A Geometric Approach to Lower Bounds for Approximate Near-Neighbor Search and Partial Match

Rina Panigrahy
Microsoft Research
rina@microsoft.com

Kunal Talwar
Microsoft Research
kunal@microsoft.com

Udi Wieder
Microsoft Research
uwieder@microsoft.com

Abstract

This work investigates a geometric approach to proving cell probe lower bounds for data structure problems. We consider the *approximate nearest neighbor search problem* on the Boolean hypercube $(\{0,1\}^d, \|\cdot\|_1)$ with $d = \Theta(\log n)$. We show that any (randomized) data structure for the problem that answers c-approximate nearest neighbor search queries using t probes must use space at least $n^{1+\Omega(1/ct)}$. In particular, our bound implies that any data structure that uses space $\tilde{O}(n)$ with polylogarithmic word size, and with constant probability gives a constant approximation to nearest neighbor search queries must be probed $\Omega(\log n/\log\log n)$ times. This improves on the lower bound of $\Omega(\log\log d/\log\log\log d)$ probes shown by Chakrabarti and Regev [8] for any polynomial space data structure, and the $\Omega(\log\log d)$ lower bound in Pătraşcu and Thorup [26] for linear space data structures.

Our lower bound holds for the *near neighbor problem*, where the algorithm knows in advance a good approximation to the distance to the nearest neighbor.

Additionally, it is an *average case* lower bound for the natural distribution for the problem. Our approach also gives the same bound for $(2-\frac{1}{c})$-approximation to the farthest neighbor problem.

For the case of non-adaptive algorithms we can improve the bound slightly and show a $\Omega(\log n)$ lower bound on the time complexity of data structures with $O(n)$ space and logarithmic word size.

We also show similar lower bounds for the partial match problem: any randomized t-probe data structure that solves the partial match problem on $\{0,1,\star\}^d$ for $d = \Theta(\log n)$ must use space $n^{1+\Omega(1/t)}$. This implies an $\Omega(\log n/\log\log n)$ lower bound for time complexity of near linear space data structures, slightly improving the $\Omega(\log n/(\log\log n)^2)$ lower bound from [25],[16] for this range of d. Recently and independently Pătraşcu achieved similar bounds [24]. Our results also generalize to approximate partial match, improving on the bounds of [4, 25].

1 Introduction

Given a dataset of n points, the goal in the Nearest Neighbor Problem is to build a data structure such that given a query point, its nearest neighbor in the dataset can be retrieved quickly. Typically, the dataset and the query points are represented by vectors in a normed space, such as $\mathbb{R}^d$ equipped with the ℓ_1 or the ℓ_2 norm. Nearest Neighbor Search is a fundamental problem in data structures with numerous applications to web algorithms, computational biology, information retrieval, machine learning, etc. As such it has been researched extensively.

Exact algorithms for this problem suffer from the "curse of dimensionality", i.e. the query time and/or the space requirements of the best-known data structures have an exponential dependence on d, making these algorithms infeasible when the dimension d is not small. This motivates allowing for approximation. The goal in the *c-approximate nearest neighbor search* problem is to return a point in the dataset whose distance to the query point is no larger than c times the distance to the nearest neighbor. Since in many applications, the representation of the original objects as vectors is already lossy, this is acceptable. Additionally, the nearest neighbor is most useful when it is much closer to the query than the other dataset points; in this case an approximate nearest neighbor query would return the nearest neighbor itself. Indyk and Motwani [15], and Kushilevitz, Ostrovsky and Rabani [19] independently gave polynomial algorithms for approximate nearest neighbor search in high dimensions. Their approach is referred to as Locality Sensitive Hashing (LSH) and entails hashing the data points into an array such that nearby points are likely to hash into the same location and distant points are likely to hash into different locations. Their approach was further refined and applied in a large number of papers (c.f. [2],[12],[29],[10]). We remark that all these approaches are randomized. Moreover, they reduce the problem to the easier *approximate near-neighbor problem*, where given the query and a distance estimate λ, the goal is to find a point in the dataset whose distance is at most $c\lambda$, if the nearest neighbor is at distance less than λ.

The partial match problem is a close relative of the nearest neighbor problem. The dataset consists of n points from (say) $\{0,1\}^d$ as in the nearest neighbor problem. The query q is a vector from $\{0,1,\star\}^d$, and the goal is to find a point in the dataset, if any, that *matches* q, where a 0 (1) matches a 0 (1) and a $\star$ can match either a 0 or a 1. This problem is also believed to be harder than the nearest neighbor problem.

In this work we prove lower bounds for these problems in the *Cell Probe* model of Yao [30], which is a natural model for the case where data is stored in a random access memory. In this model the data structure is a static table (or distribution of tables) populated based on the dataset. Given a query point, a (possibly randomized, possibly adaptive) algorithm probes a subset of the entries in the data structure and outputs the result. Typically one studies the tradeoff between the size of the data structure, and the number of probes to the data structure needed to perform a query. The bounds proven in this setting are information-theoretic as all computation is free.

1.1 Related Work

Chakrabarti *et al.* [7] and Borodin, Ostrovsky and Rabani [6] were the first to prove lower bounds for the nearest neighbor problem, though the former only allowed for deterministic algorithms, and the latter only allowed for exact nearest neighbor search. Subsequent improvements by Liu [20] and Barkol and Rabani [4] also suffer from one of these shortcomings. In contrast, the aforementioned algorithms for the problem are both randomized and approximate. These lower bounds actually apply to the near-neighbor search problem, the approximate version of which has a constant-probe randomized data structure that uses polynomial space; thus these limitations are inherent in those approaches. Chakrabarti and Regev [8] were the first to address this shortcoming: they showed that any randomized cell probe algorithm with

$poly(n, d)$ words of size $poly(d)$ that answers approximate *nearest* neighbor queries on $\{0,1\}^d$ must make $\Omega(\log\log d/\log\log\log d)$ queries. They also show that this bound is tight by providing a matching upper-bound.

These lower bounds are very strong in that they hold for any polynomial storage data structure. However, this generality also precludes better bounds that could be proved under a more stringent and reasonable space constraint. All these bounds are proven by a reduction to lower bounds for asymmetric communication complexity, an approach pioneered by Miltersen *et al.* [21] and Ajtai [1]. However, this reduction is lossy for large t (see Section 6). Additionally, polynomial differences (e.g. space n vs. n^{20}) in the size of the data structure translate to constant multiplicative differences (e.g. $\log n$ vs. $20\log n$) in the number of bits sent by Alice, making it difficult to prove better bounds for small polynomial, or near-linear data structures (see Gal and Miltersen [14] for a discussion). Specifically for randomized approximate nearest neighbor search, Chakrabarti and Regev [8] show that a common communication complexity technique called 'richness' cannot yield any non-trivial bound. In fact, e.g., for $d = 10\log n$, there is an easy 10 round protocol where Alice sends only $\log n$ bits per round (Alice can simply send the query point q), so that the communication complexity approach must fail.

Pătraşcu and Thorup [25] got around this difficulty using a direct sum theorem for richness and showed that any data structure for approximate nearest neighbor, using space S must use $\Omega(d/\log\frac{Sd}{n})$ probes, which rearranges to a bound very similar to ours for $d = \Theta(\log n)$. However, as other previous work, their lower bound applies only to deterministic or to exact algorithms, and being based on richness, their approach cannot work for large constant approximation factors. When allowing for both randomization and approximation, Andoni, Indyk and Pătraşcu [3] show that for small $\beta > 0$, any $(\frac{1}{\beta^2})$-probe algorithm for $(1+\beta)$-approximate near neighbor problem must use space $n^{\Omega(\frac{1}{\beta^2})}$. This bound is tight for small enough $\beta > 0$ [3]. More recently, Pătraşcu and Thorup [26] claim that any randomized data structure for approximate nearest neighbor using space $\tilde{O}(n)$ requires $\Omega(\log\log d)$ queries[1].

Lower bounds for the partial match problem have also been heavily studied [21, 6, 4, 16, 25]. Jayram *et al.* [16] show that any polynomial space data structure must make $\Omega(d/\log^2 n)$ probes, while Pătraşcu and Thorup [25] show a lower bound of $\Omega(\frac{d}{\log d}/\log\frac{Sd}{n})$ probes for space S data structures. The latter bound translates to $\Omega(\log n/(\log\log n)^2)$ for space $\tilde{O}(n)$ and a logarithmic dimension. Recently and independently of our work Pătraşcu [24] improved the bound to match ours at $\Omega(\log n/\log\log n)$. These bounds however hold only when $d \in \omega(\log n)$, while our bound holds when $d \in O(\log n)$. Upper bounds have been studied by Rivest [27, 28] and more recently by Charikar *et al.* [9]. Barkol and Rabani [4] study approximate partial match and show a lower bound similar to their lower bound for exact nearest neighbor. Combined with [25] this translates to a lower bound similar to ours. As before, these bounds hold only when $d \in \omega(\log n)$, while our bound holds when $d \in O(\log n)$. Lower bounds for farthest neighbor have been studied by Andoni, Indyk and Pătraşcu [3] where they show a $n^{\Omega(1/\epsilon^2)}$ lower bound for $(1+\epsilon)$-approximate farthest neighbor approximation for constant probe data structures.

