Higher-dimensional Orthogonal Range Reporting and Rectangle Stabbing in the Pointer Machine Model

Peyman Afshani* Lars Arge Kasper Green Larsen†

MADALGO‡
Department of Computer Science
Aarhus University, Denmark.

afshani@cs.dalhouse.ca,{large,larsen}@madalgo.au.dk

Abstract

In this paper, we consider two fundamental problems in the pointer machine model of computation, namely orthogonal range reporting and rectangle stabbing. Orthogonal range reporting is the problem of storing a set of n points in d-dimensional space in a data structure, such that the t points in an axis-aligned query rectangle can be reported efficiently. Rectangle stabbing is the "dual" problem where a set of n axis-aligned rectangles should be stored in a data structure, such that the t rectangles that contain a query point can be reported efficiently. Very recently an optimal $O(\log n + t)$ query time pointer machine data structure was developed for the three-dimensional version of the orthogonal range reporting problem. However, in four dimensions the best known query bound of $O(\log^2 n/\log\log n + t)$ has not been improved for decades.

We describe an orthogonal range reporting data structure that is the first structure to achieve significantly less than $O(\log^2 n + t)$ query time in four dimensions. More precisely, we develop a structure that achieves query time of $O(\log n(\log n/\log\log n)^{d-4+1/(d-2)} + t)$ using $O(n(\log n/\log\log n)^d)$ space in dimensions $d \geq 4$. Ignoring $\log\log n$ factors, this speeds up the best previous query time by a $\log^{1-1/(d-2)} n$ factor. For the rectangle stabbing problem, we show that any data structure that uses nh space must use $\Omega(\log n(\log n/\log h)^{d-2} + t)$ time to answer a query. This improves the previous results by a $\log h$ factor, and is the first lower bound that is optimal for a large range of h, namely for $h \geq \log^{d-2+\varepsilon} n$ where $\varepsilon > 0$ is an arbitrarily small constant. By a simple geometric transformation, our result also implies an improved query lower bound for orthogonal range reporting.

*Work was supported by Natural Sciences and Engineering Research Council of Canada (NSERC PDF)
†Is supported by a Google Europe Fellowship in Search and Information Retrieval
‡Center for Massive Data Algorithmics—a center of the Danish National Research Foundation.

1 Introduction

In this paper we study two fundamental range searching problems, rectangle stabbing and orthogonal range reporting. In orthogonal range reporting, the goal is to store an input set, S, of n points in d-dimensional space in a data structure such that the t points contained in[1] an axis-aligned query rectangle can be reported efficiently. Rectangle stabbing is the "dual" problem in which the goal is to store an input set, Q, of n rectangles in d-dimensional space such that the t rectangles that contain a query point can be reported efficiently.

Orthogonal range reporting is a central problem in several fields, including spatial databases and computational geometry, and it has been studied extensively [4, 5]. Rectangle stabbing (a.k.a point enclosure) is also one of the classical problems in computational geometry [12]. We study both problems in the pointer machine model of computation [17]. In 2-D, orthogonal range reporting was completely characterized more than two decades ago. Very recently, a complete characterization of the three-dimensional case was also achieved. In higher dimensions, however, the status of the problem is much more mysterious and it has been the topic of some speculations. Unlike orthogonal range reporting, only the two-dimensional case was characterized for rectangle stabbing.

In this paper we present two main results. For orthogonal range reporting, we improve on the best known query time in dimensions four and above. Curiously, the improvement grows with dimension. For rectangle stabbing, we obtain the first tight query time lower bound. By known techniques, this gives an improved query time lower bound for orthogonal range reporting as well.

1.1 Previous Results

In this section, we review the previous results on orthogonal range reporting and rectangle stabbing in the pointer machine. We only review the results most related to ours, that is, results for static structures that answer queries in poly-logarithmic time and with near-linear space usage. We refer the reader to the surveys [4, 5] for further results.

Before reviewing the previous results, we need to introduce two important special cases of the orthogonal range reporting problem. In two dimensions, a query is called a *three-sided* query if at most three of the query boundaries are finite (e.g., $(a; b) \times (-\infty; c)$[2]). A query in d dimensions is called a *dominance* query if there is at most one finite boundary at each dimension.

For 2-dimensional orthogonal range reporting, the complexity of the problem was completely settled more than two decades years ago. In [8], Chazelle provided a structure that can answer queries in $O(\log n + t)$ time while using $O(n \log n / \log\log n)$ space, where t denotes the output size. This is optimal, since Chazelle later proved that any structure for d-dimensional orthogonal range reporting that answers queries in $O(\log^{O(1)} n + t)$ time must use $\Omega(n(\log n/ \log\log n)^{d-1})$ space [9]. For three-sided queries, McCreight [16] presented the priority search tree, that can answer queries in optimal $O(\log n + t)$ time and uses linear space.

Until recently the complexity of the problem remained unresolved in 3-D. For 3-D dominance queries, Afshani [1] was the first to present an optimal structure that uses linear space and answers queries in $O(\log n + t)$ time. Using a standard reduction, this provides a 3-dimensional orthogonal range reporting structure with optimal query time of $O(\log n + t)$ but with suboptimal $O(n \log^3 n)$ space. This matched two previous results by Chazelle and Guibas [11], and Bozanis et. al [6]. A data structure using optimal $O(n(\log n/ \log\log n)^2)$ space was also known [9], but it only answers queries in $O(\log^{2+\varepsilon} n + t)$ time, where $\varepsilon > 0$ is an arbitrarily small constant. Finally, in two

[1]A point on the boundary of a rectangle is not assumed to be contained in the rectangle.

[2]Throughout this paper, for two real numbers a and b, $a < b$, we use the notation $(a; b)$ to refer to the open interval from a to b.

subsequent papers, Afshani et. al [2, 3] presented an optimal solution for the general problem in 3-D, namely a data structure that uses $O(n(\log n/\log\log n)^2)$ space and has $O(\log n+t)$ query time. They also included optimal solutions for queries with four or five finite boundaries ($O(\log n + t)$ query time and $O(n \log n/\log\log n)$ space), thus completely closing the problem in 3-D.

The best structure for $d \geq 4$ answers queries in $O(\log n(\log n/\log\log n)^{d-3} + t)$ time and uses optimal $O(n(\log n/\log\log n)^{d-1})$ space [3]. For dominance queries, the same query bound can be obtained using $O(n(\log n/\log\log n)^{d-3})$ space; in this case, neither the space bound nor the query bound is known to be optimal.

On the lower bound side, Afshani et. al [3] proved that any structure for d-dimensional dominance reporting that uses nh space must have $\Omega((\log n/\log h)^{\lfloor d/2\rfloor-1} + t)$ query time. This shows with $h = \log^{O(1)} n$, the query time must be $\Omega((\log n/\log\log n)^2 + t)$ for $d = 6$, while for $d = 2, 3$, $O(\log n + t)$ query time is possible. Pin-pointing the exact dimension in which this increase occurs was left as an intriguing open problem.

Rectangle stabbing in one dimensions is the classical *interval stabbing* problem and can be solved with a variety of techniques [12, 13]. The best result uses linear space and has $O(\log n + t)$ query time. It is also possible to reduce it to 2-D dominance reporting: map an input interval $[a, b]$ to point (a, b) and a query point x into the dominance query $(-\infty; x) \times (x; -\infty)$.

In two dimensions, an optimal data structure that uses linear space and has $O(\log n + t)$ query time was developed by Chazelle [8]. Using range trees, this can be generalized to higher dimensions by paying a $\log n$ factor per dimension in space and query time, which gives a data structure with $O(n \log^{d-2} n)$ space and $O(\log^{d-1} n + t)$ query time; it is also possible to obtain $O(\log n(\log n/\log\log n)^{d-2} + t)$ query time using $O(n \log^{d-2+\varepsilon} n)$ space, for any constant $\varepsilon > 0$. This is achieved with range trees of $\log^{\varepsilon} n$ fan out. Note that d-dimensional dominance reporting is also a special case of d-dimensional rectangle stabbing and a d-dimensional stabbing query can be reduced to a $2d$-dimensional dominance query using a similar reduction outlined for the one dimensional case.

Very recently, Afshani et. al [3] proved the first non trivial query time lower bound for rectangle stabbing, showing that with nh space, rectangle stabbing requires $\Omega((\log n/\log h)^{d-1} + t)$ time. Combined with the above reduction, this gives the aforementioned query lower bound for orthogonal range reporting.

1.2 Our Results

Our main upper bound result is a data structures with $O(\log n(\log n/\log\log n)^{d-4+1/(d-2)}+t)$ query time that uses $O(n(\log n/\log\log n)^d)$ space for d-dimensional orthogonal range reporting. Ignoring $\log\log n$ factors, this is a $\log^{1-1/(d-2)} n$ improvement in query time over the fastest previous data structure. For the special case of 4-D dominance, we additionally offer a data structure that has $O(\log n\sqrt{\log n/\log\log n} + t)$ query time and uses optimal, $O(n \log n/\log\log n)$, space.

From a technical point of view, we begin by using a known technique for 3-D dominance, namely shallow cuttings, and combine it with a geometric representation of range trees to solve higher dimensional range reporting. We end up with one instance of a rectangle stabbing problem that allows us to solve many 3-D dominance instances simultaneously. Our key idea is that the rectangle stabbing problem can be made to have greatly sublinear input size, which in turn can be exploited to give us our query speed up. At a high level, this approach is inspired by recent results of Chan et. al [7] for offline 4-D dominance reporting in the word-RAM model.

On the lower bound side, we prove that with nh space, d-dimensional rectangle stabbing requires $\Omega(\log n(\log n/\log h)^{d-2} + t)$ query time. This is optimal for $h = \Omega(\log^{d-2+\varepsilon} n)$ (see Lemma 2). Furthermore, these data structures easily generalizes to answer queries in $O(\log n(\log n/\log h)^{d-2} +$

t) in nh space for any $h > \log^{d-2+\varepsilon} n$, see Lemma 2. Unlike the previous lower bounds [3, 9], we do not use Chazelle's lower bound technique [9]. Instead, we directly use a novel geometric argument. By the simple reduction mentioned earlier, this also gives an improved lower bound of $\Omega(\log n(\log n/\log h)^{\lfloor d/2\rfloor-2}+t)$ for the d-dimensional dominance reporting problem.

We describe our data structures in the next section. We first describe a simplified version of our dominance reporting data structure in 4-D that answers queries in $O(\log^{3/2} n+t)$ time using $O(n\log n)$ space. We believe it carries most of our important ideas and it is significantly easier to understand. In Subsection 2.3, we present our best dominance reporting structure in higher dimensions. Our final structure answers general d-dimensional orthogonal range reporting queries. Due to lack of space, this is presented in the full version of the paper which is included in the appendix. In Section 3, we prove our improved lower bound on rectangle stabbing and orthogonal range reporting in d dimensions.

2 Orthogonal Range Reporting Data Structures

In this section, we describe our new orthogonal range reporting data structures. We start with a brief preliminaries section to introduce some of the basic tools we make use of.

2.1 Preliminaries

We now introduce some convenient notations for talking about special cases of orthogonal range reporting.

Restricted Queries. We adopt the terminology defined in [2]: We use $Q(d,k)$ to refer to the special case of d-dimensional orthogonal range reporting, in which the query rectangles have finite ranges in k of the d dimensions, that is, are unbounded in $d-k$ dimensions. The $Q(2,1)$ and $Q(d,0)$ problems are the 3-sided planar range reporting and d-dimensional dominance reporting problems, respectively.

3-D Dominance. For two points p and q in d dimensions, we say p dominates q if every coordinate of p is greater than that of q. Thus, the dominance reporting problem is the problem of outputting the subset of the input that is dominated by a query point. In 3-D, dominance reporting can be solved optimally using an important combinatorial structure known as shallow cuttings [1].

Consider a set S of points in three dimensions. A *shallow cutting for the h-level of S*, or an *h-shallow cutting* for short, is a set $\mathcal{C}$ of $O(|S|/h)$ points such that any point q that dominates at most h points of S is dominated by a point p in $\mathcal{C}$; furthermore, every point of $\mathcal{C}$ dominates $O(h)$ points of S. The existence of such shallow cuttings was proven by Afshani [1], and more general shallow cuttings have been used extensively in the computational geometry literature (see e.g. [15]). To be useful in data structures, for a given point q, we also need a method that finds the point $p\in\mathcal{C}$ that dominates q. This is done using the following lemma.

