Improved Pointer Machine and I/O Lower Bounds for Simplex Range Reporting and Related Problems

Peyman Afshani
MADALGO,*
Department of Computer Science,
Aarhus University
peyman@cs.au.dk

Received (01 Oct., 2012)
Revised (22 Jan. 2014)
Communicated by (Gill Barequet)

We investigate one of the fundamental areas in computational geometry: lower bounds for range reporting problems in the pointer machine and the external memory models. We develop new techniques that lead to new and improved lower bounds for simplex range reporting as well as some other geometric problems.

Simplex range reporting is the problem of storing n points in the d-dimensional space in a data structure such that the k points that lie inside a query simplex can be found efficiently. This is one of the fundamental and extensively studied problems in computational geometry. Currently, the best data structures for the problem achieve $Q(n) + O(k)$ query time using $\tilde{O}((n/Q(n))^d)$ space in which the $\tilde{O}(.)$ notation either hides a polylogarithmic or an n^ε factor for any constant $\varepsilon > 0$, (depending on the data structure and $Q(n)$). The best lower bound on this problem is due to Chazelle and Rosenberg who showed any pointer machine data structure that can answer queries in $O(n^\gamma + k)$ time must use $\Omega(n^{d-\varepsilon-d\gamma})$ space. Observe that this bound is a polynomial factor away from the best known data structures.

In this article, we improve the space lower bound to $\Omega((n/Q(n))^d/2^{O(\sqrt{\log Q(n)})})$. Not only this bridges the gap from polynomial to sub-polynomial, it also offers a smooth trade-off curve. For instance, for polylogarithmic values of $Q(n)$, our space lower bound almost equals $\Omega((n/Q(n))^d)$; the latter is generally believed to be the "right" bound.

By a simple geometric transformation, we also improve the best lower bounds for the halfspace range reporting problem. Furthermore, we study the external memory model and offer a new simple framework for proving lower bounds in this model. We show that answering simplex range reporting queries with $Q(n) + O(k/B)$ I/Os requires $\Omega(B(n/(BQ(n)))^d/2^{O(\sqrt{\log Q(n)})})$ space or $\Omega((n/(BQ(n)))^d/2^{O(\sqrt{\log Q(n)})})$ blocks, in which B is the block size.

Keywords: Range searching; lower bounds; pointer machine; I/O model.

1. Introduction

Range reporting is one of the most fundamental areas in computational geometry. In a range reporting problem the goal is to preprocess an input set of n points in d dimensions

*Center for Massive Data Algorithmics, a center of the Danish National Research Foundation.

such that given a query range r, the points inside r can be found efficiently. Three types of queries are studied much more than others: simplices, halfspaces, and axis-aligned boxes and they lead to simplex range reporting, halfspace range reporting, and orthogonal range reporting, respectively.

The progress on range reporting has been significant and research in this area has resulted in development of many classical tools and techniques that have been useful in many other areas of computational geometry as well. Contrary to this, there are very few techniques available when it comes to proving lower bounds. In this paper, we develop a new framework to prove lower bounds for range reporting problems and thus obtain improved lower bounds for simplex range reporting in two fundamental models of computation, the pointer machine model and the I/O model.

1.1. *Background*

Many of the techniques that are used to solve simplex range reporting are in fact developed for the general problem of simplex range searching. In this problem, the input points are assigned weights from a semi-group and for a query simplex r, the output is the sum of the semi-group weights of the points inside r. The model of computation assumes that the sum of any number of semi-group elements fits into one register and thus the output has constant size. For more details, refer to surveys on range searching [5,3].

This is different from simplex range reporting in which the output often has super-constant size. The greatest distinction, however, lies in the way data structures are built. A data structure for a range searching problem in the semi-group model stores a number of precomputed sums and this takes space equal to the number of sums; the query is answered by selecting some of the stored precomputed sums that cover the query range and then summing them. In a range reporting data structure, storing precomputed answers would take too much space but on the other hand, the existence of an output term allows the data structure to spend more time on queries that produce large outputs. This idea, known as the "filtering search" [9], is fundamental when designing efficient range reporting data structures. Fortunately, many of the existing techniques for building data structures (such as partition trees) can be easily adapted to apply to both problems. Thus, very few papers that deal with building data structures distinguish these two variants. Unfortunately, from a lower bound point of view, the two problems are fundamentally different and there is no general technique that can be applied to both problems.

Previous results. Much of the focus on simplex range searching is devoted to building linear-size or near linear-size data structures. Such results were published as early as 1982 [23] and they offer $O(n^\gamma + k)$ query time, (the constant γ depends on the data structure) while using linear space. After a number of results that lowered the exponent γ in two and three dimensions, significant breakthroughs came via works of researchers such as Welzl, Chazelle, Sharir, and Matoušek [22,16,21] culminating in Matoušek's work [20] that presented a data structure with linear space and $O(n^{1-1/d} + k)$ query time (also an $O(n^{1-1/d})$ query time for the semi-group version). Very recently, Timothy Chan proved a new partition theorem that simplifies Matoušek's approach and offers other advantages [8]. In the external memory model and using linear space, it is possible to answer simplex range reporting queries with $O((n/B)^{1-1/d+\varepsilon} + k/B)$ I/Os in which ε is any positive constant and B is the block size [4].

At the other side of the spectrum, data structures with polylogarithmic query time were obtained through *cuttings* [13,17,12]. Chazelle et al. [16] showed that using $O(n^{d+\varepsilon})$ space it is possible to achieve query time of $O(\log n + k)$. Matoušek showed query time of $O(\log^{d+1} n + k)$ is possible using $O(n^d)$ space [20]. By combining these with linear space data structures, it is possible to strike a trade-off between query time and space usage: one can build a data structure that uses $m + O(n)$ space, for any parameter m, that

can answer queries in $O((n/m^{1/d})\log^{d+1}(m/n)+k)$ time [20]. In light of this result, it is reasonable to conjecture that the space complexity, $S(n)$, and the query complexity, $Q(n)+O(k)$, of a pointer machine data structure that answers simplex range reporting queries should respect $S(n)=O((n/Q(n))^d)$ and that the extra polylogarithmic factors in the above result are artifacts of the proof. This conjecture has not been settled yet.

All of the previous simplex range reporting results can be implemented in the pointer machine model of computation and currently there are no better bounds in the RAM model and it seems random accesses do not help to reduce the complexity of this problem.

While there are a number of lower bounds for range searching problems (in semi-group or even group models), we are aware of only three papers that are relevant to simplex range reporting. The first paper by Erickson [18] deals with hyperplane emptiness queries. As such queries can be answered using simplex range reporting queries, his lower bounds also apply to simplex range reporting problem. His results are proved in a restricted model of computation known as the *partition graph model* (intuitively, it models pointer machine data structures that work by partitioning the space). He proved that any data structure that solves hyperplane emptiness queries in the partition graph model is also able to solve halfspace range searching queries in the semi-group model. Using a recent result of Arya et al. [7], this gives the query lower bound of $\tilde{\Omega}(n^{1-1/(d+1)}/m^{1/(d+1)})$ in which the $\tilde{\Omega}(\cdot)$ notation hides polylogarithmic factors. Unfortunately, the exponent of this bound is not tight but Erickson's range emptiness lower bound is significant because proving lower bounds for emptiness queries is extremely difficult. The second paper is due to Chazelle and Liu [14] in which they proved that given a planar subdivision with n vertices, computing all the k edges intersected by a query line in $Q(n)+O(k)$ time requires $\Omega(n^2/Q(n)^2)$ space. While they do not mention simplex range reporting in the paper, the particular input instance they build for this problem also gives the same lower bound for simplex range reporting queries in the plane[a]. Unfortunately, the technique does not generalize to higher dimensions.