Restricted Models:

Stronger lower bounds were shown for restricted models of computation. Beame and Vee [5] use time-space tradeoffs for branching programs to prove lower bounds of $\Omega(d\sqrt{\log d/\log\log d})$ (or even $\Omega(d\log d)$) on the number of probes for deterministic polynomial sized data structures assuming that the query algorithm accesses the query bits in specific ways and uses limited additional storage.

The starting point of this work is a paper by Motwani, Naor and Panigrahy [22] which provided lower bounds for Locality Sensitive Hashing (LSH) schemes. An (λ, c, p, q) LSH scheme is a distribution of hash

[1]The claim in [26] is without proof

functions from a metric space to an array, so that the probability that two points of distance at most λ hash into the same location is at least p and the probability two points of distance at least $c\lambda$ hash to the same location is at most q. In [22] it is shown that if the metric space is the hypercube with the hamming distance then $\frac{\log(1/p)}{\log(1/q)} \geq \frac{1}{2c}$. In particular, this implies that for linear space, the time complexity of any LSH-based algorithm is at least $n^{\Omega(c)}$. The tightness of this bound follows from [23, 3]. The main component of the proof in [22] is an isoperimetric bound, showing that the probed memory location is *sensitive* to small perturbations of the query point. This intuition lies at the heart of our proof as well.

1.2 Our Contributions

In this work we use geometric arguments to prove a strong lower bound for c-approximate near neighbor search on the Boolean hypercube $(\{0,1\}^d, \|\cdot\|_1)$, for $d = \Omega(\log n)$. We show that any (randomized) t-probe data structure for the problem that succeeds with constant probability must use space $n^{1+\Omega(1/ct)}$. In particular, this implies that any data structure that uses space $\tilde{O}(n)$ with polylogarithmic word size, and gives a constant approximation to nearest neighbor search queries must probe the data structure at least $\Omega(\log n/\log\log n)$ times[2]. Our result applies to a more general setting, where arbitrary shared randomness between the data structure and query algorithm can be accessed for free. Additionally, it is an *average-case* lower bound for the natural distribution; to our knowledge, this is the first average case lower bound for the near neighbor problem.

Our results also imply a similar lower bound for the $(2-\frac{1}{c})$-approximate *far neighbor* problem. We also show similar lower bounds for the partial match problem: any t probe data structure for the partial match problem in $O(\log n)$ dimensions must use space $n^{1+\Omega(\frac{1}{t})}$. This implies a lower bound of $\Omega(\log n/\log\log n)$ on the number of probes for any $\tilde{O}(n)$ space data structure. Our bounds extend also to the approximate partial match problem.

As mentioned above, we diverge from previous work in bypassing asymmetric communication complexity approaches. We first use analytic techniques to prove an isoperimetric-type inequality that implies a lower bound for single probe algorithms[3]: we show that the different query points in the neighborhood of any one data point p are likely to probe many different cells in the data structure (so that, intuitively, the information about p must be present in many cells). This argument leads to a lower bound of the form $(Space/n) \geq n^{\Omega(\frac{1}{c})}$. We then show a novel reduction that converts such isoperimetry-based single probe lower bounds to cell probe lower bounds of the form $(Space/n)^t \geq n^{\Omega(\frac{1}{c})}$ for t-probe data structures. Our results imply bounds of the form $(Space/n)^t \geq n^{\Omega(\frac{1}{c^2})}$ for $(\{0,1\}^d, \|\cdot\|_2)$, and hence for Euclidean space.

Non-adaptive algorithms are algorithms where the memory locations which are probed are a function of the query point only, and not a function of the table content. We note that Locality Sensitive Hashing is a non-adaptive approach. In fact, utilizing the power of adaptivity is an interesting open problem. For non-adaptive algorithms we can improve our bound slightly and show that for $O(n)$ space and logarithmic dimension and word length, the time complexity is $\Omega(\log n)$. We also show an interesting connection between non-adaptive algorithms and locally decodable codes (see Section 4.6).

[2]For randomized, approximate near neighbor, d is $O(\log n)$ w.l.o.g., by dimension reduction arguments [17].

[3]Our lower bounds are for the search version; our instances are all YES instances of the natural decision problem.

2 Preliminaries

Approximate Near Neighbor Search (ANNS): In this work we look at the (c, λ)-approximate near-neighbor problem over the d-dimensional hypercube. Let $c, \lambda > 0$ be fixed. We are given n points $p_1, \ldots, p_n \in \{0,1\}^d$ to preprocess into a data structure D. Then given a query $q \in \{0,1\}$, the algorithm must consult the data structure D and output a $z \in \{p_i : i \in [n]\}$. The constraint is that if for some i, $d(q, p_i) \leq \lambda$, then $d(q, z) \leq c\lambda$. In particular, if p_i is within distance λ of q, and no other p_j is within $c\lambda$ of q, then the algorithm must output p_i. Note that any c-approximate nearest neighbor algorithm implies an (c, λ)-approximate near-neighbor algorithm.

Partial Match Problem: The algorithm must once again preprocess n points $p_1, \ldots, p_n \in \{0,1\}^d$ and build a data structure D. Then given a query point $q \in \{0, 1, \star\}^d$, the algorithm must output a $z \in \{p_i : i \in [n]\}$. The constraint now is that if there is a p_i such that q agrees with p_i in all entries different from $\star$, then z must satisfy this property as well. In the *approximate* partial match problem z and q are allowed to disagree on a small number of entries.

Cell Probe Complexity: The complexity of the algorithm will be measured by the size of the data structure D, and the number of accesses to D at query time. More precisely, we will assume that the data structure D has m cells holding w bits each. At query time, the (possibly randomized, adaptive) algorithm accesses t cells of D. Based on these accesses to D, it must then output an answer z. Note that there are no computational constraints on the algorithm.

Relaxed Cell Probe model: We prove our lower bound for a slightly relaxed version of the problem. Both the preprocessing and the query algorithm are given free access to a shared source of randomness. Given the points $p_1, \ldots, p_n$, the preprocessing algorithm is allowed to construct t tables $D_1, \ldots, D_t$, each consisting of m cells of width w bits each. When presented with the query, the algorithm first consults a cell in D_1. Based on the result, it then makes its second probe to D_2, and so on. Thus it accesses at most one cell in each of the D_j's. The parameters of interest are the number of probes t, the number of cells per table m, and the word size w. Clearly, the usual cell probe complexity setting is a special case where all D_j's are identical.

3 Overview of our methods

Consider a random instance of the ANNS problem consisting of n random points in a d-dimensional hypercube (with $d = \Omega(\log n)$). The query point is chosen randomly by flipping each bit of one of these points with probability ϵ (which can be thought of as picking a random point from a ball of radius ϵd around one of these points). This gives n balls as candidate query points. Any successful algorithm, when presented with such a query point must be able to output the center of the ball which is the nearest neighbor.

Viewing the algorithm execution as a sequence of table accesses: An algorithm makes a sequence of memory accesses into tables $D_1, \ldots, D_t$. The contents of these tables depends only on the set of n points in the dataset. When posed with a query point, the execution of the algorithm may be abstracted by a sequence of functions $F_1, F_2, .., F_t$ that are used to decide the locations of the memory accesses into these tables. If the algorithm is successful in reconstructing the center of the ball, an added lookup F_{t+1} to $h(p)$, for a hash function h, ensures that the last lookup maps almost all of the n balls around the points to distinct values (see Lemma 4.2).

Relation to the geometric structure of the hypercube: To develop an understanding of the core complexity of the problem, consider the case $t = 0$ when there is a single function F. Success in this case means that F maps each of the n balls to distinct values. Note that since the function F is independent of the points in the dataset, it results in a partition of the hypercube into n regions based on the output value of F. Such a restricted class of functions is known as a locality sensitive hash functions (LSH).

Any LSH has high spread: It is known [22] that there is no such LSH that maps points in the hypercube to n values that concentrates each ball entirely into one value. In fact for any LSH, we show that a ball with a random center will be shattered into $n^{\Omega(\epsilon)}$ parts. This is proven using the geometric properties of the hypercube (in Section 4.2 using the Hypercontractive inequality). Thus if we have a table of size n then for any function F that takes the query point as input to look up this table, F must spread a ball into many different table locations.