Lemma 1. *(Makris and Tsakalidis [14]) Let S be a set of points in 3-D. It is possible to construct a subdivison $\mathcal{A}_S$ of the plane into $O(|S|)$ rectangles with the following property: for any query point q in 3-D, if one projects q onto the plane (i.e., the first two coordinates of q) and finds the rectangle in $\mathcal{A}_S$ that contains the projection, then one can find a point in S that dominates q or conclude that no such point exists. The point location query can be answered in $O(\log|S|)$ time using $O(|S|)$ space.*

Rectangle Stabbing. As discussed, a subproblem that we encounter is rectangle stabbing. It turns out that we need a fast solution for rectangle stabbing when given a budget of nh space. The previous results only focus on the case when h is polylogarithmic but for our purposes, we need to go far beyond polylogarithmic space. Using range trees with fan out h, we can prove the following lemma. We omit the proof here but an interested reader can refer to the full version of the paper in the appendix.

Lemma 2. *For d-dimensional rectangle stabbing on an input of size n, we can build a data structure that has $O(\log n \cdot (\log n/\log h)^{d-2} + t)$ query time and uses $O(nh \log^{d-2} n)$ space, in which $h \geq 2$ is an arbitrary parameter.*

2.2 Simple 4-D Dominance

In this section, we present our simple solution for 4-D dominance that achieves $O(\log^{3/2} n + t)$ query time and uses $O(n \log n)$ space. We obtain the data structure by refining on the standard range tree solution which we describe below.

We use a range tree on the fourth dimension, which is a complete binary tree, T, with the input points $p_1, \ldots, p_n$ stored in sorted order of their fourth coordinate in the leaves. Associate every node v in T with the points $p_i, \ldots, p_j$ stored in the leaves of the subtree rooted at v. We use $S_v = \{p_i, \ldots, p_j\}$ to denote the set of points associated to v. We also associate an interval I_v to v as follows. If $j = n$, then $I_v = (p_{i-1}^{(4)}; \infty)$, otherwise, $I_v = (p_{i-1}^{(4)}; p_j^{(4)}]$, in which $p_i^{(4)}$ denotes the fourth coordinate of a point p_i and $p_0^{(4)} = -\infty$.

Consider a node $u \in T$ and its left child v. We define a $Q(3,0)$ *query on node* u as a $Q(3,0)$ query on the projection of the points of S_v onto the first three dimensions. It turns out that this is the subproblem that we need to solve.

Lemma 3. *A $Q(4,0)$ query $q = (x_1, \ldots, x_4)$ can be answered using a $Q(3,0)$ query on $O(\log n)$ nodes of T that lie on a root to leaf path of T. Furthermore, for every node u on the path we have $x_4 \in I_u$, and $x_4 \notin I_v$ for nodes v not on the path.*

Proof. Let $(-\infty; a)$ be the range of q in the last dimension. Start at the root r of T; let v_1 and v_2 be its left and the right child respectively. Clearly we have $x_4 \in I_r$. We have two cases.

1. a is contained in I_{v_1}: in this case, $(-\infty; a)$ does not intersect I_{v_2} so the right subtree does not contain any output points. We simply recurse on the left child.

2. a is contained in I_{v_2}: in this case, all the points associated to v_1 have their fourth coordinate smaller than a. Thus, we can consider the projection of S_{v_1} and q onto the first three dimensions and find the portion of the output that lies in S_{v_1} using a $Q(3,0)$ query, i.e., a $Q(3,0)$ query on node r. Next, we recurse on v_2.

□

One obvious solution to answer the $Q(3,0)$ queries on nodes of T is to use the data structure of Afshani [1]. But this results in $O(\log^2 n + t)$ query time for the $Q(4,0)$ problem. Until now, this has been the only way to solve 4-D dominance (with a possible increase in fanout of the range tree to $\log^{\varepsilon} n$). We now show that using shallow cuttings, we can do better. We have encapsulated the description of our data structure in two parts: an "outputting" data structures and "finder" data structures. The outputting data structures use finder data structures as black boxes and do the actual reporting of the output points while the finder data structures find a small number of crucial elements in the data structure.

The outputting data structures. Consider a node $u \in T$. Define P_u as the projection of S_v into the first three dimensions in which v is the left child of u. With this notation, a $Q(3,0)$ query on u is equivalent to a dominance query on P_u. Now for every $u \in T$, we build an h-shallow cutting $\mathcal{C}_u$ for P_u in which h is a parameter to be determined later; the only restriction that we place is that $h > \log^2 n$. For every point $p \in \mathcal{C}_u$, we implement an optimal dominance reporting structure [1] on the subset, $P_u(p) \subseteq P_u$ that is dominated by p. Finally, we store the $O(\log^2 n/\log\log n + t)$ query time and $O(n \lg n/\lg\lg n)$ space $Q(4,0)$ data structure of Afshani et. al [3] on the entire input set.

Generating the output of the query. Let $q = (x_1, \ldots, x_4)$ be a $Q(4,0)$ query and define $q' = (x_1, x_2, x_3)$. By Lemma 3, we can answer q by querying q' on $O(\log n)$ nodes, $u_1, \ldots, u_{O(\log n)}$, of T. Assume we have a finder structure that for each u_i can find a point $p_i \in \mathcal{C}_{u_i}$ that dominates q' or conclude that no such point exists. If for some i, no point in $\mathcal{C}_{u_i}$ dominates q', then it follows that the output size is at least $h > \log^2 n$; in this case, we simply query the structure of Afshani et. al [3] to report our output. This takes $O(t)$ time. Otherwise, to find points in P_{u_i} that are dominated by q', we query the dominance structure that stores $P_{u_i}(p_i)$; this takes $O(\log|P_{u_i}(p_i)| + t') = O(\log h + t')$ time in which t' is the output size of q' on u_i. Over all the $O(\log n)$ nodes, this takes $O(\log n \cdot \log h + t)$ time.

The finder data structures. Consider a node $u \in T$ and the h-shallow cutting $\mathcal{C}_u$ for P_u. Using Lemma 1, we decide if a point in $\mathcal{C}_u$ dominates q' and if so find it using a point location query on $\mathcal{A}_{\mathcal{C}_u}$. Unfortunately, this naive approach only yields $O(\log^2 n)$ query time. Our key idea is to perform all these point location queries *simultaneously*. We lift each rectangle $[x_1; x_2] \times [y_1; y_2]$ in $\mathcal{A}_{\mathcal{C}_u}$ to 3-D by forming the 3-D rectangle $[x_1; x_2] \times [y_1; y_2] \times I_u$. This creates $O(|P_u|/h)$ rectangles for a given node u and thus $O(n \log n/h)$ rectangles in total. We collect all the rectangles (over all the nodes in T) and store them in a 3-D rectangle stabbing data structure given by Lemma 2, with the space usage set to $O(n \log n)$.

The finder query. Let $q = (x_1, \ldots, x_4)$ be the query and define $q' = (x_1, x_2, x_3)$ and $q'' = (x_1, x_2, x_4)$. We claim that to obtain the result of a point location query for (x_1, x_2) on all the sets $\mathcal{A}_{\mathcal{C}_u}$, where we are to perform a $Q(3,0)$ query on u, it suffices to query the stabbing data structure with q'': By Lemma 3, we have $x_4 \in I_u$ for nodes u on the path where we are to query the $Q(3,0)$ structures and $x_4 \notin I_v$ for nodes v not on the path. By Lemma 1, we need to find the rectangle r that contains the point (x_1, x_2) which is equivalent to finding the 3-D rectangle $r \times I_u$ that contains q''. Thus, the point location query for u can be answered by looking at the result of the stabbing query.

Query time analysis. First, observe that for two nodes u_1 and u_2 at the same depth of T, the intervals I_{u_1} and I_{u_2} are disjoint. Thus, the output size of the stabbing query is $O(\log n)$. Since the ratio between the input size and space usage is $\Omega(h)$, by Lemma 2, the query time is $O(\log n \cdot \log_h n + \log n) = O(\log n \cdot \log_h n)$. We pick $h = 2^{\sqrt{\log n}}$ and obtain the following.

Theorem 1. *The 4-D dominance problem can be solved with $O(n \log n)$ space and $O(\log^{3/2} n + t)$ query time.*

2.3 Higher Dimensions

In this section we show how to extend the simple 4-D dominance result to d-dimensional dominance reporting. Before we begin, we note that similar to the 4-D case, the most interesting case of the problem is when the output size is small. For example, if we realize that the output size is larger than say $\log^{d-1} n$, we can switch to the best previous result and the query time will be optimal.

As with the 4-D case, we start with range trees. However, to get the best performance, we use range trees with large fan outs. Let $\alpha = \log^{\varepsilon} n$ in which ε is a sufficiently small constant to be determined later. We build a range tree T_d on the d-th dimension with fan out α (i.e., all nodes except leaves have α children). For a node $v_d \in T_d$, the set S_{v_d} and the interval I_{v_d} is defined as in our simple 4-D dominance solution, using the value of the d-th coordinate of the points. We also store a pointer from v_d to a range tree T_{d-1,v_d} with fan out α, built on S_{v_d} and on the $(d-1)$-th dimension. We continue this recursive construction until we reach dimension three. No range tree is built on the first three dimensions. At the end, we have a nested hierarchy of range trees, $T_{i,v_d,v_{d-1},\dots,v_{i+1}}$, for $4 \le i \le d-1$, in which v_{i+1} is a node in the range tree $T_{i+1,v_d,v_{d-1},\dots,v_{i+2}}$. A node $v_i \in T_{i,v_d,v_{d-1},\dots,v_{i+1}}$ is associated with a point set S_{v_i} and consists of all the points whose j-th coordinate, $i \le j \le d$, is inside the interval I_{v_j}. To every node v_4 in a range tree $T_{4,v_d,\dots,v_5}$, we associate a subproblem $Q_{3,v_d,\dots,v_4}$ on S_{v_4}. Note that S_{v_4} is the set of input points that lie in the region $R(Q) = (-\infty;\infty) \times (-\infty;\infty) \times (-\infty;\infty) \times I_4 \times \cdots \times I_{v_d}$. Before describing this subproblem, we need to examine how the configuration of the range trees is traversed at query time.

Query traversal. Let $q = (x_1, \dots, x_d)$ be the point representing a $Q(d, 0)$ query. We define the *query traversal* to be the set of subproblems that are reached using the following procedure. Start from the root r of the range tree T_d. Let v be the child of r such that $x_d \in I_v$. We follow the link to the range tree $T_{d-1,v}$, traverse it and then recurse on v; the recursion stops at the subproblems. To be more precise, assume a node $v_i \in T_{i,v_d,v_{d-1},\dots,v_{i+1}}$ is reached in this traversal. We follow two pointers for v_i: (i) If $i = 4$, then $Q_{3,v_d,\dots,v_4}$ is added to the query traversal; otherwise, we follow the pointer to the root of the range tree $T_{i-1,v_d,v_{d-1},\dots,v_i}$ and traverse it and (ii) we find a child u of v_i such that $x_i \in I_u$ and then recurse on u.

The subproblem. Consider a $Q := Q_{3,v_d,\dots,v_4}$ subproblem that is in the query traversal for the query q. Observe that the crucial property of the query traversal is that if Q is in the query traversal of q, then $q \in R(Q)$. The children of v_i decompose I_{v_i} into α smaller intervals which in geometric terms corresponds to cutting the i-th dimension of $R(Q)$ into α smaller parts. Based on this division, for every point $p = (p_1, \cdots, p_d) \in R(Q)$, we associate a *child-coordinate point* $f_Q(p)$, $f_Q(p) \subset [\alpha]^{d-3}$, to p in which the i-th coordinate is the index of the child of v_i whose interval contains p_i. Now, the subproblem Q is defined as the problem of outputting all the points $p \in S_{v_4}$ such that the point (x_1, x_2, x_3) dominates the projection of p into the first three dimensions and every coordinate of $f_Q(q)$ is larger than that of $f_Q(p)$ (i.e., $f_Q(q)$ dominates $f_Q(p)$).

Before describing our data structures, we need an equivalent of Lemma 3.

Lemma 4. *A $Q(d, 0)$ query q can be solved by solving all the subproblems that are reached during the query traversal of q. For each such subproblem Q, we have $q \in R(Q)$. Furthermore, the number of subproblems Q such that $q \in R(Q)$ is $O(\log_{\alpha}^{d-3} n)$.*

Proof. Consider a point $p = (p_1, \dots, p_d) \in S$ that is dominated by q. We claim that there exists a subproblem Q in the query traversal of q that outputs p. To see this, we begin from the root r of T_d and essentially follow the procedure that builds the query traversal. Assume we have reached a node $v_i \in T_{i,v_d,v_{d-1},\dots,v_{i+1}}$. In contrast to the procedure that builds the query traversal, we follow a single pointer at each step: (i) if there is a child u of v_i such that p_i and $x_i \in I_u$ then we follow the pointer to u (ii) otherwise, if $i > 4$, we follow the pointer to $T_{i-1,v_d,v_{d-1},\dots,v_i}$ but if $i = 4$, then $Q := Q_{3,v_d,\dots,v_4}$ is the desired subproblem. It is easily checked that $q \in R(Q)$ and that every coordinate of $f_Q(q)$ is greater than that of $f_Q(p)$.

