Orthogonal Range Reporting: Query Lower Bounds, Optimal Structures in 3-d, and Higher-dimensional Improvements

Peyman Afshani[*] Lars Arge[*] Kasper Dalgaard Larsen[*]

MADALGO[†]
Department of Computer Science
Aarhus University, Denmark.

{peyman,large,larsen}@madalgo.au.dk

ABSTRACT

Orthogonal range reporting is the problem of storing a set of n points in d-dimensional space, such that the k points in an axis-orthogonal query box can be reported efficiently. While the 2-d version of the problem was completely characterized in the pointer machine model more than two decades ago, this is not the case in higher dimensions.

In this paper we provide a space optimal pointer machine data structure for 3-d orthogonal range reporting that answers queries in $O(\log n + k)$ time. Thus we settle the complexity of the problem in 3-d. We use this result to obtain improved structures in higher dimensions, namely structures with a $\log n/\log\log n$ factor increase in space and query time per dimension. Thus for $d \geq 3$ we obtain a structure that both uses optimal $O(n(\log n/\log\log n)^{d-1})$ space and answers queries in the best known query bound $O(\log n(\log n/\log\log n)^{d-3} + k)$.

Furthermore, we show that any data structure for the d-dimensional orthogonal range reporting problem in the pointer machine model of computation that uses $S(n)$ space linear space must spend $\Omega((\log n/\log(S(n)/n))^{\lfloor d/2\rfloor -1} + k)$ time to answer a query. Thus, if $S(n)/n$ is poly-logarithmic, then the query time is at least $\Omega((\log n/\log\log n)^{\lfloor d/2\rfloor -1} + k)$. This is the first known non-trivial query lower bound and it has two important implications. First, it shows that the query bound increases with dimension. Second, in combination with our upper bounds it shows that the optimal query bound increases from $\Theta(\log n+k)$ to $\Omega((\log n/\log\log n)^2+k)$ somewhere between three and six dimensions.

Finally, we show that our techniques also lead to improved structures for point location in rectilinear subdivisions, that is, the problem of storing a set of n disjoint d-dimensional axis-orthogonal rectangles, such that the rectangle containing a query point q can be found efficiently.

Categories and Subject Descriptors

E.1 [**Data Structures**]: Miscellaneous

General Terms

Algorithms, Theory

Keywords

data structures, orthogonal range reporting, pointer machine, lower bounds

1. INTRODUCTION

Orthogonal range reporting is the problem of storing a set of n points in d-dimensional space, such that the k points in an axis-orthogonal query box can be reported efficiently. This is a fundamental problem in several fields, including spatial databases and computational geometry, and it has been studied extensively [3, 4]. We study the problem in the pointer machine model of computation [16]. While the two-dimensional version of the problem was completely characterized in this model more than two decades ago, this is not the case in higher dimensions.

In this paper we consider orthogonal range reporting data structures with poly-logarithmic query time in the pointer machine model. We settle the complexity of the problem in three dimensions by providing an optimal structure, and show how this structure can be used to obtain improved structures in higher dimensions. We also prove that a query time of $o((\log n/\log\log n)^{\lfloor d/2\rfloor -1})$ cannot be achieved with $O(n\log^{O(1)} n)$ space. This is the first non-trivial query lower bound for orthogonal range reporting structures that use at most a poly-logarithmic factor more than linear space, showing that the query time must increase with dimension.

1.1 Previous orthogonal range reporting results

In this section we review previous orthogonal range reporting results. For brevity we only review the structures most

[*]Work was supported in part by the Danish National Research Foundation and the Danish Strategic Research Council.

[†]Center for Massive Data Algorithmics—a center of the Danish National Research Foundation.

Space	Query	Ref.
$n(\log n/\log\log n)^3$	$\log n + k$	[2]
$n(\log n/\log\log n)^2$	$\log n(\log n/\log\log n) + k$	[2]
$n(\log n/\log\log n)^2$	$\log n + k$	*

Table 1: 3-d orthogonal range reporting results. * indicates this paper.

Space	Query	Ref.
$n\log^d n/(\log\log n)^3$	$\log^{d-2} n + k$	[2]
$n(\log n/\log\log n)^{d-1}$	$\log n(\log n/\log\log n)^{d-2} + k$	[2]
$n(\log n/\log\log n)^{d-1}$	$\log n(\log n/\log\log n)^{d-3} + k$	*

Table 2: d-dimensional orthogonal range reporting results. * indicates this paper.

relevant to our work, that is, static pointer machine structures with poly-logarithmic query bounds. Refer to surveys for further results [3, 4].

In two dimensions the orthogonal range reporting problem can be solved in $O(\log n + k)$ time using $O(n\log n/\log\log n)$ space [8]. This is optimal since Chazelle proved that any d-dimensional structure with an $O(\log^{O(1)} n + k)$ query bound has to use $\Omega(n(\log n/\log\log n)^{d-1})$ space [9].

However, for dimensions $d \geq 3$ no optimal $O(\log n + k)$ time and $O(n(\log n/\log\log n)^{d-1})$ space structure is known. Until recently, the known optimal $O(n(\log n/\log\log n)^{d-1})$ space structure only answered queries in $O(\log^{d-1+\varepsilon} n + k)$ time [9]. Using $O(n\log^{d-1} n/\log\log n)$ space another structure could answer queries in $O(\log^{d-1} n + k)$ time [8]. A structure with an $O(\log^{d-2} n + k)$ query bound, that is, an optimal query bound in 3-d, was also known but it uses $O(n\log^d n)$ space [7, 11]. Recently an improved optimal space structure was developed, answering queries in $O(\log n(\log n/\log\log n)^{d-2} + k)$ time, just as an improved $O(\log^{d-2} n + k)$ query time structure was developed, using $O(n\log^d n/(\log\log n)^3)$ space [2]. These two structures are the currently best known d-dimensional structures.