In this work we generalize this result to more than one, possibly adaptive, memory accesses. This is the technical crux of the paper (Section 4.3). For a given population of the tables, the algorithm can be viewed as a function of the query point. The LSH bound above implies that any fixed table population is unlikely to be good; and thus the expected fraction of table populations that are good is small. On the other hand, we show that one good table population can be perturbed to construct a very large number of good table populations.

4 Lower bound for ANNS

Let $\epsilon = \frac{1}{8c}$. Our input space is the boolean hypercube $\{0,1\}^d$, for an arbitrary $d \geq 10 \log n$. For any given point $y \in \{0,1\}^d$ let $\mu_{y,\epsilon}$ be the distribution over $\{0,1\}^d$ obtained by flipping each bit of y independently with probability ϵ. One can think of $\mu_{y,\epsilon}$ as essentially being uniform on a ball of radius ϵd around y. Given a set $A \subset \{0,1\}^d$ we will use the notation $\mu_{y,\epsilon}(A)$ to denote the probability that a random point from the distribution $\mu_{y,\epsilon}$ lies in A. Where ϵ is fixed, we abbreviate $\mu_{y,\epsilon}$ by μ_y.

The input distribution $\mathcal{D}$ is as follows: The dataset is drawn by picking n points $p_1, p_2, \ldots, p_n$ uniformly and independently at random from $\{0,1\}^d$. The query point q is chosen by picking a random $i \in [n]$ and then sampling q from the distribution $\mu_{p_i,\epsilon}$.

An (r, m, w)-data structure D consists of r tables, where each table has m rows storing w-bit words. Thus $D[j]$ is an array consisting of m words; we will abuse notation and refer to it as D_j when convenient.

We consider algorithms that make r (possibly adaptive) queries, where the jth query is made to a location $l_j \in [m]$ in table D_j, where $l_j = l_j(q) = l_j(q, D)$ may depend both on the query point q and on the information learnt from the first $(j-1)$ queries $D_1[l_1], \ldots, D_{j-1}[l_{j-1}]$. We say that point q gets *mapped* to cell l_j in table D_j. The tables $D_1, \ldots, D_r$ are populated in the preprocessing phase based on the data points $p_1, \ldots, p_n$. Note that the functions $l_1, \ldots, l_r$ are independent of the data points themselves.

Given a table population, a locator function l_j can be viewed as a function $F : \{0,1\}^d \to [m]$ that maps a query point q to a cell $F(q)$. Given such a function F the *volume* of a cell l is defined to be the fraction of points q from $\{0,1\}^d$ such that $F(q) = l$. We call a cell *heavy* if its volume is larger than $\frac{1}{\sqrt{m}}$ and we call it *light* otherwise.

Definition 4.1. *We say a function F is δ-focusing for a point p_i if there is a light cell l such that a $m^{-\delta}$ of the probability mass in μ_{p_i} gets mapped to cell l by the function F. We say F is a (δ, ν)-lens for a data set $\{p_1, \ldots, p_n\}$ if it is δ-focusing for at least a ν fraction of the p_i's.*

We remark that if all cells were light, F being δ-focusing is equivalent to the distribution $F(Y)$ having min-entropy at most $\delta \log m$, where Y is a random variable sampled from μ_{p_i}. Given the algorithm and D, the heaviness or lightness of a cell l in table D_j is naturally defined with respect to the corresponding locator function l_j. Similarly D is (j, δ)-focusing for a point p_i if the function l_j is δ-focusing, and D is a (j, δ, ν)-lens for a data set $\{p_1, \ldots, p_n\}$ if the function l_j is a (δ, ν)-lens. In this work, we assume that r is $O(\frac{\log n}{\log \log n})$ and that $mw \leq n^{1+\frac{\epsilon}{200r}}$. Also, we assume that $m \geq n$.

4.1 Good algorithm implies a lens

Lemma 4.2. *If there exists an algorithm A, such that with probability $\frac{1}{2}$ over the data set there is a (t, m, w)-data structure such that for half the points, A succeeds with probability at least a half, then, there is an algorithm A' for which with probability $\frac{1}{3}$ over the data set there is a $(t+1, m, w)$-data structure that is a $(t+1, \frac{1}{\log n}, \frac{1}{3})$-lens.*

Proof. Let h be a function from $\{0,1\}^d$ to $[m]$ to be fixed later. The algorithm A' is as follows: it simulates A for the first t reads, after which A outputs some point x (hopefully the correct answer to the query). The algorithm now queries location $h(x)$ in D_{t+1}. We claim that there exists a function h so that the database is $(t+1, \frac{1}{\log n}, \frac{1}{3})$-focusing with probability $\frac{1}{3}$. Indeed, let h be chosen uniformly at random from all functions from $\{0,1\}^d$ to $[m]$. By assumption, with probability $\frac{1}{2}$ it holds that for half of the points the algorithm A succeeds with probability at least $\frac{1}{2}$. We call this set of points S. For each $p \in S$ it holds that half of the mass of $\mu_{p,\epsilon}$ is mapped to $D_{t+1}[h(p)]$. Thus if $D_{t+1}[h(p)]$ is a light cell then the database is $(t+1, \frac{1}{\log n})$-focusing for p.

There are at most $\sqrt{m}$ heavy cells in D_{t+1}. Since the dataset is chosen randomly and the function h is independent of A and the dataset, the expected number of points in S that h maps to one of the heavy cells is at most $|S|/\sqrt{m}$. Thus the probability that more than $|S|/6$ of these points are mapped to heavy cells, is at most $\frac{1}{6}$. We conclude that with probability $\frac{1}{2} - \frac{1}{6}$ over the choice of h and the dataset, there are at least $\frac{n}{3}$ points for which the table is $(t+1, \frac{1}{\log n})$-focusing, as required. The determinism of A' is achieved by noting that there must exist at least one specific h for which the above condition holds, which is hardwired into A'. $\square$

4.2 Isoperimetric Inequality

Given a function $F : \{0,1\}^d \to [m]$, let $A_i \subset \{0,1\}^d$ be the set of points mapped to i. In this section we prove the isoperimetric bound which lies at the heart of our result: it shows that a small perturbation in the query point q may result in many different memory locations being read. The bound is a strengthening of a similar bound proven by Motwani *et al.* [22]. More precisely:

Lemma 4.3. *Let $A \subseteq \{0,1\}^d$ with $|A| \leq a \cdot 2^d$. Let $\epsilon \in (0, \frac{12}{13})$. Then*

$$\Pr_{y \in \{0,1\}^d}[\mu_{y,\epsilon}(A) \ \geq a^{\frac{\epsilon}{6}}] < a^{1+\frac{\epsilon}{6}}.$$

Proof. Let $\rho = 1 - 2\epsilon$ and let T_ρ denote the noise operator: for a function $f : \{0,1\}^d \to \mathbb{R}$, we define $T_\rho f(y) = E_{z \sim \mu_{y,\epsilon}} f(z)$, that is, the expectation is taken over points sampled from $\mu_{y,\epsilon}$. Let $||f||_p = (E_{x \in \{0,1\}^d}[|f(x)|^p])^{\frac{1}{p}}$.

We shall use the following property of the noise operator, see for instance [13]:

Theorem 4.4. [Hypercontractivity] $\|T_\rho f\|_2 \leq \|f\|_{1+\rho^2}$

Let $y \in \{0,1\}^d$. Then by the reversibility of the noise operator, $\mu_{y,\epsilon}(A) = (T_\rho \mathbf{1}_A)(y)$. By Chebyshev's inequality, the required probability

$$\begin{aligned}\Pr_{y \in \{0,1\}^d}[\mu_{y,\epsilon}(A) \ \geq a^{\frac{\epsilon}{6}}] &\leq \frac{E_y[(T_\rho \mathbf{1}_A)^2]}{a^{\frac{\epsilon}{3}}} \\ &= \frac{\|T_\rho \mathbf{1}_A\|_2^2}{a^{\frac{\epsilon}{3}}} \\ &\leq \frac{\|\mathbf{1}_A\|_{1+\rho^2}^2}{a^{\frac{\epsilon}{3}}} \\ &\leq a^{\frac{2}{1+(1-2\epsilon)^2} - \frac{\epsilon}{3}}\end{aligned}$$

which is easily seen to be bounded as required. □

Lemma 4.5. *Let $A_1, \ldots, A_m$ be partition of $\{0,1\}^d$ and let $L = \{i : |A_i| \leq 2^d/\sqrt{m}\}$ be the set of light cells. Then*

$$\Pr_{y \in \{0,1\}^d}[\max_{i \in L} \mu_{y,\epsilon}(A_i) \ \geq m^{-\frac{\epsilon}{12}}] < m^{-\frac{\epsilon}{12}}.$$