The third paper by Chazelle and Rosenberg [15] is the most relevant one. It operates in a general pointer machine model and it builds on Chazelle's earlier works [10,11] and offers a space lower bound of $\Omega(n^{d-d\gamma-\varepsilon})$ assuming the data structure can answer queries in $O(n^\gamma+k)$ time in which ε is any positive constant. This lower bound is a polynomial factor away from the (conjectured optimal) trade-off curve and becomes meaningless for small $Q(n)$.

Our results.

Simplex range reporting. We show any pointer machine data structure that can answer simplex range reporting queries in $Q(n)+O(k)$ time needs to consume $\Omega((n/Q(n))^d/2^{O(\sqrt{\log Q(n)})})$ space. This improves Chazelle and Rosenberg's lower bound and achieves a sub-polynomial overhead[b]. Furthermore, it offers a smooth trade-off curve. For instance, for polylogarithmic values of $Q(n)$, we are only a sub-logarithmic factor away from the trade-off curve $S(n)=\Omega((n/Q(n))^d)$.

Simplex and Slab stabbing. Our lower bound is first obtained for a different problem we call the *slab stabbing problem*: preprocess a set of slabs (the region between two parallel hyperplanes) such that given a query point, the slabs containing the query point can be reported efficiently. Note that this also gives a lower bound for simplex stabbing problem: preprocess a set of simplices such that given a query point, the simplices containing the query point can be reported efficiently.

[a]It is possible the authors are unaware of this important consequence of their results!

[b]Note that $2^{\sqrt{Q(n)}}$ lies asymptotically between $(\log Q(n))^{O(1)}$ and $Q(n)^\varepsilon$ because we have $(\log Q(n))^{O(1)}=2^{O(\log\log Q(n))}$ while $Q(n)^\varepsilon=2^{\varepsilon\log Q(n)}$.

Halfspace range reporting. Using a known geometric transformation [18], our slab stabbing lower bound offers the same space/query trade-off (i.e., $S(n) = \Omega((n/Q(n))^d/2^{O(\sqrt{\log n})})$) for halfspace range reporting in $d(d+3)/2$ dimensions. This reduction from slab stabbing is the only available technique for proving lower bounds for halfspace range reporting, making our lower bound the best known lower bound for this problem as well.

External memory. Our main result here is a new framework for proving range reporting lower bounds in the external memory model. The obvious extensions give a space lower bound that is a B factor away from the best we can obtain (B is the block size). Rather surprisingly, we prove that if an input set is difficult for geometric stabbing problem in the pointer machine model, then by replicating each range roughly B times (in fact B/α times for some parameter α which will always be smaller than $\log n$) we will obtain an input set that is difficult in the external memory model. Using this, we obtain our lower bound for simplex range reporting in the external memory model: answering queries with $Q(n) + O(k/B)$ I/Os requires $S(n) = \Omega(B(n/(BQ(n)))^d/2^{O(\sqrt{\log Q(N)})}$ space. This is the first known external memory lower bound for simplex range reporting.

2. A Simple Geometric Lower Bound Framework

In this section, we describe our simple framework that is tailored for geometric problems. We begin by introducing the model of computation.

We operate in a powerful variant of the pointer machine model which also has been employed by the previous lower bound papers [11]. Consider an abstract reporting problem in which queries output subsets of an input set $\mathcal{S}$ containing n elements. In this model, a data structure that solves such an abstract reporting problem is represented by a directed graph G with maximum out-degree of two. The set of vertices of G, $V(G)$, represents the set of memory cells used by the data structure. Each memory cell stores an input element as well as two pointers to two other memory cells. Any other information can be stored and accessed for free by the data structure. There are two main restrictions: First, it is assumed that to output an element $p \in \mathcal{S}$, the query algorithm must visit a vertex $u \in V(G)$ that stores p. Second a cell can only be accessed through navigating pointers and thus random accesses are disallowed. These imply that at the query time a connected subgraph of G is explored whose size is a lower bound for the query time. Given a query q, we call a vertex u an *output vertex* if the query outputs the element stored at u.

2.1. *Background and Preliminaries*

Most often proving data structure lower bounds requires two important components: one, a "bad" input instance that describes both an input set and a difficult to answer query set and two, a set of tools to exploit bottlenecks that the model imposes on the query algorithm.

In a pointer machine, the second component involves exploiting the constant out-degree of the memory cells: starting from a vertex (or memory cell) u, one can visit at most 2^r other vertices by making r pointer accesses. Our results as well as all the other pointer machine lower bounds for reporting problems crucially only use this and the other trivial fact that the query algorithm explores a connected subgraph of G. Nonetheless, transforming these trivial observations into working frameworks for proving lower bounds is not easy. Chazelle (and later with Rosenberg) [11,15] provided a combinatorial framework that achieved this: to get a space lower bound for data structures with $Q(n)+O(k)$ query time one needs to show the existence of subsets $q_1, \ldots, q_m \subset \mathcal{S}$, where each q_i is the output of some query, with the following two properties: (Ch.1) $|q_i| \geq Q(n)$, for every $1 \leq i \leq m$, and (Ch.2) for a fixed parameter α and for every α distinct subsets $q_{i_1}, \ldots, q_{i_\alpha}$, $|q_{i_1} \cap \cdots \cap q_{i_\alpha}| = O(1)$. If such subsets can be shown to exist, then one

automatically obtains the space lower bound of $\Omega(\sum_{i=1}^{m} |q_i|/\alpha) = \Omega(mQ(n)/\alpha)$ using their framework. Note that this is a combinatorial statement that can be succinctly summarized as follows: to output subsets $q_1, \ldots, q_m$ with properties (Ch.1) and (Ch.2), a pointer machine data structure must have $\Omega(\sum_{i=1}^{m} |q_i|/\alpha) = \Omega(mQ(n)/\alpha)$ cells.

On the other hand, the first component of a successful lower bound proof involves exploiting the special properties of the problem (rather than the model). For the geometric problems, this is where the geometry comes into play, as here we must geometrically build a set of ranges as well as a set of query points. For instance, to obtain their lower bound, Chazelle and Rosenberg [15] build m regions (simplices) $r_1, \ldots, r_m$ inside the unit hypercube with the following properties: (P1) the volume of each region is at least μ (for some parameter $\mu < 1$), and (P2) the volume of the intersection of every $\log m$ regions is much smaller, roughly $O(\mu^d m)$ (the precise bound is $O(\mu^d m(\log m)^{d-2})$). We will visit the details of the construction later but for the moment it suffices to say that the required techniques are purely geometric.