The lower bound of Chazelle discussed above provides a space lower bound for poly-logarithmic query structures. However, no non-trivial query lower bound is known for super-linear space structures. It is not even known if the query time increases with dimension; the existence of both 2-d and 3-d $O(\log n + k)$ query structures might suggest this is not the case. Furthermore, since the two best known structures answer 3-d queries in $O(\log n + k)$ time using $O(n(\log n/\log\log n)^3)$ space and $O(\log n \cdot (\log n/\log\log n) + k)$ time using $O(n(\log n/\log\log n)^2)$ space, respectively, one might also suspect that a $\log n/\log\log n$ factor improvement in query time (resp. space) is only possible with a corresponding increase in space (resp. query time). In fact, in a previous paper we conjectured that the space-query product of any orthogonal range reporting structure has to be $\Omega(n\log^{2d-2} n/(\log\log n)^{2d-3})$ [2].

1.2 Our orthogonal range reporting results

In Section 2 of this paper we describe the first 3-d orthogonal range reporting data structure that both uses optimal $O(n(\log n/\log\log n)^2)$ space and answers queries in optimal $O(\log n + k)$ time. Not only does this structure settle the complexity of the problem in 3-d, but also disproves the query-space product conjecture. Refer to Table 1. The structure relies on a new $O(n\log n/\log\log n)$ space and $O(\log n + k)$ query time structure for 3-d orthogonal range reporting where the query box is unbounded (in one direction) in *one* of the three dimensions. This is optimal (since the structure can be used to answer 2-d queries) and (maybe surprisingly) the bounds match the bounds of the known optimal structure for the case where the query box is unbounded in *two* of the three dimensions [2].

In Section 2 we show how our 3-d structure can be extended to higher dimensions with a $\log n/\log\log n$ factor increase in both query time and space per dimension. For $d \geq 3$, the resulting d-dimensional data structure uses optimal $O(n(\log n/\log\log n)^{d-1})$ space and answers queries in $O(\log n(\log n/\log\log n)^{d-3} + k)$ time, which improves the two best previous results. See Table 2.

In Section 3 we then show that any d-dimensional orthogonal range reporting data structure in the pointer machine model that uses $S(n)$ space and answers queries in $Q(n) + O(k)$ time must have $Q(n) = \Omega((\log n/\log(S(n)/n))^{\lfloor d/2\rfloor-1})$. Thus, if $S(n)/n$ is poly-logarithmic, then the query time is at least $\Omega((\log n/\log\log n)^{\lfloor d/2\rfloor-1} + k)$. This is the first non-trivial query lower bound and it has two important implications. First, it shows that the query bound increases with dimension. Second, in combination with our upper bounds it shows that the optimal query bound increases from $\Theta(\log n + k)$ to $\Omega((\log n/\log\log n)^2 + k)$ somewhere between three and six dimensions.

1.3 Rectilinear subdivision point location results

We also consider point location in rectilinear subdivisions, since the ideas used in our orthogonal range reporting structures can be used to develop improved structures for this problem as well. The point location in rectilinear subdivisions problem is the problem of storing a set of n disjoint d-dimensional axis-orthogonal rectangles, such that the rectangle (if any) containing a query point q can be found efficiently. The previously best known pointer machine structure for this problem uses linear space and answers queries in $O(\log^{d-1} n)$ time [12]. In Section 4, we design an improved structure for $d \geq 3$, that can answer queries in $O(\log n(\log n/\log\log n)^{d-2})$ time using linear space.

2. ORTHOGONAL RANGE REPORTING DATA STRUCTURES

In this section, we describe our new orthogonal range reporting data structures. Our structures rely on improved solutions to various special cases of the problem, where the query box can be unbounded or the points colored.

We use $Q(d,k)$ to refer to the special case in which the query boxes have finite ranges in k of the d dimensions, that is, are unbounded in $d-k$ dimensions. Refer to Fig. 1. The $Q(2,1)$ and $Q(d,0)$ problems are often called *3-sided planar range reporting* and *dominance reporting*, respectively. We then define the (more difficult) *concurrent $Q(d,k)$ problem* as follows: Let S be a set of n points in d-dimensional space, $A : S \to \mathcal{C}$ a function that assigns a color from a color set $\mathcal{C}$ to each point, and $\mathcal{P}$ a set of color sets (that is, $\mathcal{P} \subset 2^{\mathcal{C}}$). The problem is to store S such that for any query tuple (q, L) consisting of a $Q(d,k)$ query q and a set $L \in \mathcal{P}$ of

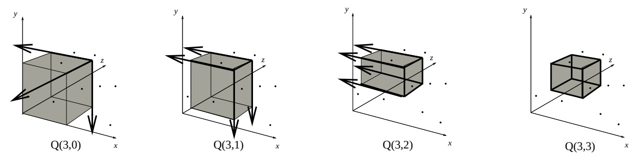

Figure 1: Three-dimensional queries.

colors, all points $p \in S$ with $p \in q$ and $A(p) \in L$ can found efficiently.

In Subsection 2.1 below we first show how to solve the concurrent $Q(3,2)$ problem efficiently. In Subsection 2.2 we then use this solution to obtain efficient solutions for the $Q(3,3)$ and $Q(d,d)$ problems, that is, for orthogonal range reporting.