Proof. Let $a_i = \frac{|A_i|}{2^d}$. The above probability is at most

$$\sum_{i \in L} a_i^{1+\frac{\epsilon}{6}} \leq \max_{i \in L} a_i^{\frac{\epsilon}{6}} \sum_{i \in L} a_i \leq m^{-\frac{\epsilon}{12}}.$$

□

Remark 4.6. *The proof above also works when the uniform measure on $\{0,1\}^d$ is replaced by the Gaussian measure on $\mathbb{R}^d$, and the noise operator T_ρ is the "Ornstein-Uhlenbeck operator" $T_\rho f(x) = E_g[f(\rho \cdot x + \sqrt{1-\rho^2} \cdot g)]$, where g is again sampled from the Gaussian measure on $\mathbb{R}^d$. This leads to a lower bound of $n^{1+\Omega(\frac{1}{c^2})}$ for c-approximate near neighbor search in $(\mathbb{R}^d, \ell_2)$.*

4.3 Lower bound for Lenses

We first give a rough outline of the proof. For the purposes of this proof outline, it will be convenient to think of t as a fixed constant, and $\frac{m}{n} \approx w \approx n^{O(\epsilon)}$.

Lemma 4.5 essentially shows that a light cell can only have a small intersection with any of the balls μ_{p_i} around the dataset points p_i. This implies that (Lemma 4.7) a fixed function F cannot be a $(\frac{\epsilon}{12}, \frac{1}{4r})$-lens, with high probability over the choice of the p_i's.

We now repeatedly use the single query lower bound to prove the main result. For a *fixed* population of D, the single query lower bound suffices to show that with probability $(1 - exp(-n \log m))$, it does not focus more than a constant fraction of the points. While the number of possible populations of D is $exp(mw)$—much too large to support a union bound argument, we can still derive an upper bound (of roughly $exp(mw - n \log m)$) on the *expected* number of D's that focus many p_i's, where the expectation is taken over random choice of the dataset $p_1, \ldots, p_n$.

We then show that focusing populations of D occur in large groups. In other words, if for a given choice of p_i's there is one population D that focuses many p_i's, then we can derive a large number (roughly $exp(mw - \frac{mw}{n^\epsilon})$) of ways to populate D that still focus many p_i's: we show in Lemma 4.13 that if D focuses p_i, then preserving a very small fraction of table cells while arbitrarily changing the rest results in a new population that (with high probability) continues to focus p_i. This, combined with the bound on the expected number of focusing populations, gives us the desired upper bound on the algorithm's success probability (Lemma 4.11).

4.3.1 A fixed F is unlikely to be a lens for a random dataset

Lemma 4.7. *The probability that a function $F : \{0,1\}^d \to [m]$ is a $(\frac{\epsilon}{12}, \frac{1}{4r})$-lens for a randomly chosen dataset $p_1, ..., p_n$ is at most $2^{(n(1-\frac{\epsilon}{48r}\log m))}$.*

Proof. Lemma 4.5 says that the probability that for a random $y \in \{0,1\}^d$, $\max_{i \in L} \mu_y(A_i)$ exceeds $m^{-\frac{\epsilon}{12}}$ is at most $m^{-\frac{\epsilon}{12}}$, where L is the set of light cells. Thus the probability that F is $\frac{\epsilon}{12}$-focusing for a random y is at most $m^{-\frac{\epsilon}{12}}$. The probability that F is $\frac{\epsilon}{12}$-focusing for at least a $\frac{n}{4r}$ of the n data points is then at most $\binom{n}{\frac{n}{4r}}(m^{-\frac{\epsilon}{12}})^{\frac{n}{4r}}$. The claim follows. □

Corollary 4.8. [Few Lenses] *For $t \leq r$, let $D_1, \ldots, D_{t-1}$ be a fixed population of the first $(t-1)$ tables of a (r, m, w)-data structure and let a dataset $p_1, \ldots, p_n$ be chosen randomly. The probability that D is a $(t, \frac{\epsilon}{12}, \frac{1}{4r})$-lens is at most $2^{(n(1-\frac{\epsilon}{48r}\log m))}$.*

Remark 4.9. *This is the only property of the approximate nearest neighbor function that we will use. Thus if we can show a similar upper bound on the probability of a fixed D being a lens for another problem and input distribution, we can derive similar lower bounds for the cell-probe complexity of the corresponding problem.*

4.3.2 Extending to t-table data structures

We introduce one more bit of notation. Let $\delta_1 = \frac{\epsilon}{20}$ and $\delta_{j+1} = \delta_j - \frac{\epsilon}{40r}$.

Definition 4.10. *We say D is t-*concentrating *for a datapoint p if D is not (j, δ_j)-focusing for p, for $j < t$, but D is (t, δ_t)-focusing for p.*

Observe that if D is (t, δ_t)-focusing for p, then it is j-concentrating for p for some $1 \leq j \leq t$. The following is the main technical result of this section:

Lemma 4.11. *Let a dataset $p_1, \ldots, p_n$ be chosen randomly. Let $m \leq n^{1+\frac{\epsilon}{200r}}$ and $w \leq n^{\frac{\epsilon}{200r}}$. For any integer $t < r$, the probability that there exists a (r, m, w)-datastructure D that is t-concentrating for more than $\frac{n}{3r}$ data points is at most $exp(-\Omega(n))$,*

Proof. For $t = 1$ the claim follows from Corollary 4.8: Note that $\delta_1 \leq \frac{\epsilon}{12}$ and observe that whether or not D is $(1, \delta)$-focusing for p depends only on the lookup algorithm and on p, but not on the contents of D.

Assume $t > 1$ and let B be the event that there is such a (r, m, w)-datastructure that is t-concentrating for more than $\frac{n}{3r}$ data points in the randomly chosen dataset. Let Y be a random variable denoting the number of $(t, \frac{\epsilon}{12}, \frac{n}{4r})$-lenses for a random dataset.

From Corollary 4.8, we conclude that

$$E[Y] \leq 2^{mwr} 2^{n(1-\frac{\epsilon}{48r}\log m)}.$$

We show that $E[Y|B]$ is much larger, thus deducing that $Pr[B] \leq E[Y]/E[Y|B]$ is small.

Lemma 4.12. [One implies many]

$$E[Y|B] \geq 2^{mwr-(2mwrm^{-\frac{\epsilon}{100r}}+\sqrt{m}w)}$$

In other words we show that one such D implies a large number of $(t, \frac{\epsilon}{12}, \frac{n}{4r})$-lenses. We conclude that

$$\begin{aligned} Pr[B] &\leq E[Y]/E[Y|B] \\ &\leq 2^{mwr}2^{n(1-\frac{\epsilon}{48r}\log m)}2^{-mwr+(2mwrm^{-\frac{\epsilon}{100r}}+\sqrt{m}w)} \\ &= 2^{2mwrm^{-\frac{\epsilon}{100r}}+\sqrt{m}w+n-n\frac{\epsilon}{48r}\log m} \end{aligned}$$

Since $mw \leq n^{1+\frac{\epsilon}{200r}}$, the first term is negligible compared to the last whenever $r = O(\log n)$. Moreover, the second term is $o(n)$. Finally, for $r \leq \frac{\epsilon}{96}\log m$, the last term is at least $2n$ in magnitude. This concludes the proof of Lemma 4.11. □

We next prove Lemma 4.12.
Intuition: To get some intuition into the proof, consider a simple case where $t = 2$ (two tables). Event B means that there is a data structure with two tables which is $(2, \delta_2)$-focusing for $\Omega(n)$ points in the dataset; for simplicity assume that all cells are light and the data structure is $(2, 0)$-focusing for these points p_i. This means that the function l_2 maps the entire ball $\mu_{p_i,\epsilon}$ to the same cell. Let us now select a random α fraction of cells in the first table and perturb the contents of the remaining cells to random values. We will argue that even after this alteration the function l_2 still does a reasonable job of focusing the ball $\mu_{p_i,\epsilon}$.

If all cells are light, by Lemma 4.5, no cell in D_1 receives more than $m^{-\epsilon/12}$ fraction of the ball under the function l_1. This means the ball breaks into at least $m^{\epsilon/12}$ small pieces scattering across different cells. Since α fraction of the cells remain unchanged after the perturbation, we can expect about α fraction of the ball to remain unchanged under the new perturbed function l_2, and this happens with high probability if $\alpha m^{\epsilon/12} >> 1$. Setting $\alpha = m^{-\epsilon/24}$ for instance would essentially imply that the new function is $\alpha = m^{-\epsilon/24}$-focusing. This means the number of data structures that are $m^{-\epsilon/24}$-focusing is at least $2^{mw-mwm^{-\epsilon/24}}$.