It remains to prove that the number of subproblems Q, s.t. $q \in R(Q)$ is $O(\log_\alpha^{d-3})$. In a range tree with fanout α, there are only $O(\log_\alpha n)$ nodes v such that I_v contains a given point. Since we have build $d-3$ levels of range trees, our claim can be proven by a simple inductive argument. $\square$

Now we are ready to describe our data structures.

The outputting data structures. Consider a $Q := Q_{3,v_d,\ldots,v_4}$ subproblem. Let P be the projection of S_{v_4} onto the first three dimensions. We partition P into $\beta := \alpha^{d-3}$ subsets as follows. For every element $\gamma \in [\alpha]^{d-3}$, we place all the points $p \in P$ with $f_Q(p) = \gamma$ in the set P_γ. We also define the set $\hat{P}_\gamma$ to be the union of all the sets $P_{\gamma'}$, $\gamma' \in [\alpha]^{d-3}$, in which γ dominates γ'. We build three categories of shallow cuttings: First, for every P_γ, we build a $\log^d n$-shallow cutting $\mathcal{C}_{d,\gamma}$ and then store the points dominated by every point $p \in \mathcal{C}_{d,\gamma}$ in an optimal 3-D dominance reporting structure. Second, for every $\hat{P}_\gamma$, we build an h-shallow cutting $\hat{\mathcal{C}}_{h,\gamma}$; the value of h will be determined later and the only restriction that we impose is that $h > \log^{2d} n$. Finally, for every point $p \in \hat{\mathcal{C}}_{h,\gamma}$, we build a $\log^{d-1} n$-shallow cutting $\hat{\mathcal{C}}_{d-1,\gamma,p}$, as well as the corresponding point location data structure of Lemma 1, on the subset of $\hat{P}_\gamma$ dominated by p. Consider a point $\gamma' \in [\alpha]^{d-3}$ that is dominated by γ. We know $P_{\gamma'} \subset \hat{P}_\gamma$ and thus every point $\hat{p} \in \hat{\mathcal{C}}_{d-1,\gamma,p}$ contains $O(\log^{d-1} n)$ points in $P_{\gamma'}$ which implies there exists a point $p \in \mathcal{C}_{d,\gamma'}$ such that p dominates $\hat{p}$. We place a pointer from $\hat{p}$ to p. Since the sets P_γ partition P, the first category of shallow cuttings take up linear space. The number of cells in the second category is $O(\alpha^{d-3}|S_{v_4}|/h)$ which is sublinear by our assumptions on h and α. Finally, the number of points in $\hat{\mathcal{C}}_{h,\gamma}$ is $O(|S_{v_4}|/h)$ and for each we store a shallow cutting that has $O(h/\log^{d-1} n)$ size and for each point in the second shallow cutting we store $O(\alpha^{d-3})$ additional pointers; thus, the total space is $O(|S_{v_4}|)$ for the subproblem Q. Over all the subproblems this sums up to $O(n \log_\alpha^{d-3} n)$.

Answering output queries. Let $q = (x_1, \ldots, x_d)$ be a $Q(d, 0)$ query and let $q' = (x_1, x_2, x_3)$. By Lemma 4, we need to answer q on $O(\log_\alpha^{d-3} n)$ subproblems. Consider one such subproblem $Q := Q_{3,v_d,\ldots,v_4}$. If we let $\gamma = f_Q(q)$ the subproblem translates to outputting the subset of $\hat{P}_\gamma$ that is dominated by q'. If no point in $\hat{\mathcal{C}}_{h,\gamma}$ dominates q', then the output size is larger than $h > \log^{d-1} n$, which means we can switch to the previous best result on dominance reporting. Assume we have a finder data structures that can find a point $p \in \hat{\mathcal{C}}_{h,\gamma}$ that dominates q' (or tell us no such point exists). Using the point location data structure implemented for $\hat{\mathcal{C}}_{d-1,\gamma,p}$ we can check if there exists a point $p' \in \hat{\mathcal{C}}_{d-1,\gamma,p}$ that dominates q'. If no such point exists, then the output size is larger than $\log^{d-1} n$ and we are done. Thus, assume we locate such a point p'. Note that finding p' takes $O(\log|\hat{\mathcal{C}}_{d-1,\gamma,p}|) = O(\log h)$ time by Lemma 1. Next, for every $\gamma' \in [\alpha]^{d-3}$, that is dominated by γ, we follow the pointer from p' to the corresponding point $p'' \in \mathcal{C}_{d,\gamma'}$ that dominates p'. As p'' also dominates q', the query can answered by querying the 3-D dominance data structure built for p''. The query time for Q becomes $O(\log h + \alpha^{d-3} \log\log n)$ which results in the overall query time of $O((\log h + \alpha^{d-3} \log\log n) \log_\alpha^{d-3} n)$.

The finder data structures. Consider a subproblem $Q := Q_{3,v_d,\ldots,v_4}$ and the shallow cutting $\hat{\mathcal{C}}_{h,\gamma}$ for some $\gamma \in [\alpha]^{d-3}$. We build the corresponding orthogonal planar subdivision $\hat{\mathcal{A}}_{h,\gamma}$ and lift a rectangle $r \in \hat{\mathcal{A}}_{h,\gamma}$ to a $d-1$-dimensional rectangle $L_Q(r) = r \times I_{v_4} \times \cdots \times I_{v_d}$. The total number of rectangles created from all the subproblems is $O(n \log_\alpha^{d-3} n/h)$ and we place them all in a $(d-1)$-dimensional rectangle stabbing structure given by Lemma 2 in which the space usage is set to $O(n)$.

The finder query. Let $q = (x_1, \dots, x_d)$ be a query and define $q' = (x_1, x_2, x_4, \dots, x_d)$. We claim the finder queries for all the subproblems can be answered by looking at the result of q' on the rectangle stabbing data structure. Consider a subproblem $Q := Q_{3,v_d,\dots,v_4}$ that is reached during the query traversal of q. By Lemma 4, $q \in R(Q)$. By Lemma 1, to answer the finder query for Q, it suffices to do a point location query on $\hat{\mathcal{A}}_{h,\gamma}$ using (x_1, x_2) as the query point. Observe that if a rectangle $r \in \hat{\mathcal{A}}_{h,\gamma}$ contains (x_1, x_2), then $L_Q(r)$ contains q'. To bound the running time, notice that the number of different values of γ is α^{d-3}, and the output size of the stabbing query is at most $O(\alpha^{d-3} \log_\alpha^{d-3} n) = O(\log^{d-3} n)$ by Lemma 4. The input size of the stabbing data structure is $O((n \log_\alpha^{d-3} n/h))$ while its space usage is set to n. The ratio of the space usage to the input size is thus $\Omega(h/\log_\alpha^{d-3} n) = \Omega(\sqrt{h})$ which means by Lemma 2, the query time is $O(\log n(\log_h n)^{d-3} + \log^{d-3} n)$ (remember, we have a $d-1$-dimensional stabbing query).

Optimizing the running time. We get that the total query time is $O(\log n(\log_h n)^{d-3} + \log^{d-3} n + (\log h + \alpha^{d-3} \log\log n) \log_\alpha^{d-3} n)$. By setting $\alpha = \log^\varepsilon n$ for a small enough ε, and picking h such that $\log h = \log^{1/(d-2)} n\, (\log\log n)^{(d-3)/(d-2)}$ we get the following:

Theorem 2. *The d dimensional dominance reporting problem can be solved using $O(n(\log n/\log\log n)^{d-3})$ space and with query time of $O(\log n(\log n/\log\log n)^{d-4+1/(d-2)} + t)$.*

In the full version of the paper, which is included in the appendix, we show that with some modifications, we can in fact answer $Q(d, d-3)$ queries. Thus, we get the following theorem.

Theorem 3. *The $Q(d, d-3)$ problem can be solved using $O(n(\log n/\log\log n)^{d-3})$ space and with query time of $O(\log n(\log n/\log\log n)^{d-4+1/(d-2)} + t)$.*

By a standard application of range trees with fan out $\alpha = \log^\varepsilon n$ we get the following theorem.

Theorem 4. *Orthogonal range reporting can be solved using $O(n(\log n/\log\log n)^{d})$ space and with query time of $O(\log n(\log n/\log\log n)^{d-4+1/(d-2)} + t)$.*

3 A Tight Lower Bound for Rectangle Stabbing

In this section we prove our lower bound for the rectangle stabbing problem. Our model of computation is identical to one used by Chazelle [9] but for sake of completeness, we include a brief description.

Consider an input set S of n elements. A data structure is a collection of (memory) cells and its space complexity is the number of cells used. Each cell can store an input element or an auxiliary data type. Furthermore, each cell can have two pointers to two other cells. Such a data structure can be represented by a graph G with $V(G)$ being the set of cells used in the data structure. A pointer from a cell u to a cell v is represented with a directed edge from u to v. The crucial restriction is that a cell can only be accessed through pointers and thus random accesses are disallowed. (It is assumed that we always begin by a pointer to a root cell.) A query q is answered by exploring (a subgraph of) G. In this model, to output an element $p \in S$, the query algorithm must visit a cell that stores p. We do not impose any restriction on how the algorithm navigates G to reach such a node. We measure the query time by the number of vertices of G visited by the query algorithm.

Consider a data structure D that answers rectangle stabbing queries in d dimensions in the above model. Let nh be the number of cells used by D. Furthermore, assume D answers every query in $f(n) + Ct$ time in which t is the output size, C is a constant, and $f(n)$ is the search time of the query. Our goal is to prove that $f(n) = \Omega(\log n \cdot \log_h^{d-2} n)$.

We build an input set, Q, of rectangles that is in fact a simplification of the ones used before [3, 9, 10]. Consider a d-dimensional cube $\mathcal{R}$ with side length m, $m > \sqrt{n}$, in which m is a parameter to be determined later. Let $r = c_r h^4$, in which c_r is a large enough constant, and let $\mathrm{Log}_r m = \lfloor \log_r m \rfloor - 1$. For every choice of $d-1$ indices, $1 \le i_1, \cdots, i_{d-1} \le \mathrm{Log}_r m$, we divide the j-th dimension of $\mathcal{R}$ into r^{i_j} equal pieces for $1 \le j \le d-1$, and the d-th dimension of $\mathcal{R}$ into $\lfloor m^{d-1}/r^{i_1+\cdots+i_{d-1}} \rfloor$ equal pieces. The number of such choices is $k = \mathrm{Log}_r^{d-1} m$. With each such choice, we create between $m/2$ and m rectangles and thus $\Theta(mk)$ rectangles are created in total. We pick m such that this number equals n, thus $n = \Theta(mk)$. Also, note that the volume of each rectangle is between m^{d-1} and $2m^{d-1}$. The crucial property of this input set is the following.

Observation 1. *The volume of the intersection of any two rectangles in $\mathcal{R}$ is at most $2m^{d-1}/r$.*

Unlike the previous attempts, our query set is very simple: a single query q chosen uniformly at random inside $\mathcal{R}$. Note that the output size of q is k. The rest of this section is devoted to prove that with positive probability answering q needs $\Omega(k \log r) = \Omega(\log_r^{d-1} m \log r) = \Omega(\log n \cdot \log_h^{d-2} n)$ time which proves our claim.

Let c be a parameter to be determined later. Call a rectangle b *expensive*, if there are more than ch cells in D that store b.

Observation 2. *There are at most n/c expensive rectangles.*

For a vertex $u \in G$, let $\mathrm{In}(u)$ denote the set of vertices in G that have a directed path of size at most $\log h$ to u. Similarly, define $\mathrm{Out}(u)$ to be the set vertices in G that can be reached from u using a path of size at most $\log h$. Call a cell v *popular* if $\mathrm{In}(v) > ch^2$. Similarly, we say a rectangle b is popular if there is a popular cell that stores b.