2.1 Solving Concurrent $Q(3,2)$

In this section we show how to solve the concurrent $Q(3,2)$ problem efficiently. To do so we first solve the concurrent $Q(3,0)$ problem.

Throughout the section, we use the following notations: Given two points p and q, we say p *dominates* q if and only if all the coordinates of p are greater than those of q. For a set S of points we let $\mathrm{Dom}_S(q)$ denote the subset of S dominated by q. A *shallow cutting for the k-level of S*, or a *k-shallow cutting* for short, is a set R of $O(|S|/k)$ points such that any 3-d point p that dominates at most k points of S is dominated by a point in R, and such that every point of R dominates $O(k)$ points of S. The existence of this kind of shallow cuttings is known [1], and more general shallow cuttings have been used extensively in the computational geometry literature (see e.g. [15]).

The following lemma and its proof plays an important role in our concurrent $Q(3,0)$ data structure.

LEMMA 1. *(Makris and Tsakalidis [14]) Let S be a set of n points in 3-d. It is possible to construct a subdivison $\mathcal{A}_S$ of the plane of size $O(n)$ using orthogonal line segments, such that for any query point q in 3-d, one can use the result of a point location query on $\mathcal{A}_S$ to find a point that dominates q or to conclude that no such point exists.*

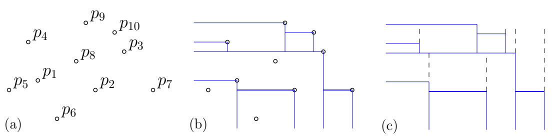

Figure 2: (a) The projection of the 3-d points sorted by the z-coordinate. (b) The arrangement $\mathcal{A}_S$. (c) The trapezoid decomposition of $\mathcal{A}_S$.

PROOF. Let $p_1, \ldots, p_n$ be the points in S projected onto the xy-plane, sorted decreasingly by their z-coordinates (Figure 2(a)). Now consider the process of shooting two rays from p_i, starting with $i = 1$, one vertical towards $y = -\infty$ and another horizontal towards $x = -\infty$, until they intersect any pre-existing ray. Next, remove all the points p_j, $j > i$, that are dominated by p_i and continue with the next point. It is easy to realise that the resulting planar arrangement, $\mathcal{A}_S$, is of size $O(n)$. Refer to Figure 2(b).

Now let q' be the projection of q onto the xy-plane and let $p \in S$ be the point at the top-right corner of the region of $\mathcal{A}_S$ containing q'. Observe that p has the largest z-coordinate among all the points that dominate q' in the xy-plane. Thus if the z-coordinate of p is greater than that of q then p dominates q. Otherwise, no point dominates q. Therefore the domination query q can easily be answered once q' is located in $\mathcal{A}_S$. □

LEMMA 2. *There exists an $O(n)$ space structure for the concurrent $Q(3,0)$ problem that answers queries in $O(|\mathcal{C}| \log\log n + \log n + k)$ time.*

PROOF. Our concurrent $Q(3,0)$ structure is constructed as follows. For every color $c \in \mathcal{C}$, we build a $(\log n)$-shallow cutting R_c for the set $S_c := \{p \in S | \mathcal{A}(p) = c\}$. For every point $v \in R_c$, we store $\mathrm{Dom}_{S_c}(v)$ in a dominance reporting data structure and also store S_c in the dominance reporting structure of [1] (that is, a $Q(3,0)$ structure). Furthermore, using Lemma 1, we build the arrangement $\mathcal{A}_c$ for the set R_c. We apply trapezoid decomposition on $\mathcal{A}_c$ (that is, we shoot vertical rays towards $y = \infty$ from every vertex; refer to Figure 2(c)) and obtain a set of $O(|R_c|)$ boxes. Then we construct a point enclosure data structure [8] on the union of the boxes obtained from $\mathcal{A}_c$ for all $c \in \mathcal{C}$, that is, a structure that can be used to find all boxes containing a given query point.

That the above structure uses linear space can be seen as follows. Each point in the shallow cutting R_c dominates $O(\log n)$ points, so $|R_c| = O(|S_c|/\log n)$. By the properties of the shallow cutting, we know that $|\mathrm{Dom}_{S_c}(v)| = O(\log n)$ for every $v \in R_c$. Since the dominance data structure of [1] uses linear space, it follows that the combined space usage of all the dominance data structures is $O(n)$. Finally, the total number of boxes in the arrangements $\mathcal{A}_c$ over all $c \in \mathcal{C}$ is $O(n/\log n)$. Thus, since the point enclosure structure of [8] also uses linear space, the space usage of the point enclosure data structure is $O(n/\log n)$.

Now consider a query tuple (q, L). With a little abuse of notation we let q denote the point defining the $Q(3,0)$ query q. For a color c, we let $k_c = |\mathrm{Dom}_{S_c}(q)|$. Clearly, $\sum_{c \in L} k_c = k$. For every $c \in L$, we want to perform a point location query with q on $\mathcal{A}_c$. Since the boxes defined from $\mathcal{A}_c$ are disjoint, we can do so in $O(\log n + |\mathcal{C}|)$ time simply by querying the point enclosure data structure with q and this way obtain a box for each of the $|\mathcal{C}|$ colors. By Lemma 1, having the point location result for $\mathcal{A}_c$ enables us to determine if (i) there is a point $v \in R_c$ that dominates q or (ii) no point of R_c dominates q. In case (i), it follows that $\mathrm{Dom}_{S_c}(q) \subseteq \mathrm{Dom}_{S_c}(v)$ and thus $\mathrm{Dom}_{S_c}(q)$ can be reported by querying the dominance reporting data structure on $\mathrm{Dom}_{S_c}(v)$; this takes $O(\log|\mathrm{Dom}_{S_c}(v)| + k_c) = O(\log\log n + k_c)$ time. In case (ii), it follows from the properties of the shallow cutting that $k_c > \log n$. In this case, we query the dominance data structure implemented on S_c using $O(\log|S_c| + k_c) = O(k_c)$ time. Summing the query time over all $c \in L$ we obtain a total query time of $O(\log n + |\mathcal{C}| + |L| \log\log n + k) = O(|\mathcal{C}| \log\log n + \log n + k)$. □