For a more general proof, suppose that B occurs, and let D be t-concentrating for at least $\frac{n}{3r}$ of the p_i's. Note that the t-concentrating property depends only on the first $(t-1)$ columns of D. We extend the idea of *perturbation* of these columns of D to obtain a large number of ways to populate the tables that are $(t, \frac{\epsilon}{12}, \frac{n}{4r})$-lenses. The existence of heavy cells further complicates things, but their small number helps handle them.

Perturbation: The perturbation is done by selecting a small number of cells of D that are left unchanged, and then arbitrarily changing the cells that are not selected. We consider the following two-step randomized selection procedure. In the first step, we select each cell with probability $m^{-\epsilon/100r}$. In the second step, each heavy cell is (deterministically) selected. Since each heavy cell contains a volume of at least $\sqrt{m}$, there are no more than $\sqrt{m}$ heavy cells in each column. Note that each cell, heavy or light, is selected with probability at least $m^{-\epsilon/100r}$.

The following Lemma shows that the perturbations of D are likely to be good lenses.

Lemma 4.13. [Perturbations create many lenses] *Let $t > 1$ be any integer less than r. Suppose that for a data point p, a table D is t-concentrating so that $m^{-\delta_t}$ of the probability mass in μ_p gets mapped to cell l_p in table D_t. Then with probability at least $(1 - exp(-m^{\frac{\epsilon}{200r}}))$ (taken over coin tosses of the selection procedure), the resulting perturbed data structure D' has the property that $m^{-\delta_t-\frac{\epsilon}{50}}$ of the probability mass in μ_p gets mapped to cell l_p in table D'_t.*

Proof. By assumption, a fraction $m^{-\delta_t}$ of the measure in μ_p gets mapped to cell l_p in table D_t.

For any $q \in \{0,1\}^d$, the location $l_t(q)$ depends only on the population of $D_1(l_1), \ldots, D_{t-1}(l_{t-1})$. Since any perturbation D' of D agrees with D on all selected entries, $l_t(q, D')$ is the same as $l_t(q, D)$ whenever all these $(t-1)$ locations are selected, which happens with probability at least $m^{-\epsilon/100}$. Thus if an $m^{-\delta_t}$ fraction of the measure in μ_p gets mapped to cell l_p in D, at least a $m^{-\delta_t - \frac{\epsilon}{100}}$ fraction of the measure on μ_p gets mapped to l_p in D', in expectation.

It remains to show that this fraction is unlikely to be much smaller than its expectation. Intuitively, this fraction depends upon many independent choices, and thus should be well concentrated.

Consider a layered directed graph G with one source node p in layer 0, and m nodes in each of layers $1, \ldots, t$, each corresponding to a cell in D; we denote by $v(j, l)$ the node in G corresponding to location $D[j][l]$. Every node in layer j has an edge to every node in layer $(j+1)$. For each point $q \in \{0,1\}^d$, we have a flow of $\mu_p(q)$ units from the source p to the last layer, passing through nodes $v(j, l_j(q, D))$. Thus a flow of at least $m^{-\delta_t}$ flows from source p to a node $v(t, l_p)$. Let G' be the subgraph of G that contains only the nodes corresponding to selected cells in layers $1, \ldots, t-1$; for convenience we deterministically select only cell l_p in layer t. Our goal is to show that with high probability, a large fraction of the flow from p to $v(t, l_p)$ survives in G'.

We construct a sequence of graphs $G = G_0, G_1, \ldots, G_t = G'$, where G_j has only the selected nodes from the last j layers (i.e. layers $t+1-j, \ldots, t$), and all the nodes from levels $1, \ldots, t-j$. We will show by induction that with probability at least $(1 - jexp(-m^{\epsilon/200r}))$, the flow from p to $v(t, l_p)$ that survives in G_j is at least $m^{-\delta_t - (j-1)\frac{\epsilon}{50r}}$. The base case when j is 1 is immediate.

By the induction hypothesis, with probability at least $(1 - jexp(-m^{\epsilon/200r}))$, the flow from p to $v(t, l_p)$ that survives in G_j is at least $m^{-\delta_t - (j-1)\frac{\epsilon}{50r}}$. Suppose that the surviving flow f_j is indeed this large; we show that with probability at least $(1 - exp(-m^{\epsilon/200r}))$, at least a $m^{-\frac{\epsilon}{50r}}$ fraction of this flow will survive in G_{j+1} as well, which would imply the induction step. Let X_l be an indicator variable for the cell $D[t-j][l]$ being selected in the first step of the selection procedure. Let $f_{j+1}(X)$ denote the flow that survives in G_{j+1}. Clearly, $E[f_{j+1}(X)] \geq f_j m^{-\frac{\epsilon}{100r}}$. Let b_l denote the amount of flow in G_j that passes through node $v(t-j, l)$. Thus $f_j = \sum_l b_l$. Let c_l the maximum amount by which $f_{j+1}(X)$ can change by flipping the variable X_l. Clearly, c_l is zero whenever cell $D[t-j][l]$ is heavy, and at most b_l otherwise. Moreover, since D is not $(t-j, \delta_{t-j})$-focusing for p, c_l is at most $m^{-\delta_{t-j}}$. Thus

$$\begin{aligned} \sum_l c_l^2 &\leq \max_l c_l \sum_l c_l \\ &\leq \max_l c_l f_j \\ &\leq m^{-\delta_{(t-j)}} f_j \end{aligned}$$

Thus by Azuma's inequality,

$$\begin{aligned} Pr[f_{j+1}(X) \leq \frac{1}{2} E f_{j+1}(X)] &\leq exp(-(E f_{j+1}(X))^2 / 8 \sum_l c_l^2) \\ &\leq exp(-f_j^2 m^{-\frac{\epsilon}{50r}} / 8 m^{-\delta_{t-j}} f_j) \\ &\leq exp(-f_j m^{-\frac{\epsilon}{50r}} / 8 m^{-\delta_{t-j}}) \\ &\leq exp(-m^{-\delta_t - (j-1)\frac{\epsilon}{50r} - \frac{\epsilon}{50r} + \delta_{t-j}} / 8) \\ &\leq exp(-m^{\frac{\epsilon}{200r}}) \end{aligned}$$

since $\delta_{t-j} - \delta_t = j\frac{\epsilon}{40r}$. This completes the proof of Lemma 4.13. $\square$

We now complete the proof of Lemma 4.12.

of Lemma 4.12. A standard Chernoff bounds argument implies that the number of cells selected in the first step of the selection process is well concentrated. Thus there exists a choice of coin tosses for the selection procedure such that the following two properties hold

1. The number of selected cells is at most $2mrm^{-\frac{\epsilon}{100r}} + \sqrt{m}$.

2. For at least $\frac{n}{3r}(1 - o(1)) \geq \frac{2n}{7r}$ of the p_i's, there is a cell l_i that gets $m^{-\delta_t - \frac{\epsilon}{50}}$ of the probability mass in μ_{p_i}.

We next account for some of these l_i's becoming heavy in a perturbed tables. Note that since the cells l_i are all light in D, at most $m^{-\frac{1}{2}}/m^{-\delta_t}$ of the l_i's can take any fixed value l. Thus one cell becoming heavy can hurt at most $m^{-\frac{1}{2}}/m^{-\delta_t}$ of the p_i's. Since there are at most $\sqrt{m}$ heavy cells in any set of tables, no more than $m^{\delta_t} < \frac{n}{8r}$ of the p_i's can have their corresponding l_i's become heavy. Thus each of these perturbed populations is a $(t, \delta_t + \frac{\epsilon}{50}, \frac{n}{4r})$-lens.