Lemma 5. *There are at most n/c popular rectangles.*

Proof. As each vertex in G has two out edges, the size of $\mathrm{Out}(u)$ is at most $2^{\log h} = h$. Also, if $v \in \mathrm{Out}(u)$, then $u \in \mathrm{In}(v)$ and vice versa. Thus,

$$\sum_{v \in G} |\mathrm{In}(v)| = \sum_{u \in G} |\mathrm{Out}(u)| \le nh^2.$$

This implies the number of vertices v with $\mathrm{In}(v) > ch^2$ is at most n/c. Our claim follows since the number of popular rectangles cannot exceed the number of popular cells. □

Lemma 6. *With probability at least 3/5, q is enclosed by at most $k/4$ rectangles that are either expensive or popular, if c is to be a chosen sufficiently large constant.*

Proof. By Observation 2 and Lemma 5, the number of expensive or popular rectangles is at most $2n/c$. As each rectangle has volume at most $2m^{d-1}$, the total volume of expensive or popular rectangles is at most $4m^{d-1}n/c$. Let A be the region of $\mathcal{R}$ that contains all the points covered by more than $k/4$ expensive or popular rectangles. We have,

$$\mathrm{Vol}(A)k/4 \le 4m^{d-1}n/c = \Theta(m^d k/c)$$

which implies $\mathrm{Vol}(A) = O(m^d/c) < 2m^d/5$ if c is large enough. Thus, with probability at least 3/5, q will not be picked in A. □

Let S' be the subset of rectangles that are neither expensive nor popular and let $n' = |S'|$. We say two rectangles $b_1, b_2 \in S'$ are *close*, if there exists a vertex $u \in G$ such that from u we can reach a cell that stores b_1 and a cell that stores b_2 with directed paths of size at most $\log h$ each.

Lemma 7. *A rectangle $b \in S'$ can be close to at most c^2h^4 other rectangles in S'.*

Proof. A rectangle $b \in S'$ is stored in at most ch vertices of G and for each such vertex v we know $\text{In}(v) \leq ch^2$. Since for every $u \in G$ we have $\text{Out}(u) \leq h$, it follows that v can be close to at most ch^3 vertices in G. Multiplying this number by ch gives the maximum number of rectangles that can be close to b. □

Consider a rectangle $b \in S'$. Let B be the subset of rectangles that are close to b. We call $\cup_{b' \in B}(b \cap b')$ the *close region* of b and denote it with C_b.

Lemma 8. *With probability at least 1/5, answering q needs $\Omega(\log n \cdot \log_h^{d-2} n)$ time.*

Proof. Consider a rectangle $b \in S'$ and set B defined above. By Lemma 7, $|B| \leq c^2h^4$. By Observation 1, for every $b' \in B$, $\text{Vol}(b \cap b') \leq 2m^{d-1}/r$ and thus

$$\text{Vol}(C_b) \leq \sum_{b' \in B} \text{Vol}(b \cap b') \leq 2c^2h^4m^{d-1}/r = 2c^2m^{d-1}/c_r.$$

As there are at most n rectangles in S', the sum of the volumes of the close regions of all the rectangles in S' is at most $n \cdot 2c^2m^{d-1}/c_r = \Theta(c^2km^d/c_r)$. Consider the region R of all the points p such that p is inside the close region of at least $k/4$ rectangles of S'. We have,

$$\text{Vol}(R)k/4 \leq \sum_{b \in S'} \text{Vol}(C_b) \leq \Theta(c^2km^d/c_r)$$

which means $\text{Vol}(R) = O(c^2m^d/c_r)$. If we choose c_r large enough, this volume is less than $2m^d/5$, which means with probability at least 3/5, the selected query will not be inside R. Combined with Lemma 6, this gives the following: with probability at least 1/5, q will be inside at least $3k/4$ rectangles that are in S' (by Lemma 6) and it will also be inside the close region of at most $k/4$ rectangles. Let q be the query when this happens.

Consider the subgraph H_q of G that is explored by the query procedure while answering q. Assume q needs to output rectangles $b_1, \cdots, b_{k'}$ from S'. We have $k' \geq 3k/4$. Let v_i be a node in H_q that stores b_i, $1 \leq i \leq k'$. Also, let w_i be the set of nodes in H_q that are reachable by traversing up to $\log h$ edges from v_i, where the direction of edged have been reversed. If two sets w_i and w_j, $1 \leq i < j \leq k'$, share a common vertex, it means that q is inside the close region of both b_i and b_j. However, we know that q is inside the close region of at most $k/4$ rectangles. This means that there are at least $k/2$ sets w_i that do not share any vertices. Thus, the size of H_q is at least $k/2 \cdot \log h = \Omega(\log n \cdot \log_h^{d-2} n)$. □

4 Conclusions

We believe our most surprising result is the $O(n \log n / \log\log n)$ space structure for 4-D dominance reporting that can answer queries in $O(\log n \sqrt{\log n / \log\log n} + t)$ time. The existence of such a structure raises the obvious open question that whether this query time can be reduced to $O(\log n + t)$. Unfortunately, with our techniques this does not seem possible. Perhaps the next big development in this area needs to come from the lower bound side and it is possible that our new lower bound argument could be useful in this venue; however, it appears the biggest challenge in the lower bound side is building the right "bad" query and inputs sets.

References

[1] P. Afshani. On dominance reporting in 3D. In *ESA'08: Proc. of the 16th conference on Annual European Symposium*, pages 41–51, 2008.

[2] P. Afshani, L. Arge, and K. D. Larsen. Orthogonal range reporting in three and higher dimensions. In *FOCS'09: Proc. of the 50th Annual Symposium on Foundations of Computer Science*, pages 149–158, 2009.

[3] P. Afshani, L. Arge, and K. D. Larsen. Orthogonal range reporting: query lower bounds, optimal structures in 3-d, and higher-dimensional improvements. In *SCG'10: Proc. of the 26th Annual Symposium on Computational Geometry*, pages 240–246, 2010.

[4] P. K. Agarwal. Range searching. In J. E. Goodman and J. O'Rourke, editors, *CRC Handbook of Discrete and Computational Geometry*. CRC Press, Inc., 2004.

[5] P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*. AMS Press, 1999.

[6] P. Bozanis, N. Kitsios, C. Makris, and A. Tsakalidis. New results on intersection query problems. *The Computer Journal*, 40:22–29, 1997.

[7] T. M. Chan, K. G. Larsen, and M. Pătraşcu. Orthogonal range searching on the RAM, revisited. In *SCG'11: Proc. of the 27th Annual Symposium on Computational Geometry*, pages 1–10, 2011.

[8] B. Chazelle. Filtering search: a new approach to query answering. *SIAM Journal on Computing*, 15(3):703–724, 1986.

[9] B. Chazelle. Lower bounds for orthogonal range searching: I. the reporting case. *Journal of the ACM*, 37(2):200–212, 1990.

[10] B. Chazelle. Lower bounds for off-line range searching. In *STOC '95: Proc. of the 27th annual ACM symposium on Theory of computing*, pages 733–740, 1995.

[11] B. Chazelle and L. J. Guibas. Fractional cascading: II. Applications. *Algorithmica*, 1:163–191, 1986.

[12] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2008.

[13] E. Hanson. The interval skip list: A data structure for finding all intervals that overlap a point. In *Algorithms and Data Structures*, volume 519 of *Lecture Notes in Computer Science*, pages 153–164. Springer Berlin / Heidelberg, 1991.

[14] C. Makris and A. Tsakalidis. Algorithms for three-dimensional dominance searching in linear space. *Inf. Process. Lett.*, 66(6):277–283, 1998.

[15] J. Matoušek. Reporting points in halfspaces. *Computational Geometry Theory and Applications*, 2(3):169–186, 1992.

[16] E. M. McCreight. Priority search trees. *SIAM Journal on Computing*, 14(2):257–276, 1985.

[17] R. E. Tarjan. A class of algorithms that require nonlinear time to maintain disjoint sets. *Journal of Computer and System Sciences*, 18:110–127, 1979.

A Introduction

In this paper we study two fundamental range searching problems, rectangle stabbing and orthogonal range reporting. In orthogonal range reporting, the goal is to store an input set, S, of n points in d-dimensional space in a data structure such that the t points contained in[3] an axis-aligned query rectangle can be reported efficiently. Rectangle stabbing is the "dual" problem in which the goal is to store an input set, Q, of n rectangles in d-dimensional space such that the t rectangles that contain a query point can be reported efficiently.

Orthogonal range reporting is a central problem in several fields, including spatial databases and computational geometry, and it has been studied extensively [4, 5]. Rectangle stabbing (a.k.a point enclosure) is also one of the classical problems in computational geometry [12]. We study both problems in the pointer machine model of computation [17]. In 2-D, orthogonal range reporting was completely characterized more than two decades ago. Very recently, a complete characterization of the three-dimensional case was also achieved. In higher dimensions, however, the status of the problem is much more mysterious and it has been the topic of some speculations. Unlike orthogonal range reporting, only the two-dimensional case was characterized for rectangle stabbing.

In this paper we present two main results. For orthogonal range reporting, we improve on the best known query time in dimensions four and above. Curiously, the improvement grows with dimension. For rectangle stabbing, we obtain the first tight query time lower bound. By known techniques, this gives an improved query time lower bound for orthogonal range reporting as well.

A.1 Previous Results

In this section, we review the previous results on orthogonal range reporting and rectangle stabbing in the pointer machine. We only review the results most related to ours, that is, results for static structures that answer queries in poly-logarithmic time and with near-linear space usage. We refer the reader to the surveys [4, 5] for further results.

Before reviewing the previous results, we need to introduce two important special cases of the orthogonal range reporting problem. In two dimensions, a query is called a *three-sided* query if at most three of the query boundaries are finite (e.g., $(a; b) \times (-\infty; c)$[4]). A query in d dimensions is called a *dominance* query if there is at most one finite boundary at each dimension.

For 2-dimensional orthogonal range reporting, the complexity of the problem was completely settled more than two decades years ago. In [8], Chazelle provided a structure that can answer queries in $O(\log n + t)$ time while using $O(n \log n / \log \log n)$ space, where t denotes the output size. This is optimal, since Chazelle later proved that any structure for d-dimensional orthogonal range reporting that answers queries in $O(\log^{O(1)} n + t)$ time must use $\Omega(n(\log n / \log \log n)^{d-1})$ space [9]. For three-sided queries, McCreight [16] presented the priority search tree, that can answer queries in optimal $O(\log n + t)$ time and uses linear space.

Until recently the complexity of the problem remained unresolved in 3-D. For 3-D dominance queries, Afshani [1] was the first to present an optimal structure that uses linear space and answers queries in $O(\log n + t)$ time. Using a standard reduction, this provides a 3-dimensional orthogonal range reporting structure with optimal query time of $O(\log n + t)$ but with suboptimal $O(n \log^3 n)$ space. This matched two previous results by Chazelle and Guibas [11], and Bozanis et. al [6]. A data structure using optimal $O(n(\log n / \log \log n)^2)$ space was also known [9], but it only answers queries in $O(\log^{2+\varepsilon} n + t)$ time, where $\varepsilon > 0$ is an arbitrarily small constant. Finally, in two

[3] A point on the boundary of a rectangle is not assumed to be contained in the rectangle.

[4] Throughout this paper, for two real numbers a and b, $a < b$, we use the notation $(a; b)$ to refer to the open interval from a to b.

subsequent papers, Afshani et. al [2, 3] presented an optimal solution for the general problem in 3-D, namely a data structure that uses $O(n(\log n/\log\log n)^2)$ space and has $O(\log n+t)$ query time. They also included optimal solutions for queries with four or five finite boundaries ($O(\log n + t)$ query time and $O(n \log n/ \log\log n)$ space), thus completely closing the problem in 3-D.

The best structure for $d \geq 4$ answers queries in $O(\log n(\log n/\log\log n)^{d-3} + t)$ time and uses optimal $O(n(\log n/\log\log n)^{d-1})$ space [3]. For dominance queries, the same query bound can be obtained using $O(n(\log n/\log\log n)^{d-3})$ space; in this case, neither the space bound nor the query bound is known to be optimal.

On the lower bound side, Afshani et. al [3] proved that any structure for d-dimensional dominance reporting that uses nh space must have $\Omega((\log n/\log h)^{\lfloor d/2\rfloor-1} + t)$ query time. This shows with $h = \log^{O(1)} n$, the query time must be $\Omega((\log n/\log\log n)^2 + t)$ for $d = 6$, while for $d = 2, 3$, $O(\log n+t)$ query time is possible. Pin-pointing the exact dimension in which this increase occurs was left as an intriguing open problem.

Rectangle stabbing in one dimensions is the classical *interval stabbing* problem and can be solved with a variety of techniques [12, 13]. The best result uses linear space and has $O(\log n + t)$ query time. It is also possible to reduce it to 2-D dominance reporting: map an input interval $[a, b]$ to point (a, b) and a query point x into the dominance query $(-\infty; x) \times (x; -\infty)$.

