ii) *$O(n/B)^d P(B) + O((n/B)^d B)$ expected preprocessing time, $O(n/B)^d S(B) + O((n/B)^d B)$ expected space, and $Q(B) + O(\log(n/B))$ expected query time.*

Proof: Part (i) is a consequence of Theorems 6.4 and 6.6. However, one technical issue arises: for emptiness queries, we could purport that the range is nonempty if the query cost exceeds the stated limit, but this would only yield a Monte Carlo algorithm. Here is a Las Vegas alternative: Store a random sample R of size r in any halfspace range emptiness data structure with $O(r\log r)$ preprocessing time, $O(r)$ space, and $O(r^{1-\delta})$ query time for some fixed $\delta > 0$. If the query halfspace h is not empty with respect to R, we are done. Otherwise, h is $O((n/r)\log n)$-shallow w.h.p.(n) by a Chernoff bound (or the *ε-net* property of random samples [32, 40, 43]). Then w.h.p.(n), the query cost from Theorem 6.4 is $O((n/B)^{1-1/\lfloor d/2\rfloor} + (n/(rB))\log^{O(1)} n)$. Setting $r = (n/B)^{1/\lfloor d/2\rfloor}\log^c n$ for a sufficiently large constant c yields the result. (Note that for $d \geq 6$, the auxiliary data structure for R is unnecessary as $\delta = 0$ still works.)

Part (ii) follows from a direct application of hierarchical cuttings in the shallow context where we only need to cover a lower envelope of n halfspaces (although no references explicitly state this result, see [10, 45] for the general idea, which involves just repeated applications of an X-sensitive shallow cutting lemma as in Lemma 6.1). □

We can proceed as in the proof of Corollary 7.7 to get a preprocessing/query-time trade-off for halfspace range emptiness. The author [16] has given another randomized reduction from linear programming queries to halfspace range emptiness. This reduction does not increase the asymptotic expected query time if it grows faster than n^{ε}, and does not increase the asymptotic preprocessing time if it grows faster than $n^{1+\varepsilon}$. Therefore:

Corollary 7.9 *For $d \geq 4$ even, there is a d-dimensional halfspace range emptiness data structure with $O(m2^{O(\log^* n)})$ preprocessing time and $O((n/m^{1/\lfloor d/2\rfloor})2^{O(\log^* n)})$ expected query time for any given $n\log n \leq m \leq n^{\lfloor d/2\rfloor}/\log^{\lfloor d/2\rfloor} n$. The same result holds for ray shooting and linear programming queries inside the intersection of n halfspaces in $\mathbb{R}^d$, and exact nearest neighbor queries of n points in $\mathbb{R}^{d-1}$, assuming $n^{1+\delta} \leq m \leq n^{\lfloor d/2\rfloor-\delta}$ for some fixed $\delta > 0$.*

One algorithmic application. Improved data structures often lead to improved algorithms for off-line problems. We close with just one such example: Given n points, we can find the extreme points (the vertices of the convex hull) by answering n linear programming queries. By choosing $m = n^{2-2/(\lfloor d/2 \rfloor+1)}$, we get the following corollary, improving the best previous worst-case time bound of $O(n^{2-2/(\lfloor d/2 \rfloor+1)} \log^{O(1)} n)$ [37, 15].

Corollary 7.10 *For $d \geq 4$ even, we can compute the extreme points of a given set of n points in $\mathbb{R}^d$ in $O(n^{2-2/(\lfloor d/2 \rfloor+1)} 2^{O(\log^* n)})$ expected time.*

By Seidel's algorithm [47], we can construct the convex hull itself in additional $O(f \log n)$ time where f denotes the output size. (For $f < n$, we could also slightly speed up the convex hull algorithm in [15] for $d \geq 6$ even, but an algorithm in [17, Theorem 6.3] is faster in this case.)

8 Conclusions

The new results of this paper completely subsume most of the results from Matoušek's SoCG'92 paper [38], at least if randomized algorithms are acceptable (which are to most researchers nowadays). Although we have resolved some of the main open questions concerning upper bounds, there are still a few minor issues. For example, is it possible to eliminate the iterated logarithmic factor in the preprocessing-time/query-time tradeoffs (see Corollary 7.7)? For halfspace range reporting for even d, can one get $O(n)$ space and $O(n^{1-1/\lfloor d/2 \rfloor} + k)$ query time, without any extra iterated logarithmic factors in either term (see the remarks after Theorem 6.4)? More importantly, for halfspace emptiness for odd d, can one get $O(n)$ space and $O(n^{1-1/\lfloor d/2 \rfloor})$ query time?

In the other direction, for simplex range searching, can one prove a tight $\Omega(n^{1-1/d})$ lower bound on the query time for linear-space data structures in, say, the semigroup model [18], without any extra logarithmic factor? Can one prove lower bounds showing in some sense that logarithmic-factor increase is necessary for multilevel partition trees (see Corollary 7.2), or that extra logarithmic factors are necessary in the query time in the case of large near-$O(n^{\lfloor d/2 \rfloor})$ space [38]? More importantly, can one prove near optimal lower bounds for halfspace range emptiness?