This gives us $2^{mwr - (2mwrm^{-\frac{\epsilon}{100r}} + \sqrt{m}w)}$ different $(t, \frac{\epsilon}{12}, \frac{n}{4r})$-lenses. Thus

$$E[Y|B] \geq 2^{mwr - (2mwrm^{-\frac{\epsilon}{100r}} + \sqrt{m}w)}.$$

$\square$

Theorem 4.14. *Let dataset $p_1, \ldots, p_n$ be chosen randomly and let A be some algorithm. Let $m \leq n^{1+\frac{\epsilon}{200r}}$ and $w \leq n^{\frac{\epsilon}{200r}}$. The probability that there exists an (r, m, w)-data structure D which is a $(r, \delta_r, \frac{1}{3})$-lens is at most $r \cdot exp(-\Omega(n))$.*

Proof. Note that any table D that is (r, δ_r)-focusing for a datapoint p must be j-concentrating for p, for some $j \leq t$. Thus by lemma 4.11, with probability $(1 - r \cdot exp(-\Omega(n))$, the number of points p for which D can be (r, δ_r)-focusing is at most $r \cdot \frac{n}{3r}$. The claim follows. $\square$

4.4 Putting it Together

Theorem 4.15. *Any t-probe randomized data structure for $\frac{1}{8\epsilon}$-approximate near neighbor search that succeeds with probability $\frac{2}{3}$ and uses word size $w \leq m^{\frac{\epsilon}{200t}}$ must use space $m > n^{1+\frac{\epsilon}{200t}}$.*

Proof. Suppose a randomized algorithm A succeeds with average probability $\frac{2}{3}$, when the average is taken over choosing a random input a distribution $\mathcal{D}$. By averaging, one can fix the coin tosses of the algorithm so that we get a deterministic algorithm with the same guarantee. We next define a distribution over input instances such that no deterministic algorithm A can succeed with probability larger than $\frac{2}{3}$.

Fix $d \geq 10 \log n$. The n points are drawn independently and uniformly at random from the hypercube $\{0,1\}^d$. The query point is constructed as follows: We first pick an $i \in [n]$ uniformly at random. q is obtained from p_i be flipping each bit of p_i independently with probability $\epsilon = \frac{1}{8c}$. The following geometric facts are standard:

Proposition 4.16. *Let $p_1, \ldots, p_n$ be sampled uniformly and independently from $\{0,1\}^d$, then*

- *With probability $(1 - \frac{1}{poly(n)})$, $\min_{i \neq j} \|p_i - p_j\|_1 \geq \frac{d}{4}$.*

- *With probability* $(1 - \frac{1}{poly(n)})$, $\min_i \|q - p_i\|_1 \leq 2\epsilon d$.

Assuming that these conditions are satisfied, any $\frac{1}{8\epsilon}$-approximate near neighbor algorithm must output p_i when given q. Thus A must succeed with probability $\frac{1}{2}$ on the distribution. By Lemma 4.2, with probability $\frac{1}{2}$, there must be a $(t+1, m, w)$ table that is a $(t+1,$
$frac1\log n, \frac{1}{3})$-lens. Theorem 4.14 then implies the result. □

Remark 4.17. ***Far Neighbor Problem:*** *The instances for far-neighbor are created by asking the query* $\overline{q}$ *instead of* q *(i.e. flipping each bit of* q*). We omit the details from this extended abstract.*

4.5 Non-adaptive Lower bounds

An (r, m, w) non-adaptive algorithm is an algorithm in which given n input points $p_1 \dots p_n$ in $\{0,1\}^d$, we prepare a table D which consists of m words, each w bits long. Given a query point q the algorithm can probe the table at most r times. For every $t \leq r$, the location of the t'th probe $l_t = l_t(q)$ depends only upon the query point q and not upon the content that was read in the previous queries. In other words, the functions $l_1, l_2, \dots, l_r$ are functions of q only. In this section we show a time-space cell-probe lower bound which is slightly stronger than the one shown for general adaptive algorithms. While non adaptivity is a harsh restriction, we stress that the best known upper bounds are all non-adaptive; thus improving the non-adaptive lower bound is of interest.

As before, fix $\epsilon = \frac{1}{8c}$. The way the dataset and the query point are sampled is slightly different from the adaptive case. We prepare the input as follows: First sample uniformly at random n points $S := s_1, \dots, s_n$. For each i we set p_i to be an independent sample from $\mu_{s_i,\epsilon/2}$. Note that in effect, the p's are uniformly sampled from $\{0,1\}^d$. We choose i at random and sample the query point q_i from $\mu_{s_i,\epsilon/2}$ as well. Note, that while it is not true that q_i is sampled from $\mu_{p_i,\epsilon}$, it is still the case that with high probability $||q_i - p_i||_1 \leq 2\epsilon d$ while $||p_i - p_j||_1 \geq \frac{d}{4}$ for $i \neq j$. The table is populated based on the p's. Our assumption regarding the correctness of the algorithm is that when the input is thus sampled, for each i, with probability $1/2$ over the choice of s_i and p_i it holds that with probability $2/3$ over the choice of q_i, the algorithm can reconstruct p_i. Note that, as before, we can fix the coin tosses of the algorithm and assume the algorithm is deterministic. We now assume that the query algorithm is given S too. The access to S is considered free and is not accounted as a memory query[4]. Clearly, giving the algorithm access to S may only increase its probability of computing p_i successfully.

Theorem 4.18. *A* (r, m, w)*-non adaptive algorithm with the above properties has* $mw \geq \epsilon d n^{1+\Omega(\epsilon/r)}$

In particular, in the important case where w is $\Theta(d)$ and m is $O(n)$, we have that $r \in \Omega(\log n)$. We note that such a bound seems to be beyond the power the communication complexity techniques.

Our methods are similar to the ones in the adaptive case. We use the isoperimetric inequality to show that the information learnt from a single query is bounded. The non-adaptivity allows us to avoid the random perturbation argument of the adaptive case and deal directly with the information learnt from each query.

Proof. Let k be some parameter to be fixed later and let L be a set of k locations chosen uniformly at random in $1, \dots, m$. For notational convenience we write $D[L]$ to indicate the set $\{D[i] \mid i \in L\}$. Note that once S is fixed, it holds that $q|S$ is independent from $p_i|S$. Also, since D was populated given the p's (i.e. S is not

[4]We emphasize that our lower bounds are for the search problem. The instance is a YES instance of the natural decision problem.

given to the preprocessing procedure), it holds that $D[L] \mid S$ is independent from $q \mid S$. It follows that once S, L and q are fixed it holds that

$$I(D[L]; p_i \mid S, L, q) = I(D[L]; p_i \mid S, L) \tag{1}$$

where $I(\cdot\,;\,\cdot)$ indicates mutual information. Now since the p_i's are mutually independent (also given S and L) we have that for every fixed S, L

$$\sum_{i=1}^{n} I(D[L]; p_i \mid S, L) \leq H(D[L] \mid S, L) \leq wk$$

where $H(\cdot)$ indicates the entropy function. Taking expectations on both sides we have:

$$\sum_{i=1}^{n} \mathbb{E}_{L,S}\big[I(D[L]; p_i \mid S, L)\big] \leq wk \tag{2}$$

We set $k := \frac{m}{n^{\Omega(\epsilon/r)}}$. Our goal is therefore to show that $\mathbb{E}_{L,S}\big[I(D[L]; p_i \mid S, L)\big] \in \Omega(\epsilon d)$, as this would immediately imply the theorem. Note that $H(p_i) - H(p_i \mid S)$ is $\Omega(\epsilon d)$, thus it is enough to show that for every i, when probing $D[L]$, the algorithm can reconstruct p_i given S with constant probability.

Denote by $l_i(q)$ the location of the $i'th$ query when the query point is q. We write $l_{[r]}(q)$ to denote $l_1(q) \cup ... \cup l_r(q)$. We say a point q_i is *good for* p_i if it is of distance at most ϵd from s_i and p_i can be reconstructed from q_i, s_i and $D[L_{[r]}(q_i)]$. Let Q_i denote the set of points which are good for p_i. The correctness of the algorithm implies that with probability at least a half, $\mu_{s_i,\epsilon/2}(Q_i) \geq \frac{2}{3}$.

Define A_j^t to be the set of $q \in \{0,1\}^d$ such that $j = l_t(q)$. In the non-adaptive domain we can assume that all cells are light; i.e. w.l.o.g for every $1 \leq j \leq m$ and $1 \leq t \leq r$ it holds that $|A_j^t| \leq \frac{2^d}{m}$. The reason is that a cell j for which A_j^t is large (for some t) could be split into $|A_j^t|/\frac{2^d}{m}$ light cells with the total number of new cells bounded by m.

Definition 4.19. *A point s_i is* shattered *if*

$$\max_{j,t} \mu_{s_i,\epsilon/2}(A_j^t) \leq n^{-\epsilon/12}$$

For every non-adaptive algorithm, Lemma 4.5 implies that the probability over the choice of s_i that s_i is not shattered, is at most $n^{-\epsilon/12}$. We conclude that with probability $\geq \frac{1}{3}$ it holds that s_i is shattered *and* $\mu_{s_i,\epsilon/2}(Q_i) \geq \frac{2}{3}$.In such a case we show that with constant probability there exists a $q \in Q_i$ such that $l_{[r]}(q) \subset L$, i.e. there exists a good point for which all the query points are covered by L, thus, a procedure which samples points from $\mu_{s_i,\epsilon/2}$ until it finds a point covered by $D[L]$ can reconstruct p_i with a constant probability.