Intuitively, requirements (Ch.1) and (Ch.2) of the framework align nicely with the properties (P1) and (P2) of the geometric construction. To formally connect them, they place n points uniformly inside the unit hypercube: by property (P1), each region has volume at least μ so on average each region will receive at least $n\mu$ points which can be used to satisfy (Ch.1), and (P2) ensures that the expected number of points at the intersection of every $\log m$ regions is $O(n\mu^d m(\log m)^{d-2})$, which means every $\log m$ queries share on average $O(n\mu^d m(\log m)^{d-2})$ points. So intuitively the only thing left to do is to set $\alpha = \log m$, and then find the right values of parameters μ and m such that we have $n\mu \geq Q(n)$ (to satisfy (Ch.1)) and $O(n\mu^d m(\log m)^{d-2}) = O(1)$ (to satisfy (Ch.2)). And then we have a lower bound.

Unfortunately, the above method of connecting the two components introduces a very difficult to resolve technical problem: if we set m and μ such that $O(n\mu^d m(\log m)^{d-2}) = O(1)$, it is still possible that for some $\alpha = \log m$ regions $r_{i_1}, \ldots, r_{i_\alpha}$, a lot of points, that is $\omega(1)$ points, are placed at their intersection by pure bad luck even though on average there was supposed to be only $O(1)$ points present at their intersection. In fact, it can be shown that such unlucky events are guaranteed (with high probability) to happen since a uniform random placement of points is guaranteed to create regions with higher than average densities and there are no known techniques to fix or "smoothen out" such irregularities. To resolve this technical issue, Chazelle and Rosenberg were forced to make a suboptimal choice of values for μ and m to guarantee that $n\mu^d m(\log m)^{d-2} \leq n^{-\varepsilon}$ for some fixed constant $\varepsilon > 0$. This introduces the n^{ε} factor reduction in their space lower bound.

To summarize this discussion, Chazelle's (and Rosenberg's) framework is composed of two parts: a purely combinatorial theorem and a purely geometric construction and to "glue" them together they use a uniformly distributed set of points inside the unit cube. Unfortunately, the uniformly distributed point set is guaranteed to have irregularities that the combinatorial theorem cannot deal with and thus their lower bound loses some efficiency.

2.2. *Our Framework*

In contrast to Chazelle's two-layered framework, we use a more direct attack, by combining both components in one theorem. To do that, first we restrict ourselves to the case when $\mathcal{S}$ is a set of n geometric regions inside a d-dimensional hypercube $\mathcal{Q}$ of volume one. We call an elements of $\mathcal{S}$ a range. The queries are points inside $\mathcal{Q}$ and the output of a query q is the subset of ranges that contain q. We call such a problem a *geometric stabbing problem*. Observe that geometric stabbing problems are "dual" of range reporting problems in the sense that the roles of the points and the ranges are swapped. A careful reader can check that this swapping of the roles in itself is effective and it can reduce the gap in Chazelle's framework from n^{ε} to $Q(n)^{\varepsilon}$. To get further improvements, we need to work harder.

The main result of this section is the following.

Theorem 1. *Assume we have a data structure for a geometric stabbing problem that uses at most $S(n)$ space and answers queries within $Q(n) + O(k)$ time in which n is the input size and k is the output size. Assume for this problem we can construct an input set $\mathcal{S}$ of n ranges such that (i) every point of $\mathcal{Q}$ is contained in exactly t ranges in which t is a parameter greater than $Q(n)$ and (ii) the volume of the intersection of every α ranges is at most v, for two parameters $\alpha < t$ and v. Then, we must have $S(n) = \Omega(tv^{-1}/2^{O(\alpha)}) = \Omega(Q(n)v^{-1}/2^{O(\alpha)})$.*

The above theorem admittedly looks a bit strange and unintuitive. However, its greatest advantage is that it offers a unified framework. We no longer need to deal with points and thus the technical difficulty of dealing with irregularities in a random distribution of the points is avoided; a construction of ranges with small intersection volume property directly results in a lower bound.

To prove the above theorem, we consider the directed graph G that represents the memory layout of the data structure. We say a subgraph H of G is a *hub of size α* if H is a tree[c] with at most α vertices. Before giving the intuition behind our proof, we bound the maximum possible number of hubs.

Lemma 1. *The number of hubs of size α is $O(|V(G)|2^{2\alpha})$.*

Proof. Consider a vertex u. We count the maximum number of hubs of size α with u as their root. Let u_1 and u_2 be the two nodes that u points to. If H is a hub of size i rooted at u, and H_1 and H_2 are subtrees of H rooted on u_1 and u_2, then the size of H_2 is $i - j - 1$ in which j is the size of H_1. Let $f(i)$ denote the maximum number of hubs of size i that can share a common root. We have

$$f(i) = \sum_{0 \leq j < i} f(j)f(i - j - 1)$$

and $f(0) = 1$. The solution to this recursive formula is known as a Catalan number and it is not too hard to show that $f(i) \leq 2^{2i}$ (e.g., it counts number of expressions with i matching parentheses which is less than the number of binary strings of size $2i$). Summing this for $i = 1$ to α gives the maximum number of hubs with a common root, which is $O(2^{2\alpha})$. Thus, the total number of hubs of size α is $O(|V(G)|2^{2\alpha})$. □

Now consider the subgraph of G explored at query time. As we are operating in the pointer machine model, the query algorithm starts from the root of G and begins exploring G by traversing edges. For every vertex u visited by the query algorithm, we mark the in-edge of u that is used to reach u for the first time during the query. Note that each vertex (except the root) receives exactly one marked in-edge. Thus, the marked edges form a tree T. We call a subtree H of T an *α-heavy hub* if the size of H is $O(\alpha)$ and α of its vertices store output elements of the query. Note that the definition of an α-heavy hub only makes sense with respect to the subgraph explored at query time. In other words, different queries define different α-heavy hubs.

Our general attack plan is the following. Intuitively, a hub represents a set of vertices that can be explored together with little cost (a packed cluster of vertices). Lemma 1 can be used to prove that the total number of different α-heavy hubs is $|V(G)|2^{O(\alpha)}$, i.e., directly proportional to the size of the data structure. On the other hand, if the output size is large enough, then the query time is $O(k)$, which means, a substantial fraction of the nodes in T store the output elements (ranges). This can be used to prove that T

[c]This is in fact a special directed "tree" in which all the edges are directed away from the root. Thus, the technically correct term to use is "arborescence". For simplicity, we will stick with the word "tree".

contains many α-heavy hubs. However, an α-heavy hub stores α output ranges and thus the query point should be inside all of these α ranges. In other words, the α-heavy hub can only be used for the queries inside the intersection region of the α output ranges. However, each point in $\mathcal{Q}$ can be a query point which means $\mathcal{Q}$ is covered by the regions coming from the intersection of α ranges. Thus, the maximum volume of the intersection region, and the maximum number of available α-heavy hubs can be combined to prove Theorem 1. Now we present the details.

In the next lemma, we prove that the subtree T explored at query time must indeed contain many α-heavy hubs.