In two dimensions, an optimal data structure that uses linear space and has $O(\log n + t)$ query time was developed by Chazelle [8]. Using range trees, this can be generalized to higher dimensions by paying a $\log n$ factor per dimension in space and query time, which gives a data structure with $O(n \log^{d-2} n)$ space and $O(\log^{d-1} n + t)$ query time; it is also possible to obtain $O(\log n(\log n/\log\log n)^{d-2} + t)$ query time using $O(n \log^{d-2+\varepsilon} n)$ space, for any constant $\varepsilon > 0$. This is achieved with range trees of $\log^{\varepsilon} n$ fan out. Note that d-dimensional dominance reporting is also a special case of d-dimensional rectangle stabbing and a d-dimensional stabbing query can be reduced to a $2d$-dimensional dominance query using a similar reduction outlined for the one dimensional case.

Very recently, Afshani et. al [3] proved the first non trivial query time lower bound for rectangle stabbing, showing that with nh space, rectangle stabbing requires $\Omega((\log n/\log h)^{d-1} + t)$ time. Combined with the above reduction, this gives the aforementioned query lower bound for orthogonal range reporting.

A.2 Our Results

Our main upper bound result is a data structures with $O(\log n(\log n/\log\log n)^{d-4+1/(d-2)}+t)$ query time that uses $O(n(\log n/\log\log n)^d)$ space for d-dimensional orthogonal range reporting. Ignoring $\log\log n$ factors, this is a $\log^{1-1/(d-2)} n$ improvement in query time over the fastest previous data structure. For the special case of 4-D dominance, we additionally offer a data structure that has $O(\log n\sqrt{\log n/\log\log n} + t)$ query time and uses optimal, $O(n \log n/\log\log n)$, space.

From a technical point of view, we begin by using a known technique for 3-D dominance, namely shallow cuttings, and combine it with a geometric representation of range trees to solve higher dimensional range reporting. We end up with one instance of a rectangle stabbing problem that allows us to solve many 3-D dominance instances simultaneously. Our key idea is that the rectangle stabbing problem can be made to have greatly sublinear input size, which in turn can be exploited to give us our query speed up. At a high level, this approach is inspired by recent results of Chan et. al [7] for offline 4-D dominance reporting in the word-RAM model.

On the lower bound side, we prove that with nh space, d-dimensional rectangle stabbing requires $\Omega(\log n(\log n/\log h)^{d-2} + t)$ query time. This is optimal for $h = \Omega(\log^{d-2+\varepsilon} n)$ (see Lemma 2). Furthermore, these data structures easily generalizes to answer queries in $O(\log n(\log n/\log h)^{d-2} +$

t) in nh space for any $h > \log^{d-2+\varepsilon} n$, see Lemma 2. Unlike the previous lower bounds [3, 9], we do not use Chazelle's lower bound technique [9]. Instead, we directly use a novel geometric argument. By the simple reduction mentioned earlier, this also gives an improved lower bound of $\Omega(\log n(\log n/\log h)^{\lfloor d/2\rfloor-2} + t)$ for the d-dimensional dominance reporting problem.

We describe our data structures in the next section. We first describe a simplified version of our dominance reporting data structure in 4-D that answers queries in $O(\log^{3/2} n + t)$ time using $O(n \log n)$ space. We believe it carries most of our important ideas and it is significantly easier to understand. In Subsection B.3, we present our best dominance reporting structure in higher dimensions. Our final structure appears in Subsection B.4 and it answers general d-dimensional orthogonal range reporting queries. In Section C, we prove our improved lower bound on rectangle stabbing and orthogonal range reporting in d dimensions.

B Orthogonal Range Reporting Data Structures

In this section, we describe our new orthogonal range reporting data structures. We start with a brief preliminaries section to introduce some of the basic tools we make use of.

B.1 Preliminaries

We now introduce some convenient notations for talking about special cases of orthogonal range reporting.

Restricted Queries. We adopt the terminology defined in [2]: We use $Q(d, k)$ to refer to the special case of d-dimensional orthogonal range reporting, in which the query rectangles have finite ranges in k of the d dimensions, that is, are unbounded in $d-k$ dimensions. The $Q(2, 1)$ and $Q(d, 0)$ problems are the 3-sided planar range reporting and d-dimensional dominance reporting problems, respectively.

3-D Dominance. For two points p and q in d dimensions, we say p dominates q if every coordinate of p is greater than that of q. Thus, the dominance reporting problem is the problem of outputting the subset of the input that is dominated by a query point. In 3-D, dominance reporting can be solved optimally using an important combinatorial structure known as shallow cuttings [1].

Consider a set S of points in three dimensions. A *shallow cutting for the h-level of S*, or an *h-shallow cutting* for short, is a set $\mathcal{C}$ of $O(|S|/h)$ points such that any point q that dominates at most h points of S is dominated by a point p in $\mathcal{C}$; furthermore, every point of $\mathcal{C}$ dominates $O(h)$ points of S. The existence of such shallow cuttings was proven by Afshani [1], and more general shallow cuttings have been used extensively in the computational geometry literature (see e.g. [15]). To be useful in data structures, for a given point q, we also need a method that finds the point $p \in \mathcal{C}$ that dominates q. This is done using the following lemma.

Lemma 1. *(Makris and Tsakalidis [14]) Let S be a set of points in 3-D. It is possible to construct a subdivison $\mathcal{A}_S$ of the plane into $O(|S|)$ rectangles with the following property: for any query point q in 3-D, if one projects q onto the plane (i.e., the first two coordinates of q) and finds the rectangle in $\mathcal{A}_S$ that contains the projection, then one can find a point in S that dominates q or conclude that no such point exists. The point location query can be answered in $O(\log |S|)$ time using $O(|S|)$ space.*

Rectangle Stabbing. As discussed, a subproblem that we encounter is rectangle stabbing. It turns out that we need a fast solution for rectangle stabbing when given a budget of nh space. The previous results only focus on the case when h is polylogarithmic but for our purposes, we need to go far beyond polylogarithmic space. Using range trees with fan out h, we can prove the following lemma.

Lemma 2. *For d-dimensional rectangle stabbing on an input of size n, we can build a data structure that has $O(\log n \cdot (\log n/\log h)^{d-2} + t)$ query time and uses $O(nh \log^{d-2} n)$ space, in which $h \geq 2$ is an arbitrary parameter.*

Proof. Let Q be the input set of rectangles and let m be the total number of corners in the input. We divide the d-dimensional space into h regions, using $h-1$ hyperplanes perpendicular to the d-th dimension, such that each region contains roughly m/h rectangle corners. This creates $h-2$ slabs, the regions sandwiched between two consecutive hyperplanes. We say an input rectangle spans a slab b if it crosses b but it does not have any corners inside b. Let $Q_s(b)$ be the subset of Q that spans b and let $Q_c(b)$ be the subset that crosses but does not span b. Observe that we can use a $(d-1)$-dimensional rectangle stabbing data structure on $Q_s(b)$ and output those rectangles in $Q_s(b)$ that contain the query point; to find the output among $Q_c(b)$ we can simply recurse.

Thus, we build a $(d-1)$-dimensional rectangle stabbing data structure on $Q_s(b)$ for each slab and then recurse on $Q_c(b)$. We use Chazelle's data structure as a base case for $d = 2$ [8]. Let $S_k(m)$ denote the space complexity of the algorithm on a k-dimensional input with m corners. We have,

$$S_d(m) \leq hS_d(m/h) + hS_{d-1}(m)$$

which solves to $S_d(m) = O(n \log_h^{d-2})$ using with $S_2(m) = O(m)$ as the base case. If we denote the query time by $Q_d(m) + O(t)$ then the recursion for the query time is

$$Q_d(m) \leq Q_d(m/h) + \log h + Q_{d-1}(m)$$

which solves to $O_d(m) = O(\log n \cdot (\log_h n)^{d-2} n)$.

□

B.2 Simple 4-D Dominance

In this section, we present our simple solution for 4-D dominance that achieves $O(\log^{3/2} n + t)$ query time and uses $O(n \log n)$ space. We obtain the data structure by refining on the standard range tree solution which we describe below.

We use a range tree on the fourth dimension, which is a complete binary tree, T, with the input points $p_1, \dots, p_n$ stored in sorted order of their fourth coordinate in the leaves. Associate every node v in T with the points $p_i, \dots, p_j$ stored in the leaves of the subtree rooted at v. We use $S_v = \{p_i, \dots, p_j\}$ to denote the set of points associated to v. We also associate an interval I_v to v as follows. If $j = n$, then $I_v = (p_{i-1}^{(4)}; \infty)$, otherwise, $I_v = (p_{i-1}^{(4)}; p_j^{(4)}]$, in which $p_i^{(4)}$ denotes the fourth coordinate of a point p_i and $p_0^{(4)} = -\infty$.

Consider a node $u \in T$ and its left child v. We define a $Q(3,0)$ *query on node* u as a $Q(3,0)$ query on the projection of the points of S_v onto the first three dimensions. It turns out that this is the subproblem that we need to solve.

Lemma 3. *A $Q(4,0)$ query $q = (x_1, \dots, x_4)$ can be answered using a $Q(3,0)$ query on $O(\log n)$ nodes of T that lie on a root to leaf path of T. Furthermore, for every node u on the path we have $x_4 \in I_u$, and $x_4 \notin I_v$ for nodes v not on the path.*

Proof. Let $(-\infty; a)$ be the range of q in the last dimension. Start at the root r of T; let v_1 and v_2 be its left and the right child respectively. Clearly we have $x_4 \in I_r$. We have two cases.

1. a is contained in I_{v_1}: in this case, $(-\infty; a)$ does not intersect I_{v_2} so the right subtree does not contain any output points. We simply recurse on the left child.

2. a is contained in I_{v_2}: in this case, all the points associated to v_1 have their fourth coordinate smaller than a. Thus, we can consider the projection of S_{v_1} and q onto the first three dimensions and find the portion of the output that lies in S_{v_1} using a $Q(3,0)$ query, i.e., a $Q(3,0)$ query on node r. Next, we recurse on v_2.

□

One obvious solution to answer the $Q(3,0)$ queries on nodes of T is to use the data structure of Afshani [1]. But this results in $O(\log^2 n + t)$ query time for the $Q(4,0)$ problem. Until now, this has been the only way to solve 4-D dominance (with a possible increase in fanout of the range tree to $\log^\varepsilon n$). We now show that using shallow cuttings, we can do better. We have encapsulated the description of our data structure in two parts: an "outputting" data structures and "finder" data structures. The outputting data structures use finder data structures as black boxes and do the actual reporting of the output points while the finder data structures find a small number of crucial elements in the data structure.

The outputting data structures. Consider a node $u \in T$. Define P_u as the projection of S_v into the first three dimensions in which v is the left child of u. With this notation, a $Q(3,0)$ query on u is equivalent to a dominance query on P_u. Now for every $u \in T$, we build an h-shallow cutting $\mathcal{C}_u$ for P_u in which h is a parameter to be determined later; the only restriction that we place is that $h > \log^2 n$. For every point $p \in \mathcal{C}_u$, we implement an optimal dominance reporting structure [1] on the subset, $P_u(p) \subseteq P_u$ that is dominated by p. Finally, we store the $O(\log^2 n/\log\log n + t)$ query time and $O(n \lg n/\lg\lg n)$ space $Q(4,0)$ data structure of Afshani et. al [3] on the entire input set.

Generating the output of the query. Let $q = (x_1, \ldots, x_4)$ be a $Q(4,0)$ query and define $q' = (x_1, x_2, x_3)$. By Lemma 3, we can answer q by querying q' on $O(\log n)$ nodes, $u_1, \ldots, u_{O(\log n)}$, of T. Assume we have a finder structure that for each u_i can find a point $p_i \in \mathcal{C}_{u_i}$ that dominates q' or conclude that no such point exists. If for some i, no point in $\mathcal{C}_{u_i}$ dominates q', then it follows that the output size is at least $h > \log^2 n$; in this case, we simply query the structure of Afshani et. al [3] to report our output. This takes $O(t)$ time. Otherwise, to find points in P_{u_i} that are dominated by q', we query the dominance structure that stores $P_{u_i}(p_i)$; this takes $O(\log |P_{u_i}(p_i)| + t') = O(\log h + t')$ time in which t' is the output size of q' on u_i. Over all the $O(\log n)$ nodes, this takes $O(\log n \cdot \log h + t)$ time.