Below we show how Lemma 2 combined with a grid building technique can be used to solve the concurrent $Q(3,2)$ problem. Our strategy is inspired by the previous grid building techniques [5], in particular the work of Karpinski and Nekrich [13]. We need the following technical observation.

OBSERVATION 1. *Define $f(n) = \sqrt{nt}$, $f^{(1)}(n) = f(n)$ and $f^{(k)}(n) = f(f^{(k-1)}(n))$. Let $f^*(n)$ be the minimum integer k s.t., $f^{(k)}(n) \le t \log n$. We have $f^*(n) \le \lceil \log\log n - \log\log\log n \rceil$.*

PROOF. It is easy to check that $f^{(i)}(n) = t^{(2^i-1)/2^i} n^{1/2^i} \le tn^{1/2^i}$. Thus, $f^{(\lceil \log\log n - \log\log\log n \rceil)} \le tn^{\log\log n/\log n} = t\log n$. □

We are now ready to describe our concurrent $Q(3,2)$ structure, assuming without loss of generality that the query boxes are unbounded towards $-\infty$ along the z-axis.

Structure: Let $t := |\mathcal{P}||\mathcal{C}|\log^2 n$. Consider the projection of the points in S on the xy-plane and a $(\sqrt{n/t}) \times (\sqrt{n/t})$ grid such that each vertical or horizontal slab contains the projection of $\sqrt{nt}$ points. In each slab we construct a concurrent $Q(3,0)$ queries structure (Lemma 2). We also construct a grid data structure on all the points in S. We will describe this structure below. Finally we recurse on the points in each slab. Note that in the recursions, the grid size and the value of t vary according to the size of the subproblem. We stop the recursion when a subproblem contains less than $(|\mathcal{C}||\mathcal{P}|)^2\sqrt{\log n}$ points.

The grid data structure is defined as follows. First consider altering the x- and y-coordinates of the points in S such that if the projection of a point p lies in a cell c of the grid, then the x- and y-coordinates of p are changed to those of the bottom-left vertex of c. Let S' denote the set of modified points. The modification ensures that the points of S' lie on the grid vertices when projected onto the xy-plane. For every grid vertex p and every color c, we use $S'_{p,c}$ to denote the points of S' that have color c and lie on p when projected onto the xy-plane. The grid data structure now consists of a number of linked lists, combined denoted by D_g, and a number of query optimal $Q(3,2)$ structures by Chazelle and Guibas [11] called D. D_g consists of a list on the points in each $S'_{p,c}$, sorted by increasing z-coordinate. D consists of a $Q(3,2)$ structure for each colorset $L \in \mathcal{P}$, constructed on a set S'_L consisting of the points with the smallest z-coordinate for every set $S'_{p,c}$ with $c \in L$. Note that $|S'_L| = O(n|L|/t) = O(n|\mathcal{C}|/t)$.

Space use: The combined size of the $Q(3,0)$ data structures is $O(n)$ by Lemma 2. We claim that the grid data structures take $O(n)$ space as well. Clearly this is true for D_g, as each point is stored only once. This is also true for D since the $Q(3,2)$ data structure for S'_L takes $O(|S'_L|\log^2|S'_L|) = O(n/|\mathcal{P}|)$ space [11] and there are $|\mathcal{P}|$ such sets. Thus, the space use, $S(n)$, of the data structure is given by the recurrance

$$S(n) = 2\sqrt{\frac{n}{t}}\,S(\sqrt{nt}) + O(n), \quad t = |\mathcal{C}||\mathcal{P}|\log^2 n.$$

This solves to $S(n) = 2^r O(n)$ where r is the number of recursion steps. By Observation 1, after $f^*(n)$ steps of recursion the size of the subproblems are reduced to $O(t\log n) = O(|\mathcal{C}||\mathcal{P}|\log^3 n)$. Note that in these subproblems the value of t is $O(|\mathcal{C}||\mathcal{P}|\log^2(|\mathcal{C}||\mathcal{P}|\log^3 n))$. It is easily verified that three more recursion steps reduce the size of the subproblem to $O(|\mathcal{C}||\mathcal{P}|\log^{3/8} n \log^{O(1)}(|\mathcal{C}||\mathcal{P}|\log n)) < (|\mathcal{C}||\mathcal{P}|)^2\sqrt{\log n}$. Thus, the number of recursions is bounded by $r = \log\log n - \log\log\log n + O(1)$ and we get $S(n) = O(n\log n/\log\log n)$.

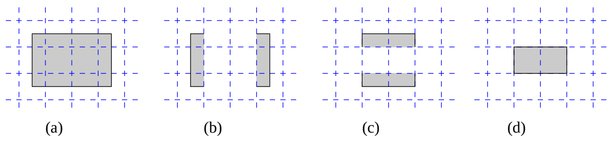

Figure 3: (a) The projection of the $Q(3,2)$ query on the xy-plane. (b) The two vertical queries. (c) The two horizontal queries. (d) The grid query.