Lemma 2. *Consider the tree T explored at query time. If T contains $\Omega(|T|)$ output vertices, then there exists a set $\mathcal{H}$ of $\Omega(|T|/\alpha)$ α-heavy hubs.*

Proof. Let k be the number of output vertices in T. We find $\mathcal{H}$ by playing the following game. We initialize the game by placing one pebble on each output vertex of T and assign a size value to every node of T which is initially one. Thus, in total k pebbles are placed on T and total size of all the vertices is $|V(T)| = O(k)$.

At each step of the game we operate on a leaf u of T. Let w be the parent of u. In a typical step, we take all the pebbles on u and place them on w, then add the size of u to w and then we delete u. The exception is when there are α ore more pebbles on u. In this case, we do not perform the typical step and instead look at the size of u: if it is greater than $c\alpha$ (the first case), for a large enough constant c, then we delete u and throw away all the existing pebbles on u and we call these pebbles *wasted*. Otherwise (the second case), we have found an α-heavy hub (to see this, imagine rewinding the pebble game from u; we reach α output vertices using at most $c\alpha$ edge traversals). We add the α-heavy hub that we just found to $\mathcal{H}$ and then remove u and its pebbles from the game. In the special case when u contained more than α pebbles, we only use α pebbles to find an α-heavy hub and the excess pebbles are again wasted.

Observe that when we perform the exceptional step, the number of pebbles on u is less than 2α: u begins by having at most one pebble and since it has out-degree of two, it can receive at most $\alpha - 1$ pebbles from each of its children during a typical step. Thus, in this case the exceptional step wastes less than α pebbles.

At the end of the game, we will have a set of at most $\alpha - 1$ pebbles left at the root of T and all the other pebbles will either be wasted or placed in a hub in $\mathcal{H}$. In the first case of the exceptional step, wasting less than 2α pebbles corresponds to deleting $c\alpha$ vertices of T and in the second case, α pebbles are placed in a hub and less than α are wasted. Thus, it follows that at most $2k/3$ pebbles will be wasted if c is chosen large enough. The rest of the pebbles, $k/3 - \alpha + 1$, will be in hubs. This means $\mathcal{H}$ contains at least $(k/3 - \alpha + 1)/\alpha$ hubs. □

Proof of Theorem 1.

By Lemma 1, the total number of hubs of size $O(\alpha)$ is $O(|V(G)|2^{O(\alpha)})$. Consider a hub H of size $O(\alpha)$. There are $\binom{O(\alpha)}{\alpha} = O(2^{O(\alpha)})$ possible ways for H to be an α-heavy hub at query time, over all the possible queries. However, an α-heavy hub H contains α output vertices that output α ranges $r_1, \ldots, r_\alpha$ which means the hub can only be used by query points that are inside the intersection of $r_1, \ldots, r_\alpha$; we call this region the *intersection region of H*. By our assumption, the intersection region of any hub has volume at most v.

Let T be the tree explored at query time, for a given query. Remember that each point of $\mathcal{Q}$ is covered by t ranges and since the query time is $Q(n) + O(t)$, it follows that $|V(T)| \leq Q(n) + O(t) = O(t)$. But $V(T)$ must also contain t output vertices and thus by Lemma 2, we can identify $\Omega(t/\alpha)$ α-heavy hubs in T, meaning, the query point must be inside the intersection region of $\Omega(t/\alpha)$ α-heavy hubs. As every point of $\mathcal{Q}$ can be used as a query, it follows that the intersection regions of all the possible α-heavy hubs have

to cover $\mathcal{Q}$ at least $\Omega(t/\alpha)$ times. A simple volume computation thus yields that

$$2^{O(\alpha)} vO\left(|V(G)|2^{O(\alpha)}\right) \geq \Omega\left(\frac{t}{\alpha}\right)$$

and since $S(n) \geq |V(G)|$

$$S(n) = \Omega\left(\frac{t}{v2^{O(\alpha)}}\right). \qquad \square$$

Remark I. The idea of studying geometric stabbing problems and using volume-based arguments comes from a very recent work of Afshani et al. [2] where an optimal lower bound for rectangle stabbing in d-dimensions was proved. However, there are two major differences: first, their proof techniques are insufficient for us (e.g., they only consider the case $\alpha = 2$) and second, instead of having $Q(n) \leq t$, they build a specific input instance and prove that $Q(n) = \Omega(t \log(S(n)/n))$.

Remark II. It is easy to see that our volume-based argument can be generalized to any reasonable measure. We will not delve into details as we cannot immediately find any algorithmic consequences of such generalizations.

3. Lower Bounds

In this section, we use our framework to prove our claimed lower bounds.

3.1. *Slab Enclosure*

Preliminaries. Let O be the origin. In this article, we use $\mathbb{R}^d$ to denote the d-dimensional Euclidean space. For d real values $A_1, \ldots, A_d$, a hyperplane is the set of points $(X_1, \ldots, X_d) \in \mathbb{R}^d$ that satisfy the equation $X_d = A_d + \sum_{1 \leq i \leq d-1} A_i X_i$ and its dual is the point $(A_1, \ldots, A_d)$. For a real value, τ, we define a *slab with thickness* τ as the region of space between two parallel hyperplanes with distance τ from each other. Consider a slab whose lower boundary is a hyperplane h_1 with equation $X_d = A_d + \sum_{1 \leq i \leq d-1} A_i X_i$ and its upper boundary is a hyperplane h_2 with equation $X_d = A_d + A + \sum_{1 \leq i \leq d-1} A_i X_i$. We call A the *vertical distance* between h_1 and h_2 or the vertical distance of the slab. The dual of this slab is a vertical line segment of length A (note that A is at least τ, the distance between h_1 and h_2).

We define the *slab stabbing problem* as follows: given a set $\mathcal{S}$ of n slabs, store them in a data structure such that the slabs containing a given query point q can be found efficiently. Thus, the dual of the slab stabbing problem is the following: given a set $\overline{\mathcal{S}}$ of vertical line segments, store them in a data structure such that the line segments intersecting a query hyperplane $\overline{q}$ can be found efficiently.

To prove a lower bound for the slab stabbing problem, we use Theorem 1 and thus we need an input set of slabs such that the volume of the intersection of a few of them is small. In [10], Chazelle showed the existence of a set of slabs such that the intersection of any $\log n$ of them has a very small volume. Later with Rosenberg, they used this construction to prove their simple range reporting lower bound [16]. Unfortunately for us, setting $\alpha = \log n$ in Theorem 1 destroys any chance of obtaining a reasonable lower bound (although with some easy tweaks, it is possible to set $\alpha = \varepsilon \log n$ and reclaim Chazelle and Rosenberg's result). In essence, we need a more sensitive lemma.