The finder data structures. Consider a node $u \in T$ and the h-shallow cutting $\mathcal{C}_u$ for P_u. Using Lemma 1, we decide if a point in $\mathcal{C}_u$ dominates q' and if so find it using a point location query on $\mathcal{A}_{\mathcal{C}_u}$. Unfortunately, this naive approach only yields $O(\log^2 n)$ query time. Our key idea is to perform all these point location queries *simultaneously*. We lift each rectangle $[x_1; x_2] \times [y_1; y_2]$ in $\mathcal{A}_{\mathcal{C}_u}$ to 3-D by forming the 3-D rectangle $[x_1; x_2] \times [y_1; y_2] \times I_u$. This creates $O(|P_u|/h)$ rectangles for a given node u and thus $O(n \log n/h)$ rectangles in total. We collect all the rectangles (over all the nodes in T) and store them in a 3-D rectangle stabbing data structure given by Lemma 2, with the space usage set to $O(n \log n)$.

The finder query. Let $q = (x_1, \ldots, x_4)$ be the query and define $q' = (x_1, x_2, x_3)$ and $q'' = (x_1, x_2, x_4)$. We claim that to obtain the result of a point location query for (x_1, x_2) on all the sets $\mathcal{A}_{\mathcal{C}_u}$, where we are to perform a $Q(3, 0)$ query on u, it suffices to query the stabbing data structure with q'': By Lemma 3, we have $x_4 \in I_u$ for nodes u on the path where we are to query the $Q(3, 0)$ structures and $x_4 \notin I_v$ for nodes v not on the path. By Lemma 1, we need to find the rectangle r that contains the point (x_1, x_2) which is equivalent to finding the 3-D rectangle $r \times I_u$ that contains q''. Thus, the point location query for u can be answered by looking at the result of the stabbing query.

Query time analysis. First, observe that for two nodes u_1 and u_2 at the same depth of T, the intervals I_{u_1} and I_{u_2} are disjoint. Thus, the output size of the stabbing query is $O(\log n)$. Since the ratio between the input size and space usage is $\Omega(h)$, by Lemma 2, the query time is $O(\log n \cdot \log_h n + \log n) = O(\log n \cdot \log_h n)$. We pick $h = 2^{\sqrt{\log n}}$ and obtain the following.

Theorem 1. *The 4-D dominance problem can be solved with $O(n \log n)$ space and $O(\log^{3/2} n + t)$ query time.*

B.3 Higher Dimensions

In this section we show how to extend the simple 4-D dominance result to d-dimensional dominance reporting. Before we begin, we note that similar to the 4-D case, the most interesting case of the problem is when the output size is small. For example, if we realize that the output size is larger than say $\log^{d-1} n$, we can switch to the best previous result and the query time will be optimal.

As with the 4-D case, we start with range trees. However, to get the best performance, we use range trees with large fan outs. Let $\alpha = \log^{\varepsilon} n$ in which ε is a sufficiently small constant to be determined later. We build a range tree T_d on the d-th dimension with fan out α (i.e., all nodes except leaves have α children). For a node $v_d \in T_d$, the set S_{v_d} and the interval I_{v_d} is defined as in our simple 4-D dominance solution, using the value of the d-th coordinate of the points. We also store a pointer from v_d to a range tree T_{d-1,v_d} with fan out α, built on S_{v_d} and on the $(d-1)$-th dimension. We continue this recursive construction until we reach dimension three. No range tree is built on the first three dimensions. At the end, we have a nested hierarchy of range trees, $T_{i,v_d,v_{d-1},\ldots,v_{i+1}}$, for $4 \le i \le d-1$, in which v_{i+1} is a node in the range tree $T_{i+1,v_d,v_{d-1},\ldots,v_{i+2}}$. A node $v_i \in T_{i,v_d,v_{d-1},\ldots,v_{i+1}}$ is associated with a point set S_{v_i} and consists of all the points whose j-th coordinate, $i \le j \le d$, is inside the interval I_{v_j}. To every node v_4 in a range tree $T_{4,v_d,\ldots,v_5}$, we associate a subproblem $Q_{3,v_d,\ldots,v_4}$ on S_{v_4}. Note that S_{v_4} is the set of input points that lie in the region $R(Q) = (-\infty;\infty) \times (-\infty;\infty) \times (-\infty;\infty) \times I_4 \times \cdots \times I_{v_d}$. Before describing this subproblem, we need to examine how the configuration of the range trees is traversed at query time.

Query traversal. Let $q = (x_1, \ldots, x_d)$ be the point representing a $Q(d, 0)$ query. We define the *query traversal* to be the set of subproblems that are reached using the following procedure. Start from the root r of the range tree T_d. Let v be the child of r such that $x_d \in I_v$. We follow the link to the range tree $T_{d-1,v}$, traverse it and then recurse on v; the recursion stops at the subproblems. To be more precise, assume a node $v_i \in T_{i,v_d,v_{d-1},\ldots,v_{i+1}}$ is reached in this traversal. We follow two pointers for v_i: (i) If $i = 4$, then $Q_{3,v_d,\ldots,v_4}$ is added to the query traversal; otherwise, we follow the pointer to the root of the range tree $T_{i-1,v_d,v_{d-1},\ldots,v_i}$ and traverse it and (ii) we find a child u of v_i such that $x_i \in I_u$ and then recurse on u.

The subproblem. Consider a $Q := Q_{3,v_d,\ldots,v_4}$ subproblem that is in the query traversal for the query q. Observe that the crucial property of the query traversal is that if Q is in the query traversal of q, then $q \in R(Q)$. The children of v_i decompose I_{v_i} into α smaller intervals which in geometric terms corresponds to cutting the i-th dimension of $R(Q)$ into α smaller parts. Based on this division, for every point $p = (p_1, \cdots, p_d) \in R(Q)$, we associate a *child-coordinate point* $f_Q(p)$, $f_Q(p) \subset [\alpha]^{d-3}$, to p in which the i-th coordinate is the index of the child of v_i whose interval contains p_i. Now, the subproblem Q is defined as the problem of outputting all the points $p \in S_{v_4}$ such that the point (x_1, x_2, x_3) dominates the projection of p into the first three dimensions and every coordinate of $f_Q(q)$ is larger than that of $f_Q(p)$ (i.e., $f_Q(q)$ dominates $f_Q(p)$).

Before describing our data structures, we need an equivalent of Lemma 3.

Lemma 4. *A $Q(d, 0)$ query q can be solved by solving all the subproblems that are reached during the query traversal of q. For each such subproblem Q, we have $q \in R(Q)$. Furthermore, the number of subproblems Q such that $q \in R(Q)$ is $O(\log_\alpha^{d-3} n)$.*

Proof. Consider a point $p = (p_1, \ldots, p_d) \in S$ that is dominated by q. We claim that there exists a subproblem Q in the query traversal of q that outputs p. To see this, we begin from the root r of T_d and essentially follow the procedure that builds the query traversal. Assume we have reached a node $v_i \in T_{i,v_d,v_{d-1},\ldots,v_{i+1}}$. In contrast to the procedure that builds the query traversal, we follow a single pointer at each step: (i) if there is a child u of v_i such that p_i and $x_i \in I_u$ then we follow the pointer to u (ii) otherwise, if $i > 4$, we follow the pointer to $T_{i-1,v_d,v_{d-1},\ldots,v_i}$ but if $i = 4$, then $Q := Q_{3,v_d,\ldots,v_4}$ is the desired subproblem. It is easily checked that $q \in R(Q)$ and that every coordinate of $f_Q(q)$ is greater than that of $f_Q(p)$.

It remains to prove that the number of subproblems Q, s.t. $q \in R(Q)$ is $O(\log_\alpha^{d-3})$. In a range tree with fanout α, there are only $O(\log_\alpha n)$ nodes v such that I_v contains a given point. Since we have build $d - 3$ levels of range trees, our claim can be proven by a simple inductive argument.

□

Now we are ready to describe our data structures.

The outputting data structures. Consider a $Q := Q_{3,v_d,\ldots,v_4}$ subproblem. Let P be the projection of S_{v_4} onto the first three dimensions. We partition P into $\beta := \alpha^{d-3}$ subsets as follows. For every element $\gamma \in [\alpha]^{d-3}$, we place all the points $p \in P$ with $f_Q(p) = \gamma$ in the set P_γ. We also define the set $\hat{P}_\gamma$ to be the union of all the sets $P_{\gamma'}$, $\gamma' \in [\alpha]^{d-3}$, in which γ dominates γ'. We build three categories of shallow cuttings: First, for every P_γ, we build a $\log^d n$-shallow cutting $\mathcal{C}_{d,\gamma}$ and then store the points dominated by every point $p \in \mathcal{C}_{d,\gamma}$ in an optimal 3-D dominance reporting structure. Second, for every $\hat{P}_\gamma$, we build an h-shallow cutting $\hat{\mathcal{C}}_{h,\gamma}$; the value of h will be determined later and the only restriction that we impose is that $h > \log^{2d} n$. Finally, for every point $p \in \hat{\mathcal{C}}_{h,\gamma}$, we build a $\log^{d-1} n$-shallow cutting $\hat{\mathcal{C}}_{d-1,\gamma,p}$, as well as the corresponding point location data structure of Lemma 1, on the subset of $\hat{P}_\gamma$ dominated by p. Consider a point $\gamma' \in [\alpha]^{d-3}$ that is dominated by γ. We know $P_{\gamma'} \subset \hat{P}_\gamma$ and thus every point $\hat{p} \in \hat{\mathcal{C}}_{d-1,\gamma,p}$ contains $O(\log^{d-1} n)$ points in $P_{\gamma'}$ which implies there exists a point $p \in \mathcal{C}_{d,\gamma'}$ such that p dominates $\hat{p}$. We place a pointer from $\hat{p}$ to p. Since the sets P_γ partition P, the first category of shallow cuttings take up linear space. The number of cells in the second category is $O(\alpha^{d-3}|S_{v_4}|/h)$ which is sublinear by our assumptions on h and α. Finally, the number of points in $\hat{\mathcal{C}}_{h,\gamma}$ is $O(|S_{v_4}|/h)$ and for each we store a shallow cutting that has $O(h/\log^{d-1} n)$ size and for each point in the second shallow cutting we store $O(\alpha^{d-3})$ additional pointers; thus, the total space is $O(|S_{v_4}|)$ for the subproblem Q. Over all the subproblems this sums up to $O(n \log_\alpha^{d-3} n)$.

Answering output queries. Let $q = (x_1, \ldots, x_d)$ be a $Q(d, 0)$ query and let $q' = (x_1, x_2, x_3)$. By Lemma 4, we need to answer q on $O(\log_\alpha^{d-3} n)$ subproblems. Consider one such subproblem $Q := Q_{3,v_d,\ldots,v_4}$. If we let $\gamma = f_Q(q)$ the subproblem translates to outputting the subset of $\hat{P}_\gamma$ that is dominated by q'. If no point in $\hat{\mathcal{C}}_{h,\gamma}$ dominates q', then the output size is larger than $h > \log^{d-1} n$, which means we can switch to the previous best result on dominance reporting. Assume we have a finder data structures that can find a point $p \in \hat{\mathcal{C}}_{h,\gamma}$ that dominates q' (or tell us no such point exists). Using the point location data structure implemented for $\hat{\mathcal{C}}_{d-1,\gamma,p}$ we can check if there exists a point $p' \in \hat{\mathcal{C}}_{d-1,\gamma,p}$ that dominates q'. If no such point exists, then the output size is larger than $\log^{d-1} n$ and we are done. Thus, assume we locate such a point p'. Note that finding p' takes $O(\log |\hat{\mathcal{C}}_{d-1,\gamma,p}|) = O(\log h)$ time by Lemma 1. Next, for every $\gamma' \in [\alpha]^{d-3}$, that is dominated by γ, we follow the pointer from p' to the corresponding point $p'' \in \mathcal{C}_{d,\gamma'}$ that dominates p'. As p'' also dominates q', the query can answered by querying the 3-D dominance data structure built for p''. The query time for Q becomes $O(\log h + \alpha^{d-3} \log\log n)$ which results in the overall query time of $O((\log h + \alpha^{d-3} \log\log n) \log_\alpha^{d-3} n)$.

The finder data structures. Consider a subproblem $Q := Q_{3,v_d,\ldots,v_4}$ and the shallow cutting $\hat{\mathcal{C}}_{h,\gamma}$ for some $\gamma \in [\alpha]^{d-3}$. We build the corresponding orthogonal planar subdivision $\hat{\mathcal{A}}_{h,\gamma}$ and lift a rectangle $r \in \hat{\mathcal{A}}_{h,\gamma}$ to a $d-1$-dimensional rectangle $L_Q(r) = r \times I_{v_4} \times \cdots \times I_{v_d}$. The total number of rectangles created from all the subproblems is $O(n \log_\alpha^{d-3} n/h)$ and we place them all in a $(d-1)$-dimensional rectangle stabbing structure given by Lemma 2 in which the space usage is set to $O(n)$.