Since s_i is shattered it holds that $\mu_{s_i,\epsilon/2}(A_j^t)$ is at most $n^{-\epsilon/12}$ for any j, t.Since $\mu_{s_i,\epsilon/2}(Q_i) \geq \frac{2}{3}$ we conclude that there are at least $\frac{2n^{\epsilon/12}}{3r}$ different good q with disjoint hash locations. Each one of these q is covered by L with probability at least $\frac{r}{n^{\epsilon/12}}$, so with probability $\geq \frac{1}{2}$ at least one of them is covered and p_i is reconstructed. □

4.6 Connection to Locally Decodable Codes

Let $m = m(n, q, k)$ be the minimum size of a data structure that can reconstruct each of n symbols $p_1, \ldots, p_n$ by reading one q-tuple out of k possible disjoint q-tuples, where the minimum is taken over all possible configurations of k disjoint q-tuples. When k is $\Theta(n)$ and q is $O(1)$ this problem is tightly related to the problem of designing small locally decodable codes (LDC)[18]: if an adversary corrupts a small constant fraction of the data structure, it is still possible to reconstruct a symbol p_i with constant probability by reading a random r-tuple. Locally decodable codes have been extensively studied, and proving a superpolynomial lower bound on m is a well known open problem.

Our work reveals an interesting connection between the query complexity of non-adaptive algorithms and $m(n, q, k)$ with a different choice of parameters. We have shown that non-adaptive algorithms with query complexity q must be able to reconstruct the point from $n^{\Omega(\epsilon)}$ disjoint q-tuples. Thus, a bound of the form $m(n, q, n^{\Omega(\epsilon)}) \in n^{1+\Omega(\epsilon)}/q$ would imply that locality sensitive hashing techniques are (essentially) the best one can achieve with non-adaptive algorithms. The reduction to LDC only uses the fact that most points are shattered, it does not use the geometry of the Near Neighbor problem in any other way. The contra-positive implies the following:

Theorem 4.20. *A non adaptive algorithm for $\epsilon-$ANNS, with space s and query complexity t implies that $m(n, n^{\Omega(\epsilon)}, t) \leq s$.*

5 Partial Match Lower bounds

The Partial Match Problem: The partial match problem consists of building a data structure that supports partial match queries. The dataset consists of entries from $\{0,1\}^d$ and the query is taken from $\{0,1,\star\}^d$ where $\star$ is interpreted as a 'don't cares'. A query q is then compared against a dataset entry p bitwise where a match is interpreted to mean that a 0 matches a 0, a 1 matches a 1 and a $\star$(a don't care) in q matches either a 0 or a 1. Given such a query, the objective of the data structure and the algorithm is to find a matching entry from the database if any.

We prove a lower bound for this problem similar to the lower bound proven for the near neighbor search problem. Our approach is to prove an isoperimetric inequality similar to Lemma 4.5. To that end we first define the distribution from which the data is drawn.

Our Distribution: Let H_δ denote the set of points in $\{0,1\}^d$ with exactly δd 1's. Each entry in the dataset is chosen uniformly and independently in H_δ. Once the data structure is built we sample a query point 'around' a random dataset point p as follows: first, we convert the 1's in p to $\star$'s. Then we convert an additional $(\frac{d}{2} - \delta d)$ $0's$ to $\star$'s at random so that the resulting query point q has exactly $d/2$ 0's and $d/2$ 1's. Observe that for each point p the query point q is randomly sampled from $\binom{(1-\delta)d}{(\frac{1}{2}-\delta)d}$ possible choices. Observe that for $d \geq \Omega(\log n)$, with high probability the query point generated in this way can only be matched to the point p used to generate q, and not to any other point in the dataset.

We will use the notation $\nu_{p,\delta}$ to denote the distribution of q obtained from a database entry p. Our goal is Lemma 5.3 which is an isoperimetric inequality for the operator $\nu_{p,\delta}$.

Let $F : \{0,\star\}^d \to [m]$ where $d = \Omega(\log n)$. Light and heavy cells are defined as before with respect to the query space consisting of $\{0,\star\}^d$ with exactly equal number of 0's and $\star$'s. Let L denote the set of light cells. We first show that two random query points around p are unlikely to map to the same light cell.

Lemma 5.1. *Let p be a random database entry. Let q_1 and q_2 be two random points chosen from the distribution ν_p.* $\Pr[(F(q_1) = F(q_2) = i) \wedge (i \in L)] \leq m^{-\Omega(1)}$

Proof. Since F operates on $\{0, \star\}^d$, we can substitute '$\star$' by '1' and think of it as a function $F : H_{\frac{1}{2}} \to [m]$. We will now use the fact that there is no hash function F for the near neighbor search problem that hashes two nearby points to the same bucket. Let q_1 and q_2 be two independent samples from $\nu_{p,\delta}$. It is easy to check that $\mathbb{E}[||q_1 - q_2||_1] = d(1/2 - \delta)/(1 - \delta) =: h_0$. Furthermore, by standard concentration bounds we have $\Pr[||q_1 - q_2||_1 \leq h_0/2] \leq m^{-\Omega(1)}$.

For any $h > h_0/2$, let $\epsilon = h/d$ and $\epsilon_0 = h_0/2d$. From Lemma 4.5 we know that for a random point $v_1 \in \{0,1\}^d$ and a random point v_2 chosen from $\mu_{v_1,\epsilon}$, $\Pr[F(v_1) = F(v_2) \in L] \leq m^{-\epsilon/6} \leq m^{-\epsilon_0/6}$. In our case the points q_1, q_2 are sampled from a somewhat different distribution, yet Lemma 4.5 is useful for our distribution as well. We use the following key observation:

Lemma 5.2. *Let E denote the event that v_1 and v_2 have equal number of $0's$ and $1's$. For every h and every ϵ, δ, if v_2 is sampled from $\mu_{v_1,\epsilon}$ and q_1, q_2 are sampled from $\nu_{p,\delta}$ it holds that*

$\{v_1, v_2 \mid E, ||v_1 - v_2||_1 = h\}$ *is distributed as*

$$\{q_1, q_2 \mid ||q_1 - q_2||_1 = h\}$$

Proof. The proof is a straight forward coupling. Both v_1 and q_1 are chosen uniformly in $H_{\frac{1}{2}}$, so they could be coupled to each other. Now, v_2 is sampled uniformly from all points in $H_{\frac{1}{2}}$ which are of Hamming distance h from v_1. Note that we do not condition on p, thus q_2 is also sampled uniformly among all points in $H_{\frac{1}{2}}$ which are of Hamming distance h from q_1. The Lemma follows. □

Note that the probability of E is high, at least $\Omega(1/d)$. Furthermore, if $\epsilon = h/d$ it holds that $\Pr[||v_1 - v_2||_1 = h \mid E] \in \Omega(1/\sqrt{d})$. Now we have

$$\begin{aligned}
&\Pr[F(q_1) = F(q_2) \in L] \\
&\quad \leq \Pr[F(q_1) = F(q_2) \in L \mid ||q_1 - q_2||_1 > h_0/2] \\
&\qquad + Pr[||q_1 - q_2||_1 \leq h_0/2] \\
&\quad \leq \Pr[F(q_1) = F(q_2) \in L \mid ||q_1 - q_2||_1 > h_0/2] + m^{-\Omega(1)} \\
&\quad \leq \max_{h > h_0/2} \Pr[F(q_1) = F(q_2) \mid ||q_1 - q_2||_1 = h] + m^{-\Omega(1)}.
\end{aligned}$$

By Lemma 5.2 the above expression is at most

$$\leq \max_{h > h_0/2} \Pr[F(v_1) = F(v_2 \in L) \mid E, ||v_1 - v_2||_1 = h] + m^{-\Omega(1)}$$

where v_2 is sampled from μ_{ϵ,v_1} and $\epsilon = h/d$

$$\begin{aligned}
&\leq \max_{h > h_0/2} \frac{\Pr[F(v_1) = F(v_2) \in L]}{\Pr[E]\Pr[||v_1 - v_2||_1 = h \mid E]} + m^{-\Omega(1)} \\
&\leq \max_{h > h_0/2} \Pr[F(v_1) = F(v_2) \in L] O(d^{3/2}) + m^{-\Omega(1)} \\
&\leq m^{-\Omega(1)} O(d^{3/2}) + m^{-\Omega(1)} \\
&= m^{-\Omega(1)}
\end{aligned}$$

□

Let A_i denote the set of query points that are mapped to cell i by F.