Queries: Consider a query tuple (q, L). If the projection of q on the xy-plane is completely contained in a horizontal or vertical slab, then we answer the query recursively in that slab; at the bottom of the recursions where the number of points is less than $(|\mathcal{C}||\mathcal{P}|)^2\sqrt{\log n}$ we simply answer the query brute-force. Otherwise, q can be decomposed into two "vertical" $Q(3,1)$ queries (Figure 3(b)), two "horizontal" $Q(3,1)$ queries (Figure 3(c)), and one "grid" query, q_g, (Figure 3(d)).

We use D_g to answer the grid query. Note that as the grid query completely spans all the grid cells it intersects, answering the grid query on S' yields the same result as answering it on S. We first query D_L with q_g. This takes $O(\log n + k)$ time since all the reported points have colors from L and are contained in q_g. For every output point, we go to the corresponding list in D_g, and traverse the list until the z-coordinate of the points exceed that of q_g. It is easy to see that this procedure answers q_g correctly in $O(\log n + k)$ time.

Now, consider a vertical $Q(3,1)$ query q_1 (the horizontal queries can be answered in a similar fashion). We answer q_1 recursively on the corresponding slab. One step down in the recursion, q_1 will be decomposed into one vertical $Q(3,1)$ query, two horizontal $Q(3,0)$ queries, and one grid query. The grid query is answered as before, the $Q(3,0)$ queries are answered using the $Q(3,0)$ data structures built in each slab, and the $Q(3,1)$ query is answered recursively. Thus, the query time for a $Q(3,1)$ query is given by the recurrance

$$Q(n) = Q(\sqrt{nt}) + O(|\mathcal{C}|\log\log n + \log n)$$

which solves to $Q(n) = O(|\mathcal{C}|(\log\log n)^2 + \log n)$. Overall, the query time is $O(|\mathcal{C}|(\log\log n)^2 + (|\mathcal{C}||\mathcal{P}|)^2\sqrt{\log n} + \log n + k) = O((|\mathcal{C}||\mathcal{P}|)^2\sqrt{\log n} + \log n + k)$. Thus, we obtain the following:

THEOREM 1. *There exists an $O(n\log n/\log\log n)$ space structure for the concurrent $Q(3,2)$ problem that answers queries in $O((|\mathcal{C}||\mathcal{P}|)^2\sqrt{\log n} + \log n + k)$ time.*

COROLLARY 1. *There exists an $O(n\log n/\log\log n)$ space structure for the $Q(3,2)$ problem that can answer queries in $O(\log n + k)$ time. This is optimal.*

PROOF. A $Q(3,2)$ problem is a concurrent $Q(3,2)$ problem with $|\mathcal{C}| = |\mathcal{P}| = 1$ and thus the bounds follow from Theorem 1. Since a data structure for $Q(3,2)$ problem solves $Q(2,2)$, the optimality of the space bound follows from [9]. □

2.2 Solving $Q(3,3)$ and $Q(d,d)$

Having solved the concurrent $Q(3,2)$ problem efficiently, we can now obtain our efficient structures for the $Q(3,3)$ and

$Q(d,d)$ problems in a simple way using the following lemma, where $Q_{\mathcal{A},\mathcal{C},\mathcal{P}}(d,k)$ denotes a concurrent $Q(d,k)$ problem.

LEMMA 3. *(Afshani et al. [2]) Assume for any $S, \mathcal{C}, \mathcal{A}$, and $\mathcal{P}$ in which S is a set of n points in d-dimensional space, $\mathcal{C}$ is a set of colors, $\mathcal{A} : S \rightarrow \mathcal{C}$ and $\mathcal{P} \subset 2^{\mathcal{C}}$, that there exists a data structure that solves the $Q_{\mathcal{A},\mathcal{C},\mathcal{P}}(d,k)$ problem using $S(n, |\mathcal{C}|, |\mathcal{P}|)$ space and with $Q(n, |\mathcal{C}|, |\mathcal{P}|) + O(k)$ query time. Then a $Q_{\mathcal{A}',\mathcal{C}',\mathcal{P}'}(d+1, k+1)$ problem (respectively a $Q_{\mathcal{A}',\mathcal{C}',\mathcal{P}'}(d, k+1)$ problem) can be solved using $O(\sum_{i=1}^{\log_\alpha n} \alpha^i S(n/\alpha^i, |\mathcal{C}'|\alpha, |\mathcal{P}'|\alpha))$ space and with the query time of $O(\log_\alpha n \cdot Q(n, |\mathcal{C}'|\alpha, |\mathcal{P}'|\alpha) + k)$ (respectively $O(Q(n, |\mathcal{C}'|\alpha, |\mathcal{P}'|\alpha) + k)$) for any parameter α.*

THEOREM 2. *There exists an $O(n(\log n/\log\log n)^2)$ space data structure for the concurrent $Q(3,3)$ problem that can answer queries in $O((|\mathcal{C}||\mathcal{P}|)^2 \log^{3/4} n + \log n + k)$ time.*

PROOF. Apply Lemma 3 with $\alpha = \log^{1/16} n$ to Theorem 1. □

COROLLARY 2. *There exists an $O(n(\log n/\log\log n)^2)$ space structure for the orthogonal range reporting problem in three-dimensional space that can answer queries in $O(\log n + k)$ time. This is optimal.*

COROLLARY 3. *There exists an $O(n(\log n/\log\log n)^{d-1})$ space data structure for the orthogonal range reporting problem in d-dimensional space that can answer queries in $O(\log n(\log n/\log\log n)^{d-3} + k)$ time.*

PROOF. We get a data structure with the claimed bounds after $d-3$ applications of Lemma 3 to Theorem 2 with $\alpha = \log^\varepsilon n$ for a sufficiently small constant $\varepsilon > 0$. □

3. LOWER BOUND

In this section, we prove a query lower bound for the orthogonal range reporting problem in d dimensions. Like the previous lower bounds, we use Chazelle's general machinery on the complexity of navigation in a pointer machine [9]. However, unlike the previous attempts, we apply the lower bound techniques to the seemingly "dual" problem of point enclosure, and obtain a query lower bound rather than a space lower bound. Then we reduce point enclosure to orthogonal range reporting.