The finder query. Let $q = (x_1, \ldots, x_d)$ be a query and define $q' = (x_1, x_2, x_4, \ldots, x_d)$. We claim the finder queries for all the subproblems can be answered by looking at the result of q' on the rectangle stabbing data structure. Consider a subproblem $Q := Q_{3,v_d,\ldots,v_4}$ that is reached during the query traversal of q. By Lemma 4, $q \in R(Q)$. By Lemma 1, to answer the finder query for Q, it suffices to do a point location query on $\hat{\mathcal{A}}_{h,\gamma}$ using (x_1, x_2) as the query point. Observe that if a rectangle $r \in \hat{\mathcal{A}}_{h,\gamma}$ contains (x_1, x_2), then $L_Q(r)$ contains q'. To bound the running time, notice that the number of different values of γ is α^{d-3}, and the output size of the stabbing query is at most $O(\alpha^{d-3} \log_\alpha^{d-3} n) = O(\log^{d-3} n)$ by Lemma 4. The input size of the stabbing data structure is $O((n \log_\alpha^{d-3} n/h))$ while its space usage is set to n. The ratio of the space usage to the input size is thus $\Omega(h/\log_\alpha^{d-3} n) = \Omega(\sqrt{h})$ which means by Lemma 2, the query time is $O(\log n (\log_h n)^{d-3} + \log^{d-3} n)$ (remember, we have a $d-1$-dimensional stabbing query).

Optimizing the running time. We get that the total query time is $O(\log n (\log_h n)^{d-3} + \log^{d-3} n + (\log h + \alpha^{d-3} \log\log n) \log_\alpha^{d-3} n)$. By setting $\alpha = \log^\varepsilon n$ for a small enough ε, and picking h such that $\log h = \log^{1/(d-2)} n\, (\log\log n)^{(d-3)/(d-2)}$ we get the following:

Theorem 2. *The d dimensional dominance reporting problem can be solved using $O(n(\log n/\log\log n)^{d-3})$ space and with query time of $O(\log n(\log n/\log\log n)^{d-4+1/(d-2)} + t)$.*

In the next subsection, we show that with some modifications, our data structure can in fact answer $Q(d, d-3)$ queries which in turn is used to obtain our general d-dimensional orthogonal range reporting data structure.

B.4 General orthogonal range reporting queries

The main result of this subsection is the following theorem.

Theorem 3. *The $Q(d, d-3)$ problem can be solved using $O(n(\log n/\log\log n)^{d-3})$ space and with query time of $O(\log n(\log n/\log\log n)^{d-4+1/(d-2)} + t)$.*

The data structure that we build is very similar to one used for Theorem 2. We begin by building the exact same set of range trees. The definition of the subproblem $Q_{3,v_d,\dots,v_4}$ will be different. However, as with the dominance structure, it is more convenient to define the query traversal first.

Query traversal. Without loss of generality, we can assume the $Q(d, d-3)$ query interested is $q = (-\infty; y_1) \times (-\infty; y_2) \times (-\infty; y_3) \times (x_4; y_4) \times \cdots \times (x_d; y_d)$. We now define the query traversal. As before, the traversal starts at the root of T_d. Assume a node $v_i \in T_{i+1,v_d,v_{d-1},\dots,v_{i+1}}, i \geq 4$, is reached in this traversal. We maintain the invariant that at least one of x_i or y_i is contained in I_{v_i}. Let $u_1, \cdots, u_\alpha$ be the children of v_i. We consider four cases:

(i) $x_i \in I_{v_i}, y_i \in I_{v_i}$ and x_i and y_i are in I_{u_j} for some j: In this case, we follow the pointer to u_j.

(ii) $x_i \in I_{v_i}, y_i \in I_{v_i}$ but $x_i \in I_{u_j}$ and $y_i \in u_\ell$ for $j < \ell$: In this case, we follow three pointers, first to u_j, second to u_ℓ, and third to the root of the range tree $T_{i,v_d,\dots,v_i}$ if $i > 4$ but if $i = 4$, we add $Q_{3,v_d,\dots,v_4}$ to the query traversal.

(iii) $x_i \in I_{v_i}, y_i \notin I_{v_i}$: In this case, x_i will be contained in I_{u_j} for some j. We follow two pointers, one to u_j and another to $T_{i,v_d,\dots,v_i}$ if $i > 4$ but if $i = 4$, we add $Q_{3,v_d,\dots,v_4}$ to the query traversal.

(iv) $x_i \notin I_{v_i}, y_i \in I_{v_i}$: Similar to the previous case, for an index j, I_{u_j} will contains y_i. We follow two pointers, one to u_j and another to $T_{i,v_d,\dots,v_i}$ if $i > 4$ but if $i = 4$, we add $Q_{3,v_d,\dots,v_4}$ to the query traversal.

The subproblem. Consider a $Q := Q_{3,v_d,\dots,v_4}$ subproblem that is in the query traversal for the query q. As before, the region $R(Q)$ is defined as $R(Q) = (-\infty, \infty) \times (-\infty, \infty) \times (-\infty, \infty) \times I_4 \times \cdots \times I_{v_d}$ and it contains all the points in S_{v_4}. The crucial property here is that $R(Q)$ contains at least one corner of q. For a point $p \in R(Q)$, the child-coordinate point $f_Q(p)$ is defined in the exact same way. The query in the subproblem Q is defined using a $(d-3)$-dimensional rectangle and is denoted by $\tau_Q(q)$, in which the coordinates are either integers from $[\alpha]$, $-\infty$, or ∞. Let $\tau_Q(q) = (a_4, b_4) \times \cdots \times (a_d, b_d)$. For an index i, $4 \leq i \leq d$, the values a_i and b_i are defined below, depending on which condition of the query traversal was true during the query traversal at node v_i. Note that for v_i only cases, (ii), (iii), and (iv) could be valid as case (i) does not result in a traversal of a lower dimensional range tree. Consider the notation used in the case analysis of the query traversal.

- If case (ii) was true at v_i, then $a_i = j$ and $b_i = \ell$.
- If case (iii) was true at v_i, then $a_i = j$ and $b_i = \infty$.
- If case (iv) was true at v_i, then $a_i = -\infty$ and $b_i = j$.

With this definition, the subproblem Q is defined as outputting all the points $p \in S_{v_4}$ such that the point (y_1, y_2, y_3) dominates the projection of p into the first three dimensions and $f_Q(p)$ is contained in the rectangle $\tau_Q(q)$.

Before describing our data structures, we need an equivalent of Lemma 4.

Lemma 9. *A $Q(d,d-3)$ query q can be answered by solving all the subproblems that are reached during the query traversal of q. For each such subproblem Q, $R(Q)$ contains at least one corner of q. Furthermore, the number of subproblems Q that contains a corner of q is $O(\log_\alpha^{d-3} n)$.*

Proof. Let $q = (-\infty; y_1) \times (-\infty; y_2) \times (-\infty; y_3) \times (x_4; y_4) \times \cdots \times (x_d; y_d)$ be the query rectangle. Consider a point $p = (p_1, \ldots, p_d) \in S$ that is contained in q. We claim that there exists a subproblem Q in the query traversal of q that outputs p. To find Q, we begin from the root r of T_d and follow the procedure that builds the query traversal, and trace the footsteps of the query traversal, except that we follow exactly one pointer at each case. Assume we have reached a node $v_i \in T_{i,v_d,v_{d-1},\ldots,v_{i+1}}$. We maintain the invariant that at least one of x_i or y_i is contained in I_{v_i}.

We review the four cases that were considered for the query traversal.

(i) $x_i \in I_{v_i}, y_i \in I_{v_i}$ and x_i and y_i are in I_{u_j} for some j: In this case, we have $p_i \in I_{u_j}$ and we follow the pointer to u_j.

(ii) $x_i \in I_{v_i}, y_i \in I_{v_i}$ but $x_i \in I_{u_j}$ and $y_i \in u_\ell$ for $j < \ell$: we follow one the three pointers followed at the query traversal. If $p_i \in I_{u_j}$ we follow the pointer to u_j, if $p_i \in I_{u_\ell}$ we follow the pointer to u_ℓ, but otherwise, we follow the pointer to the root of the range tree $T_{i,v_d,\ldots,v_i}$ if $i > 4$ but if $i = 4$, then $Q_{3,v_d,\ldots,v_4}$ is the desired subproblem.

(iii) $x_i \in I_{v_i}, y_i \notin I_{v_i}$: In this case, x_i will be contained in I_{u_j} for some j. If $p_i \in I_{u_j}$, then we follow the pointer to u_j, otherwise, we follow the pointer to $T_{i,v_d,\ldots,v_i}$ if $i > 4$ but if $i = 4$, then $Q_{3,v_d,\ldots,v_4}$ is the desired subproblem.

(iv) $x_i \notin I_{v_i}, y_i \in I_{v_i}$: Similar to the previous case, for an index j, I_{u_j} will contains y_i. If $p_i \in I_{u_j}$, then we follow the pointer to u_j, otherwise, we follow the pointer to $T_{i,v_d,\ldots,v_i}$ if $i > 4$ but if $i = 4$, then $Q_{3,v_d,\ldots,v_4}$ is the desired subproblem.

It is straightforward to verify that when a subproblem Q is reached, $R(Q)$ contains a corner of q and the rectangle $\tau_Q(q)$ contains the point $f_Q(p)$. Thus, p will be outputted by solving Q.

Rectangle q has a constant number of corners and thus the number of subproblems Q such that $R(Q)$ contains a corner of q is $O(\log_\alpha^{d-3})$.

□

We now describe our data structures.

The outputting data structures. Consider a $Q := Q_{3,v_d,\ldots,v_4}$ subproblem. Let P be the projection of S_{v_4} onto the first three dimensions. Like the $Q(d,0)$ case, we partition P into α^{d-3} subsets. For every element $\gamma \in [\alpha]^{d-3}$, we place all the points $p \in P$ with $f_Q(p) = \gamma$ in the set P_γ. Observe that the number possible rectangles $\tau_Q(q)$ that can be queried on this subproblem is less than α^{2d}. For every such $\tau = \tau_Q(q)$, define the set $\hat{P}_\tau$ to be the union of all the sets P_γ in which γ is in the interior of τ. With these definitions, the rest of our outputting data structures is almost identical to the $Q(d,0)$ case: we build three categories of shallow cuttings: First, for every P_γ, we build a $\log^d n$-shallow cutting $\mathcal{C}_{d,\gamma}$ and then store the points dominated by every point $p \in \mathcal{C}_{d,\gamma}$ in an optimal 3-D dominance reporting structure. Second, for every $\hat{P}_\tau$, we build an h-shallow cutting $\hat{\mathcal{C}}_{h,\tau}$; the value of h will be determined later and the only restriction that we impose is that $h > \log^{2d} n$. Finally, for every point $p \in \hat{\mathcal{C}}_{h,\tau}$, we build a $\log^{d-1} n$-shallow cutting $\hat{\mathcal{C}}_{d-1,\tau,p}$, as well as the corresponding point location data structure, on the subset of $\hat{P}_\tau$ dominated by p. As before, every point $\hat{p} \in \hat{\mathcal{C}}_{d-1,\tau,p}$ is dominated by a point $p \in \mathcal{C}_{d,\gamma'}$ and we place a pointer from $\hat{p}$ to p. It is easy to see that the total space consumption is linear for Q and thus $O(n \log_\alpha^{d-3} n)$ over all the subproblems.

Answering output queries. Let $q = (-\infty; y_1) \times (-\infty; y_2) \times (-\infty; y_3) \times (x_4; y_4) \times \cdots \times (x_d; y_d)$ be the $Q(d, d-3)$ query and let $q' = (y_1, y_2, y_3)$. By Lemma 9, we need to answer q on $O(\log_\alpha^{d-3} n)$ subproblems. Consider one such subproblem $Q = Q_{3,v_d,\ldots,v_4}$. If we let $\tau = \tau_Q(q)$, then the subproblem translates to outputting the subset of $\hat{P}_\tau$ that is dominated by q'. If no point in $\hat{\mathcal{C}}_{h,\tau}$ dominates q', then the output size is larger than $h > \log^{d-1} n$, and we are done by switching to the best previous orthogonal reporting data structure. Assume we have a finder data structures that can find a point $p \in \hat{\mathcal{C}}_{h,\tau}$ that dominates q' (or tell us no such point exists). Using the point location data structure implemented for $\hat{\mathcal{C}}_{d-1,\tau,p}$, in $O(\log h)$ time we can check if there exists a point $p' \in \hat{\mathcal{C}}_{d-1,\tau,p}$ that dominates q'. If no such point exists, then the output size is larger than $\log^{d-1} n$ and we are done. Thus, assume we locate such a point p'. Next, for every $\gamma' \in [\alpha]^{d-3}$, s.t., γ' is contained in τ, we follow the pointer from p' to the corresponding point $p'' \in \mathcal{C}_{d,\gamma'}$ that dominates p'. As p'' also dominates q', the query can answered by querying the 3-D dominance data structure built for p''. The query time for Q becomes $O(\log h + \alpha^{d-3} \log\log n)$ which results in the overall query time of $O((\log h + \alpha^{d-3} \log\log n) \log_\alpha^{d-3} n)$.