All of our input slabs have the same thickness 2τ for a parameter τ to be determined later. This means each slab is determined by a single point: for a given point $p \in \mathbb{R}^d$, we define the slab S_p as $\left\{q \in \mathbb{R}^d : |< q, p > -\|p\|^2| \leq \tau \|p\|\right\}$. Let $\mathcal{Q}$ be the d-dimensional hypercube $[1, 2]^d$ and let $\mathcal{Q}^{\text{top}}$ be its top face, i.e., $\mathcal{Q}^{\text{top}} = [1, 2]^{d-1} \times [2]$. Let $\mathcal{P}$ be a set of t points placed on $\mathcal{Q}^{\text{top}}$. For every $p \in \mathcal{P}$, we take the slab S_p and tile $\mathcal{Q}$ using S_p (Figure 1(left and center)): we place disjoint copies of S_p so that they cover all of $\mathcal{Q}$. Let

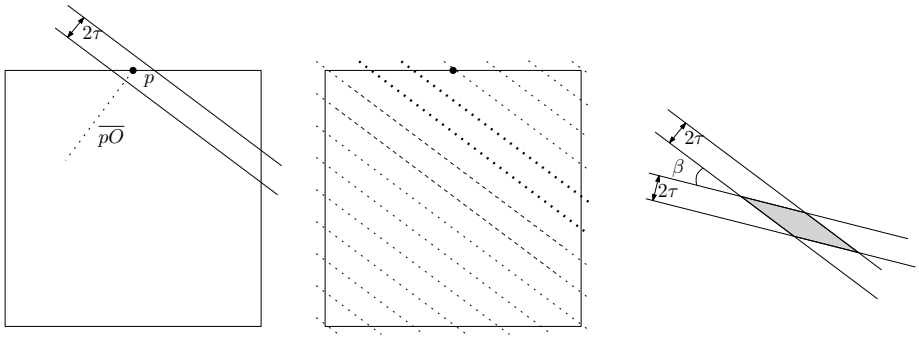

Fig. 1. (left) The slab $S_p = \left\{q \in \mathbb{R}^d : |< q, p > -\|p\|^2| \leq \tau\|p\|\right\}$ is depicted. The thickness (the distance between two defining hyperplanes) is 2τ and both hyperplanes are perpendicular to the line segment that connects p to the origin. (center) Tiling $\mathcal{Q}$ using copies of S_p. (right) The intersection of d slabs forms a parallelotope.

$\mathcal{S}_p$ denote the set of slabs obtained by this tiling. Our final set of slabs $\mathcal{S}$ is defined as $\mathcal{S} = \bigcup_{p\in\mathcal{P}} \mathcal{S}_p$.

The following two lemmas follow closely from the previous work [10,15] although nowhere they have been explicitly listed as lemma. Because of this, and also since we use a different set of parameters, we provide short proofs. The first lemma computes the volume of a parallelotope formed by the intersection of d slabs using elementary techniques.

Lemma 3. [15] *Consider d points $p_1, \ldots, p_d \in \mathcal{P}$ and let $V = \bigcap_{1\leq i\leq d} S_{p_i}$. We have $\mathrm{Vol}(V) = O\left(\tau^d/v_p\right)$ in which v_p is the $(d-1)$-dimensional volume of convex hull of $p_1, \ldots, p_d$.*

Proof. It is easy to see that V is in indeed a parallelotope. First, S_{p_i} is the region between two parallel hyperplanes: $h_i = \left\{x \in \mathbb{R}^d :< x, p_i > -\|p_i\|^2 = -\tau\|p_i\|\right\}$ describes the lower boundary and the upper boundary is described by $h'_i = \left\{x \in \mathbb{R}^d :< x, p_i > -\|p_i\|^2 = \tau\|p_i\|\right\}$. As there are d such slabs (Figure 1(right)) and no two slabs contain parallel hyperplanes, the intersection becomes a parallelotope. The lowest vertex of V, u, satisfies

$$< u, p_i >= \|p_i\|^2 - \tau\|p_i\|, \quad \text{for } 1 \leq i \leq d.$$

For every $1 \leq i \leq d$, there exists another vertex, u_i, adjacent to u. u_i satisfies

$$\begin{cases} < u_i, p_j >= \|p_j\|^2 - \tau\|p_j\| & \text{for } 1 \leq j \leq n, j \neq i \\ < u_i, p_i >= \|p_i\|^2 + \tau\|p_i\|. \end{cases}$$

Note that $\mathrm{Vol}(V) = \det(u_1 - u, u_2 - u, \ldots, u_d - u)$. Observe that $< u_i - u, p_j >= 0$ for $i \neq j$ and $< u_i - u, p_i >= 2\tau\|p_i\|$.

Let A_u and A_p be the matrices in which the i-th row is $u_i - u$, and p_i, respectively. Let A be the diagonal matrix in which $A(i,i) = 2\tau\|p_i\|$. With this notation $\mathrm{Vol}(V) = \det(A_u)$. It is easily verified that $A_u A_p^T = A$. Note that $\det(A) = (2\tau)^d \prod_{1\leq i\leq d}\|p_i\|$. Furthermore, $\det(A_p)/d!$ is the volume the simplex formed by points $p_1, \ldots, p_d$ and the origin; since the points $p_1, \ldots, p_d$ lie on a hyperplane that has distance two from origin, it follows that $\det(A_p)/d! = 2u_p/d$. Thus,

$$\mathrm{Vol}(V) = \frac{(2\tau)^d \prod_{1\leq i\leq d}\|p_i\|}{2(d-1)!u_p} = O\left(\frac{\tau^d}{u_p}\right).$$

□

Lemma 4. [15] *Consider α points $p_1, \ldots, p_\alpha \in \mathcal{P}$ and let $V = \bigcap_{1 \le i \le \alpha} S_{p_i}$. We have $\mathrm{Vol}(V) = O(\tau^d \alpha^{d-1}/v)$ in which v is the $(d-1)$-dimensional volume of convex hull of $p_1, \ldots, p_\alpha$.*

Proof. Since $p_1, \ldots, p_\alpha$ are inside a $(d-1)$-dimensional space, we can triangulate the convex hull of $p_1, \ldots, p_\alpha$ into $O(\alpha^{d-1})$ simplices. We pick the simplex σ with the largest volume and let v_σ be its $(d-1)$-dimensional volume. Thus, $v_\sigma = \Omega(v/\alpha^{d-1})$. V is a subset of the intersection of slabs corresponding to the vertices of σ and thus from the previous lemma we know $\mathrm{Vol}(V) = O(\tau^d/v_\sigma) = O(\tau^d \alpha^{d-1}/v)$. □

The above lemmas imply that a set of slabs have small intersection if and only if the convex hull of the points that give rise to those slabs is large. The next lemma is the main tool that we borrow from [10].

Lemma 5. [10] *Consider the unit hypercube in d dimensions in which d is a constant. For any real value $0 < \rho < 1$, there exists a set $\mathcal{C}_\rho$ of $1/\rho^{O(1)}$ convex shapes inside the unit hypercube such that each shape in $\mathcal{C}_\rho$ has volume $O(\rho)$ and for any convex shape A of volume at most ρ, there exists a convex shape in $\mathcal{C}_\rho$ that completely contains A.*

We are now ready to describe the set $\mathcal{P}$ which defines our set of slabs: $\mathcal{P}$ is chosen by placing t points uniformly at random inside $\mathcal{Q}^{\mathrm{top}}$, for a parameter t, and the thickness τ of the hyperplanes is chosen to be t/n. With this choice of parameters, it is easy to verify that for each point $p \in \mathcal{P}$, we can tile $\mathcal{Q}$ using $\Theta(n/t)$ copies of slab S_p. Remember that $\mathcal{S} = \bigcup_{p \in \mathcal{P}} \mathcal{S}_p$ is our input set. Thus, $|\mathcal{S}| = \Theta(n)$. Clearly, each point of $\mathcal{Q}$ is covered by exactly t slabs. From this point forward, for a region r, we use $\mathrm{Vol}(r)$ to denote the d-dimensional volume of $r \cap \mathcal{Q}$. It is easy to see that for each $r \in \mathcal{S}_p$, $\mathrm{Vol}(r) = O(t/n)$.