In the point enclosure problem, we are to preprocess a set of axis-aligned boxes in d-dimensional space such that we can report the boxes that contain a query point. Our lower bound is inspired by the lower bound of Arge et al. [6] for the 2-d point enclosure problem in the I/O-model. The following theorem is the main result of this section.

THEOREM 3. *Any structure that solves the d-dimensional point enclosure problem in the pointer machine model using $S(n)$ space and with $Q(n) + O(k)$ query time must have $Q(n) = \Omega((\log n/\log(S(n)/n))^{d-1})$.*

To prove Theorem 3, we will use the following result.

THEOREM 4. *(Chazelle [9]) Consider a reporting problem on a set S of n input elements and assume a data structure of size $S(n)$ achieves the query time of $Q(n) + O(k)$ for the problem where k is the output size. If there exists a set $\mathcal{Q} \subset 2^S$ such that*

(i) for every $q \in \mathcal{Q}$ there exists a query that outputs q

(ii) for every $q \in \mathcal{Q}$, $|q| = \Omega(Q(n))$

(iii) for every two $q_1, q_2 \in \mathcal{Q}$, $|q_1 \cap q_2| = O(1)$

then $S(n) = \Omega(|\mathcal{Q}|Q(n))$.

In our proof, the input set S will be a set of boxes in d-dimensional space, and each of the sets $q \in \mathcal{Q}$, will be a set of boxes corresponding to the output of a point enclosure query on S for some query point p. With an abuse of notation, we will refer to the elements of $\mathcal{Q}$ as queries. We thus prove our lower bound by constructing two sets S and $\mathcal{Q}$ that satisfies the conditions of Theorem 4.

Our input set and query set are almost identical to several previous lower bound constructions for orthogonal range reporting. However, we use different parameters and points and boxes play different roles; boxes are the input and points are the queries.

The input set. Let M, α and t be parameters that will be fixed later. Consider a d-dimensional box B with side lengths M. All the input boxes will have fixed volume of αM^{d-1} and will be contained in B. For every combination of indices $i_1, \ldots, i_{d-1} \in \{1, \ldots, \lfloor (1/d) \log_t M \rfloor\}$, we build a box of dimensions $M/t^{i_1} \times M/t^{i_2} \times \cdots \times M/t^{i_{d-1}} \times \alpha t^{i_1 + \cdots + i_{d-1}}$ and place it in a set of boxes R. We refer to the set R as the set of box types, and say that a box has type $(i_1, \ldots, i_{d-1})$ if its dimensions equal that of the box in R constructed from those indices. It is easy to see that $|R| = \Theta(\log_t^{d-1} M)$. Our input set S is now obtained by selecting each type $r \in R$, and then placing $\Theta(M/\alpha)$ boxes of type r inside B. The boxes are placed by tiling them from the lower corner of B, so that the boxes of the same type are disjoint. Two boxes of different type can intersect however. For simplicity, we assume the boxes of each type completely cover B (this assumption can be removed with more details).

The query set. The query set can be constructed through a variety of techniques. Essentially, what we need is a point set of size $\Theta(M)$ with the property that any orthogonal box that contains $k \geq 2$ points has volume $\Omega(kM^{d-1})$. Intuitively, this models a point set placed uniformly at random inside B as such a randomized point set will have this property "on average" [9]. However, such a set can also be constructed deterministically [10].
The number of boxes in the input set is $\Theta((M/\alpha)\log_t^{d-1} M)$. We thus have:

$$n = \Theta((M/\alpha)\log_t^{d-1} M). \tag{C-1}$$

We are now ready assign values to M, α and t that satisfy conditions (i) and (ii) of Theorem 4. Because of the tiling, any given query point is contained in precisely $\Theta(\log_t^{d-1} M)$ boxes, one for each type. To satisfy condition (i) of Theorem 4, we therefore require that

$$\log_t^{d-1} M \geq Q(n). \tag{C-2}$$

Now consider two query points p_1 and p_2. To ensure condition (ii) of Theorem 4, we need to select M, α and t such that only a constant number of input boxes can contain both p_1 and p_2. By the property of the point set, the box b spanned by p_1 and p_2 has volume $\Omega(M^{d-1})$. Letting $w_1 \times w_2 \times \cdots \times w_d$ be the dimensions of b, this translates to $\Pi_{i=1}^d w_i = \Omega(M^{d-1})$. Clearly any input box that contains both p_1 and p_2 must also contain b. Since only one box of each type can contain b, we simply need to bound the maximum number of different types that can potentially contain b. A box of type $(i_1, \ldots, i_{d-1})$ can contain b only if

$$M/t^{i_j} \geq w_j \quad \text{for } 1 \leq j \leq d-1 \quad \text{and} \tag{1}$$
$$\alpha t^{i_1+\cdots+i_{d-1}} \geq w_d \tag{2}$$