It seems plausible that our approach could be adapted to other kinds of partition trees for semialgebraic sets [8, 6] in some cases (where the complexity of arrangements or lower envelopes in $d-1$ dimensions is strictly less than in d dimensions). We do not know yet if our approach could lead to new results for dynamic partition trees that support insertions and deletions.

References

[1] P. Afshani and T. M. Chan. Optimal halfspace range reporting in three dimensions. In *Proc. 20th ACM-SIAM Sympos. Discrete Algorithms*, pages 180–186, 2009.

[2] P. K. Agarwal. *Intersection and Decomposition Algorithms for Planar Arrangements*. Cambridge University Press, 1991.

[3] P. K. Agarwal. Ray shooting and other applications of spanning trees with low stabbing number. *SIAM J. Comput.*, 21:540–570, 1992.

[4] P. K. Agarwal. Range searching. In *CRC Handbook of Discrete and Computational Geometry* (J. Goodman and J. O'Rourke, eds.), CRC Press, New York, 2004.

[5] P. K. Agarwal, L. Arge, J. Erickson, P. G. Franciosa, and J. S. Vitter. Efficient searching with linear constraints. *J. Comput. Sys. Sci.*, 61:194–216, 2000.

[6] P. K. Agarwal, A. Efrat, and M. Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. In *SIAM J. Comput.*, 29:912–953, 1999.

[7] P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. In *Discrete and Computational Geometry: Ten Years Later* (B. Chazelle, J. E. Goodman, and R. Pollack, eds.), AMS Press, pages 1–56, 1999.

[8] P. K. Agarwal and J. Matoušek. On range searching with semialgebraic sets. *Discrete Comput. Geom.*, 11:393–418, 1994.

[9] P. K. Agarwal and M. Sharir. Applications of a new space-partitioning technique. *Discrete Comput. Geom.*, 9:11–38, 1993.

[10] N. M. Amato, M. T. Goodrich, and E. A. Ramos. Parallel algorithms for higher-dimensional convex hulls. In *Proc. 35th IEEE Sympos. Found. Comput. Sci.*, pages 683–694, 1994.

[11] J. Bentley and J. Saxe. Decomposable searching problems I: static-to-dynamic transformation. *J. Algorithms*, 1:301–358, 1980.

[12] M. de Berg and O. Schwarzkopf. Cuttings and applications. *Int. J. Comput. Geom. Appl.*, 5:343–355, 1995.

[13] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete Comput. Geom.*, 14:463–479, 1995.

[14] T. M. Chan. Fixed-dimensional linear programming queries made easy. In *Proc. 12th ACM Sympos. Comput. Geom.*, pages 284–290, 1996.

[15] T. M. Chan. Output-sensitive results on convex hulls, extreme points, and related problems. *Discrete Comput. Geom.*, 16:369–387, 1996.

[16] T. M. Chan. An optimal randomized algorithm for maximum Tukey depth. In *Proc. 15th ACM-SIAM Sympos. Discrete Algorithms*, pages 423–429, 2004.

[17] T. M. Chan, J. Snoeyink, and C.-K. Yap. Primal dividing and dual pruning: output-sensitive construction of four-dimensional polytopes and three-dimensional Voronoi diagrams. *Discrete Comput. Geom.*, 18:433–454, 1997.

[18] B. Chazelle. Lower bounds on the complexity of polytope range searching. *J. AMS*, 2:637–666, 1989.

[19] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9:145–158, 1993.

[20] B. Chazelle. Cuttings. In *Handbook of Data Structures and Applications* (D. P. Mehta and S. Sahni, eds.), CRC Press, pages 25.1–25.10, 2005.

[21] B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10:229–249, 1990.

[22] B. Chazelle and B. Rosenberg. Simplex range reporting on a pointer machine. *Comput. Geom. Theory Appl.*, 5:237–247, 1996.

[23] B. Chazelle, M. Sharir, and E. Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Algorithmica*, 8:407–429, 1992.

[24] B. Chazelle and E. Welzl. Quasi-optimal range searching in space of finite VC-dimension. *Discrete Comput. Geom.*, 4:467–489, 1989.

[25] K. L. Clarkson. A randomized algorithm for closest-point queries. *SIAM J. Comput.*, 17:830–847, 1988.

[26] K. L. Clarkson. Las Vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, 42:488–499, 1995.

[27] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[28] D. Dobkin and H. Edelsbrunner. Organizing point sets in two and three dimensions. Report F130, Inst. Informationsverarb., Tech. Univ. Graz, Austria, 1984.

[29] H. Edelsbrunner, L. Guibas, J. Hershberger, R. Seidel, M. Sharir, J. Snoeyink, and E. Welzl. Implicitly representing arrangements of lines or segments. *Discrete Comput. Geom.*, 4:433–466, 1989.