We apply Lemma 5 to $\mathcal{Q}^{\mathrm{top}}$ (since $\mathcal{Q}^{\mathrm{top}}$ is a translation of the unit hypercube in the $(d-1)$-dimensional space) with parameter $\rho = (tx)^{-1}$ in which x is a parameter to be adjusted later. For a convex region $C \in \mathcal{C}_\rho$, we bound the probability of the event, E_C, that more than α points are placed inside C, for another parameter α. This probability is

$$\Pr[E_C] \le \binom{t}{\alpha} \mathrm{Vol}(C)^\alpha = O\left((t\rho)^\alpha\right) = O\left(x^{-\alpha}\right)$$

and thus

$$\Pr[\bigvee_{C \in \mathcal{C}_\rho} E_C] \le O\left(\frac{x^{-\alpha}}{\rho^{O(1)}}\right) = O\left(\frac{(tx)^{O(1)}}{x^\alpha}\right).$$

If we set $\alpha = \Theta(\max\{(\log t)/\log x, 1\})$ (with a large enough constant hidden in the Θ notation), the above probability will be less than one and thus there exists an input $\mathcal{P}$ such that the convex hull of any α points in $\mathcal{P}$ has $(d-1)$-dimensional volume greater than $\rho = (tx)^{-1}$. By Lemma 4, it follows that for any α slabs $S_1, \ldots, S_\alpha \in \mathcal{S}$, we have

$$\begin{aligned}\mathrm{Vol}\left(S_1 \cap \cdots \cap S_\alpha\right) &\le \tau^d \alpha^{d-1} tx = \left(\frac{t}{n}\right)^d \alpha^{d-1} tx \\ &\le \left(\frac{t}{n}\right)^d \alpha^{d-1} t 2^{\Theta(\alpha^{-1} \log t)}.\end{aligned}$$

Lemma 6. *Given parameters t and α, there exists a set of $\Theta(n)$ slabs $\mathcal{S}$ in d dimensions such that (i) the thickness of each slab in $\mathcal{S}$ is t/n (ii) every point in $\mathcal{Q}$ is contained in t slabs and (iii) for every α slabs $S_1, \ldots, S_\alpha \in \mathcal{S}$ the volume of $S_1 \cap \cdots \cap S_\alpha$ is $O(t^{d+1} \alpha^{d-1} 2^{O(\alpha^{-1} \log t)}/n^d)$.*

To prove our lower bound, we simply need to use Theorem 1 and adjust the parameters.

Theorem 2. *Answering d-dimensional slab stabbing queries in $Q(n) + O(k)$ time requires $\Omega\left((n/Q(n))^d\, 2^{-O(\sqrt{\log Q(n)})}\right)$ space.*

Proof. We pick $t = Q(n)$ in Lemma 6. Combined by Theorem 1, we obtain that

$$S(n) = \Omega\left(Q(n)\frac{n^d}{Q(n)^{d+1}\alpha^{d-1}2^{O(\alpha^{-1}\log t)}}\frac{1}{2^{O(\alpha)}}\right).$$

We set $\alpha = \sqrt{\log Q(n)}$. Observe that $\alpha^{d-1}2^{O(\alpha^{-1}\log t)}2^{O(\alpha)} = (\log Q(n))^{O(d)}2^{O\left(\sqrt{\log Q(n)}\right)} = 2^{O\left(\sqrt{\log Q(n)}\right)}$ since d is assumed to be a constant. Thus,

$$S(n) = \Omega\left(\left(\frac{n}{Q(n)}\right)^d 2^{-O\left(\sqrt{\log Q(n)}\right)}\right).$$

□

3.2. *Simplex Range Reporting Lower Bound*

In this section we prove our lower bound for simplex range reporting. We use our result on slab stabbing problem. Unfortunately, we are unable to give a general reduction from simplex range reporting to the slab stabbing problem. However, we can do it for some special cases which include the instances that we used to obtain our slab stabbing lower bound.

Lemma 7. *Consider an input $\mathcal{S}$ for the slab stabbing problem. Let d_{min} and d_{max} be the minimum and the maximum vertical distance of slabs in $\mathcal{S}$. If $d_{max}/d_{min} = O(1)$, then the slab stabbing problem can be solved using a data structure for simplex range reporting with no asymptotical increase in query or space complexities.*

Proof. For an integer i, $0 \le i \le \log(d_{\max}/d_{\min})$, let $\mathcal{S}_i$ be the subset of $\mathcal{S}$ that contains slabs with vertical distance of at least $2^i d_{\min}$ and less than $2^{i+1}d_{\min}$. Let H_i^{low} and H_i^{high} be the set of hyperplanes that form the lower boundary and the upper boundary of the slabs in $\mathcal{S}_i$ respectively. We build simplex range reporting data structures one for each set of points dual to H_i^{low} and H_i^{high} and for each $0 \le i \le \log(d_{\max}/d_{\min})$.

Let q be the query point. Consider an index $i, 0 \le i \le \log(d_{\max}/d_{\min})$. Let q^{high} (resp. q^{low}) be the point obtained from q by adding (resp. subtracting) $2^i d_{\min}$ to the d-th coordinate of q. Using simplex range reporting data structures, we find all the hyperplanes in H_i^{low} that pass between q and q_{low} (in primal space this translates to finding all the points between two query hyperplanes). Similarly, we find all the hyperplanes in H_i^{high} that pass between q and q^{high}. We output the slabs corresponding to the hyperplanes found. Since $\log d_{\max}/d_{\min} = O(1)$, we only impose a constant factor overhead on the query time and space complexity of the simplex range reporting data structure used.

To see the correctness, consider a slab $S \in \mathcal{S}_i$ and let A be the vertical distance of S. As $A < 2^{i+1}d_{\min}$, if S contains q, then either the lower boundary of S will pass above q^{low} or its upper boundary will pass below q^{high} and thus it will be reported. However, since $A \ge 2^i d_{\min}$, if S does not contain q, then the lower boundary of S will not pass between q and q^{low} and the upper boundary of S will not pass between q and q^{high} as well. Thus, S will not be reported if it does not contain q. □

Thus, we simply need to bound the vertical distance of the slabs that we built in the previous section.