By multiplying all the inequalities of (1) we obtain that $M^{d-1}/t^{i_1+\cdots+i_{d-1}} \geq w_1 \ldots w_{d-1} = \Omega(M^{d-1}/w_d)$, which results in the requirement $w_d = \Omega(t^{i_1+\cdots+i_{d-1}})$. Combined with (2) we get that a box of type $(i_1, \ldots, i_{d-1})$ can contain b only if $c_1 \cdot w_d \geq t^{i_1+\cdots+i_{d-1}} \geq w_d/\alpha$ for some constant c_1. Thus there are only $O(\log_t \alpha)$ different values of $s := i_1 + \cdots + i_{d-1}$ that satisfy (1) and (2), and these values are consecutive integers. By (1) we get that the lowest value of s satisfying the requirements is obtained by letting $i_j = \lceil \log_t(M/w_j) \rceil$ for all j. It follows that i_j must satisfy $\log_t(M/w_j) \leq i_j \leq \log_t(M/w_j) + O(\log_t \alpha)$, which means the total number of different choices of indices, and thus types possibly containing b, is bounded by $O(\log_t^{d-1} \alpha)$. To satisfy condition (ii) of Theorem 4 we therefore must have

$$\log_t^{d-1} \alpha = O(1). \tag{C-3}$$

Any choice of parameters M, α and t that satisfy conditions (C-1),(C-2), and (C-3) will give us a space lower bound of $\Omega(|\mathcal{Q}|Q(n))$. To satisfy (C-3), we set $t = \alpha$, to satisfy (C-2) we pick α such that $\log_\alpha^{d-1} M = Q(n)$, and a slightly more complicated formula gives a value for M that satisfies (C-1). These values give the space lower bound of $\Omega(|\mathcal{Q}|Q(n)) = \Omega(MQ(n)) = \Omega(M \log_\alpha^{d-1} M) = \Omega(n\alpha)$. It can be easily verified that with this choice of parameters $\log M = \Theta(\log n)$ and thus $\log_\alpha^{d-1} M = \Theta(\log_\alpha^{d-1} n)$; combined with (C-2) this gives $\log_\alpha^{d-1} n = \Theta(Q(n))$. We solve this for α and obtain that $\log \alpha = \Theta(\log n / Q(n)^{1/(d-1)})$. Substituting this value in the space lower bound we obtain that

$$Q(n) = \Omega\left(\left(\frac{\log n}{\log \frac{S(n)}{n}}\right)^{d-1}\right).$$

This concludes the proof of Theorem 3.

Corollary 4. *Any structure that solves the d-dimensional orthogonal range reporting problem in the pointer machine model using $S(n)$ space and with $Q(n) + O(k)$ query time must have $Q(n) = \Omega((\log n / \log(S(n)/n))^{\lfloor d/2 \rfloor - 1})$.*

Proof. It is easy to see that d-dimensional point enclosure reduces to $(2 \cdot d)$-dimensional orthogonal range reporting: an input box $[a_1, b_1] \times \cdots \times [a_d, b_d]$ is mapped to an input point $(a_1, b_1, \ldots, a_d, b_d)$ and a query point $(q_1, \ldots, q_d)$ is mapped to a query box $[-\infty, q_1] \times [q_1, \infty] \times \ldots [-\infty, q_d] \times [q_d, \infty]$. $\square$

4. POINT LOCATION IN RECTILINEAR SUBDIVISIONS

In this section we consider point location in d-dimensional rectilinear subdivisions. We improve the bounds for this problem using an approach similar to the one employed in Section 2.

The input to the concurrent version of the problem is a set S of n boxes in d dimensions and a color assignment $\mathcal{A} : S \to \mathcal{C}$. The color assignment has the property that all the boxes mapped to the same color are disjoint. A query consists of a point q and a set of colors $L \subseteq \mathcal{C}$. The task is to preprocess the input, such that given a query (q, L), we can report the boxes r in S for which $q \in r$ and $\mathcal{A}(r) \in L$. We use $P_{\mathcal{A},\mathcal{C}}(d)$ to denote this problem.

First we solve the concurrent 2-d point location in rectilinear subdivisions problem.

Lemma 4. *There exists an $O(n)$ space structure for $P_{\mathcal{A},\mathcal{C}}(2)$ that answers queries in $O(|\mathcal{C}|^2 + \log n)$ time.*

Proof. We construct a linear-space 2-d point enclosure data structure on all the input boxes [8]. To answer a query (q, L), we query the point enclosure data structure with q. This returns a set R of all boxes in S that contain q and takes $O(\log n + |R|)$ time. We output each $r \in R$ if $\mathcal{A}(r) \in L$. The total time is then bounded by $O(|R||\mathcal{C}| + \log n)$. Since the boxes with the same color are disjoint, at most one box of each color can contain q, thus $|R| \leq |\mathcal{C}|$ and the result follows. $\square$

To obtain our high dimensinal point location data structure we need an analogous of Lemma 3. Let X_{d+1} be a sorted set containing the $2n$ values of the coordinates of the last dimension of boxes in S. Build a balanced tree T of fanout t on X_{d+1} and associate each box $r \in S$ with the least common ancestor of the two leaf nodes containing the coordinates of r in the last dimension. Thus, every node v in T is associated with a set $S_v \subseteq S$ of input boxes. We project S_v into the first d dimensions and implement a data structure for $P_{\mathcal{A}_v,\mathcal{C}_v}(d)$ on the projected points in which $\mathcal{A}_v$ and $\mathcal{C}_v$ are defined as follows: Let $c_1, \ldots, c_t$ be the children of v. Define $\mathcal{C}_v = \mathcal{C} \times \{1, \ldots, t\} \times \{1, \ldots, t\}$, and for a box $r \in S_v$, we define $\mathcal{A}_v(r) = (\mathcal{A}(r), i, j)$, in which where i is the index of the child whose subtree contains the smallest of r's coordinates in the last dimension, and j is the index of the child containing the largest.