Lemma 5.3. $\Pr_p[\max_{i \in L} \nu_p(A_i) \geq m^{-c/2}] \leq m^{-c/2}$

Proof.

$$\begin{aligned}
\Pr[(F(q_1) = F(q_2) = i) \wedge (i \in L)] &= \sum_{i \in L} \Pr[F(q_1) = F(q_2) = i] \\
&= \sum_{i \in L} \nu_p(A_i)^2 \\
&\geq \max_{i \in L} \nu_p(A_i)^2
\end{aligned}$$

But $\Pr_p[(F(q_1) = F(q_2) = i) \wedge (i \in L)] = E_p[\Pr[(F(q_1) = F(q_2) = i) \wedge (i \in L)]]$ So by Lemma 5.1 $E_p[\max_i \nu_p(A_i)^2] \leq m^{-c}$. By Markov's inequality, $\Pr_p[\max_i \nu_p(A_i) \geq m^{-c/2}] \leq m^{-c/2}$ □

Lemma 5.3 has the same form as Lemma 4.5, which is the only property of the near-neighbor problem we used. Thus we can show that

Theorem 5.4. *There exists a universal constant c such that any t-probe randomized data structure for the partial match search problem that succeeds with probability $\frac{2}{3}$ and uses word size $w \leq m^{\frac{c}{200t}}$ must use space $m > n^{1+\frac{c}{200t}}$.*

Remark 5.5. *With some minor changes it is easy to show that the lower bound also holds for the* approximate *partial match problem, where the match between the query point and the dataset entry is allowed to have a small rate of errors.*

6 Connection to Communication Complexity

Miltersen *et al.* [21] show the following connections between cell probe complexity and two-player asymmetric communication complexity:

Theorem 6.1. *[21] Let f be computable by a t-probe data structure using $m = 2^a$ words of size b bits each. Then f has a t-round protocol where Alice, holding the query communicates $a = \log m$ bits per round, and Bob, holding the dataset communicates w bits per round.*

Theorem 6.2. *[21] Let f have a t-round protocol where Alice and Bob communicate a and b bits per round respectively. Then f has a t-probe data structure that stores 2^{ta} words of size tb each.*

Thus while, we have equivalence between the models for $t = 1$, the reduction is lossy for large t.

Our results can be re-interpreted as showing asymmetric communication complexity lower bounds for a multiple non-interacting servers setting. Indeed consider the following multiple server communication complexity model: There are t servers $S_1, \ldots, S_t$, each holding a copy of the dataset P. Alice is given the query q and wants to compute a function $f(q, P)$. Alice, and each of the servers additionally have access to a uniform random string R. An $[a, b, t, \delta]$ multiserver protocol in this setting is one where in the ith round, Alice sends a bits to server S_i and gets a b bit response. The servers are isolated and not allowed to communicate with each other, and at the end of t rounds, Alice must output $f(q, P)$ with probability $(1 - \delta)$, where the probability is taken over the randomness in R. Note that this model is similar to one used in Private Information Retrieval [11] but it allows for arbitrary shared randomness. Moreover, if the servers were allowed to communicate the model would reduce to the usual two-party asymmetric communication complexity setting.

The following theorems are to be compared with Theorems 6.1 and 6.2:

Theorem 6.3. *Suppose a function f can be computed with probability $(1-\delta)$ by a t-probe data structure with $m = 2^a$ cells of word size b. Then f has a $[a = \log m, b, t, \delta]$ protocol.*

Proof. Each of the servers constructs the data structure. The message to server i specifies the $(\log m)$-bit address of the ith query, and the response from server i is the word stored in the corresponding location in the data structure. □

Theorem 6.4. *Suppose a function f has an $[a, b, t, \delta]$ multiserver protocol where the servers use r bits of randomness. Then f can be computed with probability $(1-\delta)$ by a t-probe data structure with 2^a cells of word size $bt + r$.*

Proof. The jth cell in data structure contains the concatenation of the responses that the servers would give when they receive the binary representation of j from Alice, along with the randomness used by the servers. □

Thus this communication complexity model is a more faithful proxy for the cell probe model that asymmetric communication complexity, as t grows.

7 Conclusions

We presented a new non-communication-complexity-based approach to prove cell probe lower bounds. While we show lower bounds for approximate near neighbor and partial match, several open questions remain. It is natural to try to prove similar isoperimetric-type inequalities for other problems, which would imply cell-probe lower bounds using our techniques.

There is still a gap between our lower bounds and the known upper bounds for approximate near-neighbor. A first step may be to try and improve our non-adaptive lower bounds to match the upper bound. More generally, understanding the gap between adaptive and non-adaptive algorithms for near-neighbor and for partial match is an interesting open question.

Acknowledgments

We thank Sergey Yekhanin for pointing out the connection to locally decodable codes and Piotr Indyk and Alexandr Andoni for useful discussions.

References

[1] M. Ajtai. A lower bound for finding predecessors in yao's call probe model. *Combinatorica*, 8(3):235–247, 1988.

[2] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.

[3] A. Andoni, P. Indyk, and M. Pătraşcu. On the optimality of the dimensionality reduction method. In *Proc. of 47 FOCS*, pages 449–458, 2006.

[4] O. Barkol and Y. Rabani. Tighter lower bounds for nearest neighbor search and related problems in the cell probe model. *J. Comput. Syst. Sci.*, 64(4):873–896, 2002.

[5] P. Beame and E. Vee. Time-space tradeoffs, multiparty communication complexity, and nearest-neighbor problems. In *Proc. of 34th STOC*, pages 688–697, 2002.

[6] A. Borodin, R. Ostrovsky, and Y. Rabani. Lower bounds for high dimensional nearest neighbor search and related problems. In *Proc. of 31st STOC*, pages 312–321. ACM, 1999.

[7] A. Chakrabarti, B. Chazelle, B. Gum, and A. Lvov. A lower bound on the complexity of approximate nearest-neighbor searching on the hamming cube. In *Proc. of 31st STOC*, pages 305–311. ACM, 1999.

[8] A. Chakrabarti and O. Regev. An optimal randomised cell probe lower bound for approximate nearest neighbour searching. In *Proc. of 45th FOCS*, pages 473–482, 2004.

[9] M. Charikar, P. Indyk, and R. Panigrahy. New algorithms for subset query, partial match, orthogonal range searching, and related problems. In *ICALP*, pages 451–462, 2002.

[10] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proc. of 34th STOC*, pages 380–388. ACM, 2002.

[11] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

[12] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc of 20th Symposium on Computational Geometry (SoCG)*, pages 253–262, 2004.

[13] I. Dinur and E. Friedgut. Lectore notes on analystical methods in combinatorics and computer science.

[14] A. Gál and P. B. Miltersen. The cell probe complexity of succinct data structures. *Theor. Comput. Sci.*, 379(3):405–417, 2007.

[15] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. of 28th STOC*, pages 604–613, 1998.

[16] T. S. Jayram, S. Khot, R. Kumar, and Y. Rabani. Cell-probe lower bounds for the partial match problem. In *Proc. of 35th STOC*, pages 667–672, 2003.

[17] W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz maps into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[18] J. Katz and L. Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 80–86. ACM, 2000.

[19] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. In *Proc. of 30th STOC*, pages 614–623, 1998.

[20] D. Liu. A strong lower bound for approximate nearest neighbor searching. *Inf. Process. Lett.*, 92(1):23–29, 2004.

[21] P. B. Miltersen, N. Nisan, S. Safra, and A. Wigderson. On data structures and asymmetric communication complexity. *J. Comput. Syst. Sci.*, 57(1):37–49, 1998.

[22] R. Motwani, A. Naor, and R. Panigrahy. Lower bounds on locality sensitive hashing. In *Proc. of 22nd SCG*, pages 154–157, 2006.

[23] R. Panigrahy. Entropy based nearest neighbor search in high dimensions. In *Proc. of 17th SODA*, pages 1186–1195, 2006.

[24] M. Pătraşcu. Data (structures). In *Proc. of 49th FOCS*, 2008.

[25] M. Pătraşcu and M. Thorup. Higher lower bounds for near-neighbor and further rich problems. In *Proc. of 47th FOCS*, pages 646–654, 2006.

[26] M. Pătraşcu and M. Thorup. Randomization does not help searching predecessors. In *Proc. of 18th SODA*, pages 555–564, 2007.

[27] R. L. Rivest. *Analysis of Associative Retrieval Algorithms*. PhD thesis, Stanford University, 1974.

[28] R. L. Rivest. Partial-match retrieval algorithms. *SIAM J. Comput.*, 5(1):19–50, 1976.

[29] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proc of the Ninth International Conference on Computer Vision*, page 750, 2003.

[30] A. C. Yao. Should tables be sorted? *J. ACM*, 28(3):615–628, 1981.