Theorem 3. *Any structure that answers d-dimensional simplex range reporting queries in $Q(n) + O(k)$ time must use $\Omega\left((n/Q(n))^d\, 2^{-O(\sqrt{\log Q(n)})}\right)$ space.*

Proof. Consider a slab S_p with its lower boundary being $h = \{x \in \mathbb{R}^d :< x, p > -\|p\|^2 = -\tau\|p\|\}$ and its upper boundary being $h' = \{x \in \mathbb{R}^d :< x, p > -\|p\|^2 = \tau\|p\|\}$. Let x and $x + v$ be two points on lower and upper boundary respectively in which v is a vertical vector, i.e., $v = (0, 0, \ldots, 0, A)$ where A is the vertical distance of S_p. We have

$$< x, p > -\|p\|^2 = -\tau\|p\| \quad \text{and} \quad < x + v, p > -\|p\|^2 = \tau\|p\|.$$

As p is chosen on Q^{top}, its d-th coordinate is two. Thus,

$$2A =< v, p >= 2\tau\|p\| = \Theta(\tau).$$

As all the slabs we constructed in the previous section had the same thickness, the theorem follows from Lemma 7 and Theorem 2. □

3.2.1. *Halfspace Range Reporting Lower Bounds*

Currently the best known lower bounds for halfspace range reporting problem are obtained through reductions from the slab stabbing problem (or its dual) via a simple geometric transformation: Consider a slab S with hyperplanes h_1 and h_2 being its lower and upper boundaries respectively. Let $X_d = A_d + \sum_{1 \le i \le d-1} A_i X_i$ and $X_d = A_d + A + \sum_{1 \le i \le d-1} A_i X_i$ be equations describing h_1 and h_2 respectively. S contains points $q = (q_1, \ldots, q_d)$ such that $q_d \ge A_d + \sum_{1 \le i \le d-1} A_i q_i$ and $q_d \le A_d + A + \sum_{1 \le i \le d-1} A_i q_i$ and thus $|q_d - A_d - A/2 - \sum_{1 \le i \le d-1} A_i q_i| \le A/2$. Thus, $(q_d - A_d - A/2 - \sum_{1 \le i \le d-1} A_i q_i)^2 \le A^2/4$. By introducing some new variables, one for each q_i^2, $1 \le i \le d$, and one for each $q_i q_j$, $1 \le i < j \le d$, this can be turned into a linear inequality in $d(d+3)/2$ variables, a halfspace in $(d(d+3)/2)$-dimensional space. Thus, slab enclosure can be solved with a $(d(d+3)/2)$-dimensional halfspace range reporting structure.

Theorem 4. *Any data structure that answers $(d(d+3)/2)$-dimensional halfspace range reporting queries in $Q(n) + O(k)$ time requires $\Omega\left((n/Q(n))^d\, 2^{-O(\sqrt{\log Q(n)})}\right)$ space.*

4. I/O Model Lower Bounds

The I/O model is a widely used model of computation that deals with massive data. In this model, data sits on a disk of conceptually infinite size which is divided into blocks and each block can store up to B records. All computation must be performed in a main memory of size M. By performing an I/O, it is possible to transfer one block between the main memory and the disk and the goal is to minimize number of I/Os performed by the algorithm.

Similar to the indexibility model [19], the crucial restriction we impose on the query algorithm is the *indivisibility of the records* (the *indivisibility assumption* for short) which regards the records atomic and unbreakable: each block can store up to B records and to output a record a block storing the record should be loaded into the main memory. We allow all the other informations be stored and accessed for free.

To be more precise, consider an abstract reporting problem in which queries output subsets of an input set $\mathcal{S}$ containing n elements. Under the indivisibility assumption, if the data structure uses $S(n)$ space, then it consists of $S(n)/B$ blocks such that each block stores up to B elements of $\mathcal{S}$. To answer a query, the data structure can choose m blocks and load them to the main memory and output a subset of the elements stored in those blocks. The cost of answering the query is m. In many problems, the goal is to achieve a query cost of the form $Q(n) + O(k/B)$ in which k is the output size.

Previously, only one lower bound framework was used for reporting problems. The heart of this framework is the Redundancy Theorem [19] and its refinement [6] that ties the space overhead of the data structure to a combinatorial structure of the query set, in a very similar manner to Chazelle's framework.

Here we use our new techniques to provide another lower bound framework. A rather non-trivial and counter-intuitive consequence of our framework is that to build a difficult input instance in the I/O model, we simply need to find a difficult input instance in the pointer machine model and duplicate each input element a number of times.

Theorem 5. *Assume that there exists an input $\mathcal{S}$ of n regions for a geometric stabbing problem such that (i) every point of $\mathcal{Q}$ is contained in exactly t regions and (ii) the volume of the intersection of every α regions is at most v in which $\alpha < t$ and t are two parameters. Then, there exists an input $\mathcal{S}'$ of $N = \Theta(nB/\alpha)$ regions such that any external memory data structure that answers geometric stabbing queries on $\mathcal{S}'$ using $Q(N) + O(k/B)$ I/Os, for $Q(N) = O(t/\alpha)$, consumes $B\Omega(tv^{-1}/2^{O(\alpha)}) = B\Omega(Q(N)v^{-1}/2^{O(\alpha)})$ space.*

Remark. The easy way to generalize Theorem 1 yields a much less effective lower bound that is a factor of B smaller. So the fact that we build another input instance using $\mathcal{S}$ is crucial.

The rest of this subsection is devoted to the proof of the above theorem.

$\mathcal{S}'$ is made by replicating every region in $\mathcal{S}$ β times, for a parameter β. We set $\beta = \Theta(B/\alpha)$ but its exact value will be determined later. For every $r \in \mathcal{S}$, we create β identical copies of r and place them at the same location as r (think of it as duplicating r $\beta - 1$ times). $\mathcal{S}'$ is the set of all these copies. We denote the size of $\mathcal{S}'$ by N and thus $N = \beta n$. We say two regions $r, r' \in \mathcal{S}'$ are *different* if they are not copies of the same region in $\mathcal{S}$. Now consider a data structure that stores $\mathcal{S}'$ using $S(N)$ space that can answer geometric stabbing queries using $Q(N) + O(k/B)$ I/Os.

We call a tuple of α elements a *hub (of size α)* if they are all different and there exists a block that stores all of them. Since the data structure contains $S(N)/B$ blocks, it follows that the total number of hubs is at most $\binom{B}{\alpha}S(N)/B$.

Now consider a particular query. Note that every point of $\mathcal{Q}$ is contained in $t\beta$ regions of $\mathcal{S}'$ and thus the output size is $t\beta$. A hub (of size α) whose all elements are outputted at query time is called an *output hub (of size α)*. The query algorithm reads $Q(N)+O(k/B)$ blocks and since $k = \beta t$ and $Q(N) = O(t/\alpha)$, it follows that $Q(N)+O(k/B) = O(k/B)$. It is easy to show that there are at least $k/(2B)$ blocks that each outputs at least k_b elements and $k_b = \Theta(B)$; we call these blocks *heavy*. We pick $\beta = k_b/\alpha = \Theta(B/\alpha)$ (for simplicity assume β is an integer). We now lower bound the number of output hubs.