The finder data structures. Consider a subproblem $Q := Q_{3,v_d,\ldots,v_4}$ and the shallow cutting $\hat{\mathcal{C}}_{h,\tau}$ for all the $O(\alpha^{2d})$ choices of τ. We build the corresponding orthogonal planar subdivision $\hat{\mathcal{A}}_{h,\tau}$ and lift a rectangle $r \in \hat{\mathcal{A}}_{h,\gamma}$ to a $d-1$-dimensional rectangle $L_Q(r) = r \times I_{v_4} \times \cdots \times I_{v_d}$. The total number of rectangles created from all the subproblems is $O(n \log_\alpha^{d-3} n/h)$ and we place them all in a $(d-1)$-dimensional rectangle stabbing structure given by Lemma 2 in which the space usage is set to $O(n \log^{d-3} n)$.

The finder query. Let $q = (-\infty; y_1) \times (-\infty; y_2) \times (-\infty; y_3) \times (x_4; y_4) \times \cdots \times (x_d; y_d)$ be the $Q(d, d-3)$ query, and let $q' = (y_1, y_2, y_3)$. We query the stabbing data structure using every corner of q, after removing the third coordinate. We claim the results of these queries is sufficient to answer all the finder queries.

Consider a subproblem Q that is in the query traversal of q. By Lemma 9, $R(Q)$ contains a corner $\delta = (y_1, y_2, y_3, \delta_4, \cdots, \delta_d)$ of q in which δ_i for $4 \le i \le d$ is either x_i or y_i. As discussed, the stabbing data structure will be queried with point $\delta' = (y_1, y_2, \delta_4, \cdots, \delta_d)$. Let $\tau = \tau_Q(q)$ and consider the shallow cutting $\hat{\mathcal{C}}_{h,\tau}$ and the corresponding planar subdivision $\hat{\mathcal{A}}_{h,\tau}$. By Lemma 1, it suffices to find a rectangle $r \in \mathcal{A}_{h,\tau}$ that contains the point (y_1, y_2). Observe that since $\delta \in R(Q)$, we have $\delta_i \in I_{v_i}, 4 \le i \le d$. Thus, if r contains (y_1, y_2), then it follows that $L_Q(r)$ contains the point δ'.

As before, the output size of the stabbing query is at most $O(\log^{d-3} n)$ by Lemma 9 and by Lemma 2, the query time is $O(\log n (\log_h n)^{d-3} + \log^{d-3} n)$. Using the same value of h as before, we get the following theorem.

Theorem 5. *$Q(d, d-3)$ queries can be answered in of $O(\log n(\log n/\log\log n)^{d-4+1/(d-2)} + t)$ time using a data structure that consumes $O(n(\log n/\log\log n)^{d-3})$ space.*

Using an standard application of range trees with fan out $\alpha = \log^\varepsilon n$ we get the following theorem.

Theorem 6. *Orthogonal range reporting can be solved using $O(n(\log n/\log\log n)^d)$ space and with query time of $O(\log n(\log n/\log\log n)^{d-4+1/(d-2)} + t)$.*

C A Tight Lower Bound for Rectangle Stabbing

In this section we prove our lower bound for the rectangle stabbing problem. Our model of computation is identical to one used by Chazelle [9] but for sake of completeness, we include a brief description.

Consider an input set S of n elements. A data structure is a collection of (memory) cells and its space complexity is the number of cells used. Each cell can store an input element or an auxiliary data type. Furthermore, each cell can have two pointers to two other cells. Such a data structure can be represented by a graph G with $V(G)$ being the set of cells used in the data structure. A pointer from a cell u to a cell v is represented with a directed edge from u to v. The crucial restriction is that a cell can only be accessed through pointers and thus random accesses are disallowed. (It is assumed that we always begin by a pointer to a root cell.) A query q is answered by exploring (a subgraph of) G. In this model, to output an element $p \in S$, the query algorithm must visit a cell that stores p. We do not impose any restriction on how the algorithm navigates G to reach such a node. We measure the query time by the number of vertices of G visited by the query algorithm.

Consider a data structure D that answers rectangle stabbing queries in d dimensions in the above model. Let nh be the number of cells used by D. Furthermore, assume D answers every query in $f(n) + Ct$ time in which t is the output size, C is a constant, and $f(n)$ is the search time of the query. Our goal is to prove that $f(n) = \Omega(\log n \cdot \log_h^{d-2} n)$.

We build an input set, Q, of rectangles that is in fact a simplification of the ones used before [3, 9, 10]. Consider a d-dimensional cube $\mathcal{R}$ with side length m, $m > \sqrt{n}$, in which m is a parameter to be determined later. Let $r = c_r h^4$, in which c_r is a large enough constant, and let $\mathrm{Log}_r m = \lfloor \log_r m \rfloor - 1$. For every choice of $d-1$ indices, $1 \le i_1, \cdots, i_{d-1} \le \mathrm{Log}_r m$, we divide the j-th dimension of $\mathcal{R}$ into r^{i_j} equal pieces for $1 \le j \le d-1$, and the d-th dimension of $\mathcal{R}$ into $\lfloor m^{d-1}/r^{i_1+\cdots+i_{d-1}} \rfloor$ equal pieces. The number of such choices is $k = \mathrm{Log}_r^{d-1} m$. With each such choice, we create between $m/2$ and m rectangles and thus $\Theta(mk)$ rectangles are created in total. We pick m such that this number equals n, thus $n = \Theta(mk)$. Also, note that the volume of each rectangle is between m^{d-1} and $2m^{d-1}$. The crucial property of this input set is the following.

Observation 1. *The volume of the intersection of any two rectangles in $\mathcal{R}$ is at most $2m^{d-1}/r$.*

Unlike the previous attempts, our query set is very simple: a single query q chosen uniformly at random inside $\mathcal{R}$. Note that the output size of q is k. The rest of this section is devoted to prove that with positive probability answering q needs $\Omega(k \log r) = \Omega(\log_r^{d-1} m \log r) = \Omega(\log n \cdot \log_h^{d-2} n)$ time which proves our claim.

Let c be a parameter to be determined later. Call a rectangle b *expensive*, if there are more than ch cells in D that store b.

Observation 2. *There are at most n/c expensive rectangles.*

For a vertex $u \in G$, let $\mathrm{In}(u)$ denote the set of vertices in G that have a directed path of size at most $\log h$ to u. Similarly, define $\mathrm{Out}(u)$ to be the set vertices in G that can be reached from u using a path of size at most $\log h$. Call a cell v *popular* if $\mathrm{In}(v) > ch^2$. Similarly, we say a rectangle b is popular if there is a popular cell that stores b.

Lemma 5. *There are at most n/c popular rectangles.*

Proof. As each vertex in G has two out edges, the size of $\mathrm{Out}(u)$ is at most $2^{\log h} = h$. Also, if $v \in \mathrm{Out}(u)$, then $u \in \mathrm{In}(v)$ and vice versa. Thus,

$$\sum_{v \in G} |\mathrm{In}(v)| = \sum_{u \in G} |\mathrm{Out}(u)| \le nh^2.$$

This implies the number of vertices v with $\mathrm{In}(v) > ch^2$ is at most n/c. Our claim follows since the number of popular rectangles cannot exceed the number of popular cells. □

Lemma 6. *With probability at least $3/5$, q is enclosed by at most $k/4$ rectangles that are either expensive or popular, if c is to be a chosen sufficiently large constant.*

Proof. By Observation 2 and Lemma 5, the number of expensive or popular rectangles is at most $2n/c$. As each rectangle has volume at most $2m^{d-1}$, the total volume of expensive or popular rectangles is at most $4m^{d-1}n/c$. Let A be the region of $\mathcal{R}$ that contains all the points covered by more than $k/4$ expensive or popular rectangles. We have,

$$\mathrm{Vol}(A)k/4 \leq 4m^{d-1}n/c = \Theta(m^d k/c)$$

which implies $\mathrm{Vol}(A) = O(m^d/c) < 2m^d/5$ if c is large enough. Thus, with probability at least $3/5$, q will not be picked in A. □

Let S' be the subset of rectangles that are neither expensive nor popular and let $n' = |S'|$. We say two rectangles $b_1, b_2 \in S'$ are *close*, if there exists a vertex $u \in G$ such that from u we can reach a cell that stores b_1 and a cell that stores b_2 with directed paths of size at most $\log h$ each.

Lemma 7. *A rectangle $b \in S'$ can be close to at most c^2h^4 other rectangles in S'.*

Proof. A rectangle $b \in S'$ is stored in at most ch vertices of G and for each such vertex v we know $\mathrm{In}(v) \leq ch^2$. Since for every $u \in G$ we have $\mathrm{Out}(u) \leq h$, it follows that v can be close to at most ch^3 vertices in G. Multiplying this number by ch gives the maximum number of rectangles that can be close to b. □

Consider a rectangle $b \in S'$. Let B be the subset of rectangles that are close to b. We call $\cup_{b' \in B}(b \cap b')$ the *close region* of b and denote it with C_b.

Lemma 10. *With probability at least 1/5, answering q needs $\Omega(\log n \cdot \log_h^{d-2} n)$ time.*

Proof. Consider a rectangle $b \in S'$ and set B defined above. By Lemma 7, $|B| \leq c^2h^4$. By Observation 1, for every $b' \in B$, $\mathrm{Vol}(b \cap b') \leq 2m^{d-1}/r$ and thus

$$\mathrm{Vol}(C_b) \leq \sum_{b' \in B} \mathrm{Vol}(b \cap b') \leq 2c^2h^4m^{d-1}/r = 2c^2m^{d-1}/c_r.$$

As there are at most n rectangles in S', the sum of the volumes of the close regions of all the rectangles in S' is at most $n \cdot 2c^2m^{d-1}/c_r = \Theta(c^2km^d/c_r)$. Consider the region R of all the points p such that p is inside the close region of at least $k/4$ rectangles of S'. We have,

$$\mathrm{Vol}(R)k/4 \leq \sum_{b \in S'} \mathrm{Vol}(C_b) \leq \Theta(c^2km^d/c_r)$$

which means $\mathrm{Vol}(R) = O(c^2m^d/c_r)$. If we choose c_r large enough, this volume is less than $2m^d/5$, which means with probability at least $3/5$, the selected query will not be inside R. Combined with Lemma 6, this gives the following: with probability at least $1/5$, q will be inside at least $3k/4$ rectangles that are in S' (by Lemma 6) and it will also be inside the close region of at most $k/4$ rectangles. Let q be the query when this happens.

Consider the subgraph H_q of G that is explored by the query procedure while answering q. Assume q needs to output rectangles $b_1, \cdots, b_{k'}$ from S'. We have $k' \geq 3k/4$. Let v_i be a node in

H_q that stores b_i, $1 \le i \le k'$. Also, let w_i be the set of nodes in H_q that are reachable by traversing up to $\log h$ edges from v_i, where the direction of edged have been reversed. If two sets w_i and w_j, $1 \le i < j \le k'$, share a common vertex, it means that q is inside the close region of both b_i and b_j. However, we know that q is inside the close region of at most $k/4$ rectangles. This means that there are at least $k/2$ sets w_i that do not share any vertices. Thus, the size of H_q is at least $k/2 \cdot \log h = \Omega(\log n \cdot \log_h^{d-2} n)$. $\square$

D Conclusions

We believe our most surprising result is the $O(n \log n / \log\log n)$ space structure for 4-D dominance reporting that can answer queries in $O(\log n \sqrt{\log n / \log\log n} + t)$ time. The existence of such a structure raises the obvious open question that whether this query time can be reduced to $O(\log n + t)$. Unfortunately, with our techniques this does not seem possible. Perhaps the next big development in this area needs to come from the lower bound side and it is possible that our new lower bound argument could be useful in this venue; however, it appears the biggest challenge in the lower bound side is building the right "bad" query and inputs sets.