[30] H. Edelsbrunner and F. Huber. Dissecting sets of points in two and three dimensions. Report F138, Inst. Informationsverarb., Tech. Univ. Graz, Austria, 1984.

[31] H. Edelsbrunner and E. Welzl. Halfplanar range search in linear space and $O(n^{0.695})$ query time. *Inform. Process. Lett.*, 23:289–293, 1986.

[32] D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete Comput. Geom.*, 2:127–151, 1987.

[33] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. American Statistical Assoc.*, 58:13–30, 1963.

[34] J. Matoušek. Spanning trees with low crossing number. *Informatique Théorique et Applications*, 25:103–124, 1991.

[35] J. Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8:315–334, 1992.

[36] J. Matoušek. Reporting points in halfspaces. *Comput. Geom. Theory Appl.*, 2:169–186, 1992.

[37] J. Matoušek. Linear optimization queries. *J. Algorithms*, 14:432–448, 1993. Also with O. Schwarzkopf in *Proc. 8th ACM Sympos. Comput. Geom.*, pages 16–25, 1992.

[38] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete Comput. Geom.*, 10:157–182, 1993.

[39] J. Matoušek. Geometric range searching. *ACM Comput. Surv.*, 26:421–461, 1994.

[40] J. Matoušek. *Lectures on Discrete Geometry*. Springer-Verlag, 2002.

[41] J. Matoušek and O. Schwarzkopf. On ray shooting in convex polytopes. *Discrete Comput. Geom.*, 10:215–232, 1993.

[42] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[43] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms*. Prentice-Hall, Englewood Cliffs, N.J., 1994.

[44] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, 1985.

[45] E. Ramos. On range reporting, ray shooting, and k-level construction. In *Proc. 15th ACM Sympos. Comput. Geom.*, pages 390–399, 1999.

[46] E. Ramos. Linear programming queries revisited. In *Proc. 16th ACM Sympos. Comput. Geom.*, pages 176–181, 2000.

[47] R. Seidel. Constructing higher-dimensional convex hulls at logarithmic cost per face. In *Proc. 18th ACM Sympos. Theory Comput.*, pages 404–413, 1986.

[48] E. Welzl. On spanning trees with low crossing numbers. In *Data Structures and Efficient Algorithms* (B. Monien and T. Ottmann, eds.), Lect. Notes Comput. Sci., vol. 594, Springer-Verlag, pages 233–249, 1992.

[49] D. E. Willard. Polygon retrieval. *SIAM J. Comput.*, 11:149–165, 1982.

[50] A. C. Yao and F. F. Yao. A general approach to D-dimensional geometric queries. In *Proc. 17th ACM Sympos. Theory Comput.*, pages 163–168, 1985.

[51] F. F. Yao. A 3-space partition and its applications. In *Proc. 15th ACM Sympos. Theory Comput.*, pages 258–263, 1983.

[52] F. F. Yao, D. P. Dobkin, H. Edelsbrunner, and M. S. Paterson. Partitioning space for range queries. *SIAM J. Comput.*, 18:371–384, 1989.

A On Chernoff Bounds

We briefly mention the form of Chernoff bounds that we use. In our applications, we want high-probability bounds but do not need sharp concentrations (constant factor approximations suffice), so simplified versions of the bounds will do. One of the more commonly seen versions of the Chernoff bound (e.g., see [42, pages 68–70]) appears as follows: If X is the sum of independent 0-1 variables, e.g., it is binomially distributed, with $E[X] = \mu$, then

$$\begin{aligned} \Pr[X > (1+\delta)\mu] &< \left[\frac{e^\delta}{(1+\delta)^{1+\delta}}\right]^\mu \quad \text{for any } \delta > 0 \\ \Pr[X < (1-\delta)\mu] &< e^{-\mu\delta^2/2} \quad \text{for any } 0 \le \delta \le 1. \end{aligned}$$

The first probability is at most $2^{-(1+\delta)\mu}$ when $\delta > 2e-1$ (e.g., see [42, page 72]). Thus, $\Pr[X > 2e\mu + c_0 \log N] < 1/N^{c_0}$, and we can write $X \le O(\mu + \log N)$ w.h.p.(N).

The second probability for $\delta = 1/2$ is at most $e^{-\mu/8} < 1/N^{c_0}$ when $\mu > 8c_0 \log N$. Thus, $\Pr[X < \mu/2 - 4c_0 \log N] < 1/N^{c_0}$, and we can write $\mu \le O(X + \log N)$ w.h.p.(N).

The first bound holds more generally when X is the sum of independent variables X_i where each X_i is bounded between 0 and 1. (If the variables are upper-bounded by some other value, we can re-scale before applying.) The first bound also holds when X is hypergeometrically distributed, i.e., it is the number of elements of R in $\{1, \ldots, \ell\}$ where R is a random subset of size i from $\{1, \ldots, t\}$ chosen by sampling without replacement for a given i, ℓ, t (e.g., see [33]).