Lemma 8. *Every heavy block* B *has at least β^α output hubs of size α.*

Proof. Let A be a set of k_b regions that are reported from B. Let x be the maximum number of pairwise different regions in A. We can decompose A into x equivalent classes $A_1, \ldots, A_x$ in which the regions in each A_i are copies of the same region. Note that since $|A| = k_b = \beta\alpha$ we have $x \geq \alpha$. Let X be the number of output hubs of size α and let $X_{i,0}$ be the number of output hubs of size i that have no element from A_1 or A_2. Each output hub has either zero or one element from A_1 or A_2 and thus we have

$$X = X_{\alpha,0} + (|A_1| + |A_2|)X_{\alpha-1,0} + |A_1||A_2|X_{\alpha-2,0}.$$

If we keep $|A_1| + |A_2|$ fixed, then X is minimized when the difference between $|A_1|$ and $|A_2|$ is maximized. Thus, the minimum of X is achieved when $x = \alpha$ and $|A_i| = \beta$ for $1 \leq i \leq \alpha$. And in that case we have $X = \beta^\alpha$. □

For a hub H to be used as an output hub, the query must be inside the geometric region formed by the intersection of all the α regions that form H. Since the regions that form a hub are different, the intersection has volume at most v. Thus, the total volume of the geometric regions of all the hubs in the data structure is $\binom{B}{\alpha}vS(n)/B$. However, as we saw, to answer any query we must access $k/(2B)$ heavy blocks and each heavy block has β^α output hubs. Thus, the total number of output hubs for any query is at

least $\beta^\alpha k/(2B)$. Each hub can only produce one output hub. Since every point of $\mathcal{Q}$ can be used as a query, the geometric regions of the hubs must cover $\mathcal{Q}$ at least $\beta^\alpha k/(2B)$ times. We must have

$$\begin{aligned}\frac{\binom{B}{\alpha} v S(n)}{B} &\geq \beta^\alpha \frac{k}{2B} = \beta^\alpha \frac{t\beta}{2B} = \beta^\alpha \frac{\Omega(Q(N)\alpha)\beta}{2B} \\ &= \Omega(\beta^\alpha Q(N)) = O\left(\frac{(B/\alpha)^\alpha Q(N)}{2^{O(\alpha)}}\right)\end{aligned}$$

and thus

$$\frac{S(N)}{B} = \Omega\left(\frac{(B/\alpha)^\alpha Q(N)}{\binom{B}{\alpha} v 2^{O(\alpha)}}\right) = \Omega\left(\frac{Q(N)}{v 2^{O(\alpha)}}\right).$$

Using Lemma 6 and setting $\alpha = \sqrt{\log Q(N)}$, we obtain the following theorem.

Theorem 6. *In the I/O model, any structure that answers d-dimensional simplex range reporting or d-dimensional slab stabbing queries, or $(d(d+3)/2)$-dimensional halfspace range reporting queries with $Q(N) + O(k/B)$ I/Os must use $\Omega\left(BN^d(BQ(N))^{-d}2^{-O(\sqrt{\log Q(N)})}\right)$ space.*

5. Conclusions

In this paper, we presented a number of improved lower bounds for simplex range reporting, slab stabbing, and halfspace range reporting in the pointer machine. We also presented new lower bounds for the problems in the I/O model of computation. To obtain our lower bounds, we built new frameworks for proving lower bounds in both models of computation. Our framework has the advantage that given a set of geometric ranges, it can directly give a lower bound, provided the ranges have small intersection properties. As we saw, the previous papers required such set of ranges as well as additional tools that sometimes would result in difficult to solve technical problems.

An interesting aspect of our frameworks is that building a single input instance is enough to prove good lower bounds in both models of computation. This is unlike the previous attempts where obtaining I/O lower bounds always required tweaking the internal memory constructions in non-trivial ways. For instance, for orthogonal range reporting, the pointer machine lower bounds were obtained as early as 1990 [11] but the equivalent I/O lower bounds were obtained almost two decades later [1].

At the end, we must mention that there are still very interesting questions that are left open. Recently, an important open problem that is directly related to our work came from Timothy Chan [8] who asked whether logarithmic factor increases are necessary in multi-level partition trees. For instance, we can now ask whether it is possible to prove a space lower bound of $\Omega((n/Q(n))^d \log n)$ for data structures that answer simplex-simplex containment queries (preprocess a set of simplices such that the simplices contained in a query simplex can be outputted efficiently) in $O(Q(n) + k)$ time for polylogarithmic values of $Q(n)$.

6. Acknowledgement

The author thanks the anonymous reviewers for excellent suggestions that greatly improved the presentation of the paper.

References

1. 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.

2. P. Afshani, L. Arge, and K. G. Larsen. Higher-dimensional orthogonal range reporting and rectangle stabbing in the pointer machine model. In *SCG '12: Proc. of the 26th Annual Symposium on Computational Geometry*, pages 323–332, 2012.
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, L. Arge, J. Erickson, P. G. Franciosa, and J. S. Vitter. Efficient searching with linear constraints. *J. Comput. Syst. Sci.*, 61:194–216, 2000.
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. L. Arge, V. Samoladas, and K. Yi. Optimal external memory planar point enclosure. *Algorithmica*, 54(3):337–352, 2009.
7. S. Arya, D. M. Mount, and J. Xia. Tight lower bounds for halfspace range searching. *Discrete and Computational Geometry*, 47(4):711–730, 2012.
8. T. M. Chan. Optimal partition trees. In *SCG '10: Proc. of the 26th Annual Symposium on Computational Geometry*, pages 1–10. ACM, 2010.
9. B. Chazelle. Filtering search: a new approach to query answering. *SIAM Journal on Computing*, 15(3):703–724, 1986.
10. B. Chazelle. Lower bounds on the complexity of polytope range searching. *Journal of the American Mathematical Society*, 2:637–666, 1989.
11. B. Chazelle. Lower bounds for orthogonal range searching: I. the reporting case. *Journal of the ACM*, 37(2):200–212, 1990.
12. B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete and Computational Geometry*, 9(2):145–158, 1993.
13. B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10:229–249, 1990.
14. B. Chazelle and D. Liu. Lower bounds for intersection searching and fractional cascading in higher dimension. *Journal of Computer and System Sciences*, 68(2):269 – 284, 2004.
15. B. Chazelle and B. Rosenberg. Simplex range reporting on a pointer machine. *Computational Geometry*, 5(5):237 – 247, 1996.
16. B. Chazelle, M. Sharir, and E. Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Algorithmica*, 8:407–429, December 1992.
17. K. Clarkson. New applications of random sampling in computational geometry. *Discrete and Computational Geometry*, 2:195–222, 1987.
18. J. Erickson. Space-time tradeoffs for emptiness queries. *SIAM Journal on Computing*, 29(6):1968–1996, 2000.
19. J. M. Hellerstein, E. Koutsoupias, D. P. Miranker, C. H. Papadimitriou, and V. Samoladas. On a model of indexability and its bounds for range queries. *Journal of the ACM*, 49(1):35–55, 2002.
20. J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete and Computational Geometry*, 10(2):157–182, 1993.
21. J. Matoušek. Efficient partition trees. *Discrete and Computational Geometry*, 8(3):315–334, 1992.
22. E. Welzl. On spanning trees with low crossing numbers. In B. Monien and T. Ottmann, editors, *Data structures and efficient algorithms*, volume 594 of *Lecture Notes in Computer Science*, pages 233–249. Springer Berlin / Heidelberg, 1992.
23. D. E. Willard. Polygon retrieval. *SIAM Journal on Computing*, 11:149–165, 1982.