Answering Queries. Let (q, L) be a $P_{\mathcal{A},\mathcal{C}}(d+1)$ query. We start out at the root node v in T, and compute the index i of the child c_i of v containing the last coordinate of q. We then query the $P_{\mathcal{A}_v,\mathcal{C}_v}(d)$ data structure stored in v with (q', L') as the query, in which $L' = L \times \{1, \ldots, i\} \times \{i, \ldots, t\}$ and q' is the projection of q onto the first d dimensions. This returns a set $R \subseteq S_v$ of boxes. For each box $r \in R$, we check whether q is inside r in the last dimension, and if so we report r. Finally, we recurse on child c_i.

Analysis. First we argue that our color assignments satisfy the disjointness requirement: Consider two d-dimensional boxes r_1 and r_2 in S_v such that $\mathcal{A}_v(r_1) = \mathcal{A}_v(r_2) = (c, i, j)$. Let r_1' and r_2' be the $(d+1)$-dimensional boxes whose projection onto the first d dimensions is r_1 and r_2, respectively. Since $\mathcal{A}(r_1') = \mathcal{A}(r_1') = c$, we know r_1' and r_2' are disjoint. Furthermore, in the last dimension, both r_1' and r_2' contain the coordinate separating c_i and c_{i+1}. This implies r_1 and r_2 are disjoint. That the query algorithm returns the correct boxes when querying for (q, L) follows from the definition of (q', L'). Thus, we get the following lemma.

Lemma 5. *Assume for any $S, \mathcal{C}$, and $\mathcal{A}$ there exists a data structure that solves $P_{\mathcal{A},\mathcal{C}}(d+1)$ using $S(n, |\mathcal{C}|)$ space and with $Q(n, |\mathcal{C}|)$ query time. Then, $P_{\mathcal{A}',\mathcal{C}'}(d+1)$ can be solved using $S(n, |\mathcal{C}'|\alpha)$ space and with the query time of $O(\log_\alpha n \cdot Q(n, |\mathcal{C}'|\alpha))$ for any parameter α.*

Combining Lemma 5 and Lemma 4 gives us the following result.

THEOREM 5. *There exists an $O(n)$ space data structure for point location in d-dimensional rectilinear subdivisons that answers queries in $O(\log n(\log n/\log\log n)^{d-2})$ time.*

PROOF. We get a data structure with the claimed bounds after $d-2$ applications of Lemma 5 to Lemma 4 with $\alpha = \log^{\varepsilon} n$ for a sufficiently small constant $\varepsilon > 0$. $\square$

5. CONCLUSION

In this paper we described a space optimal pointer machine data structure for 3-d orthogonal range reporting that answers queries in $O(\log n + k)$ time. We also described improved higher-dimensional structure and gave the first nontrivial query lower bound for the problem. One important implication of our results is that the optimal query bound increases from $O(\log n+k)$ to $\Omega((\log n/\log\log n)^2+k)$ somewhere between three and six dimensions. One intriguing open problem is of course to pinpoint the dimension where the optimal query bound jumps. We conjecture that it happens in four dimensions. Our query upper bound increases by a $\log n/\log\log n$ factor every dimension while the increase in the lower bound happens every other dimension. Another interesting question is therefore which of the two is closer to the behavior of the optimal query bound. In this context, it is worth noting that many other bounds in computational geometry, such as the worst case convex hull complexity, and halfspace range reporting query bounds, increase every other dimension.

6. REFERENCES

[1] P. Afshani. On dominance reporting in 3D. In *Proc. European Symposium on Algorithms*, pages 41–51, 2008.

[2] P. Afshani, L. Arge, and K. D. Larsen. Orthogonal range reporting in three and higher dimensions. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 149–158, 2009.

[3] 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.

[4] P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. In B. Chazelle, E. Goodman, and R. Pollack, editors, *Discrete and Computational Geometry: Ten Years Later*. Mathematical Society Press, 1997.

[5] S. Alstrup, G. S. Brodal, and T. Rauhe. New data structures for orthogonal range searching. In *Proc. 41st Annual Symposium on Foundations of Computer Science*, pages 198–207, 2000.

[6] L. Arge, V. Samoladas, and K. Yi. Optimal external memory planar point enclosure. In *Proc. European Symposium on Algorithms, LNCS 3221*, pages 40–52, 2004.

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

[8] B. Chazelle. Filtering search: a new approach to query-answering. *SIAM J. Comput.*, 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, Apr. 1990.

[10] B. Chazelle. Lower bounds for off-line range searching. In *STOC '95: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 733–740, New York, NY, USA, 1995. ACM.

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

[12] H. Edelsbrunner, G. Haring, and D. Hilbert. Rectangular point location in d dimensions with applications. *The Computer journal*, 29:76–82, 1986.

[13] M. Karpinski and Y. Nekrich. Space efficient multi-dimensional range reporting. In *COCOON '09: Proceedings of the 15th Annual International Conference on Computing and Combinatorics*, pages 215–224, 2009.

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

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

[16] 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.