Lower bounds on time-space trade-offs for approximate near neighbors

Lower Bounds on Time–Space Trade-Offs for Approximate Near Neighbors

Alexandr Andoni
Columbia

Thijs Laarhoven
IBM Research Zürich

Ilya Razenshteyn
MIT CSAIL

Erik Waingarten
Columbia

August 22, 2016

Abstract

We show tight lower bounds for the entire trade-off between space and query time for the Approximate Near Neighbor search problem. Our lower bounds hold in a restricted model of computation, which captures all hashing-based approaches. In particular, our lower bound matches the upper bound recently shown in [Laa15c] for the random instance on a Euclidean sphere (which we show in fact extends to the entire space $\mathbb{R}^d$ using the techniques from [AR15]).

We also show tight, unconditional cell-probe lower bounds for *one* and *two* probes, improving upon the best known bounds from [PTW10]. In particular, this is the first space lower bound (for any static data structure) for two probes which is *not* polynomially smaller than for one probe. To show the result for two probes, we establish and exploit a connection to locally-decodable codes.

1 Introduction

1.1 Approximate Near Neighbor problem (ANN)

The Near Neighbor Search problem (NNS) is a basic and fundamental problem in computational geometry, defined as follows. We are given a dataset of n points P from a metric space (X, d_X) and a distance threshold $r > 0$. The goal is to preprocess P in order to answer *near neighbor queries*: given a query point $q \in X$, return a dataset point $p \in P$ with $d_X(q, p) \leq r$, or report that there is no such point. The d-dimensional Euclidean $(\mathbb{R}^d, \ell_2)$ and Manhattan $(\mathbb{R}^d, \ell_1)$ metric spaces have received the most attention. Besides its classical applications to similarity search over many types of data (text, audio, images, etc; see [SDI06] for an overview), NNS has been also recently used for cryptanalysis [Laa15a, Laa15b] and optimization [DRT11, HLM15, ZYS16].

The performance of a NNS data structure is often characterized by two key metrics:

- the amount of memory a data structure occupies, and
- the time it takes to answer a query.

All known time-efficient data structures for NNS (e.g., [Cla88, Mei93]) require space exponential in the dimension d, which is prohibitively expensive unless d is very small. To overcome this so-called "curse of dimensionality", researchers proposed the (c, r)-*Approximate* Near Neighbor Search problem, or (c, r)-ANN. In this relaxed version, we are given a dataset P and a distance threshold $r > 0$, as well as an approximation factor $c > 1$. Given a query point q with the promise that there is at least one data point in P within distance at most r from q, the goal is to return a data point $p \in P$ within a distance at most cr from q.

This approximate version of NNS allows efficient data structures with space and query time polynomial in d and query time sublinear in n [KOR00, IM98, Ind01b, Ind01a, GIM99, Cha02, DIIM04, CR04, Pan06, AC09, AI06, TT07, AINR14, AR15, Pag16, Kap15, BDGL16, Laa15c]. In practice, ANN algorithms are often successful for similarity search even when one is interested in exact nearest neighbors [ADI$^+$06, AIL$^+$15]. We refer the reader to [HIM12, AI08, And09] for a survey of the theory of ANN, and [WSSJ14, WLKC15] for a more practical perspective.

In this paper, we study tight time–space trade-offs for ANN. Before stating our results in Section 1.6, we provide more background on the problem.

1.2 Locality-Sensitive Hashing (LSH) and beyond

A classic technique for ANN is *Locality-Sensitive Hashing* (LSH), introduced in 1998 by Indyk and Motwani [IM98, HIM12]. The main idea is to use *random space partitions*, for which a pair of close points (at distance at most r) is more likely to belong to the same part than a pair of far points (at distance more than cr). Given such a partition, the data structure splits the set P according to the partition, and, given a query, retrieves all the data points which belong to the same part as the query. To get a high probability of success, the data structure maintains several partitions and checks all of them during the query stage. LSH yields data structures with space $O(n^{1+\rho} + d \cdot n)$ and query time $O(d \cdot n^{\rho})$. For a particular metric space and approximation c, ρ measures the quality of the random space partition. Usually, $\rho = 1$ for $c = 1$ and $\rho \to 0$ as $c \to \infty$.

Since the introduction of LSH in [IM98], subsequent research established optimal values of the LSH exponent ρ for several metrics of interest, including ℓ_1 and ℓ_2. For the Hamming distance (ℓ_1), the optimal value is $\rho = \frac{1}{c} \pm o(1)$ [IM98, MNP07, OWZ14]. For the Euclidean metric (ℓ_2), it is $\rho = \frac{1}{c^2} \pm o(1)$ [IM98, DIIM04, AI06, MNP07, OWZ14].

More recently, it has been shown that better bounds on ρ are possible if the space partitions are *allowed to depend on the dataset*[1]. That is, the algorithm is based on an observation that every dataset has some structure to exploit. This more general framework of *data-dependent LSH* yields $\rho = \frac{1}{2c-1} + o(1)$ for the ℓ_1 distance, and $\rho = \frac{1}{2c^2-1} + o(1)$ for ℓ_2 [AINR14, Raz14, AR15]. Moreover, these bounds are known to be tight for data-dependent LSH [AR16].

[1] Let us note that the idea of data-dependent random space partitions is ubiquitous in practice, see, e.g., [WSSJ14, WLKC15] for a survey. But the perspective in practice is that the given datasets are not "worst case" and hence it is possible to adapt to the additional "nice" structure.

1.3 Random instances: the hardest instances

At the core of the optimal data-dependent LSH data structure for ℓ_1 from [AR15] is an algorithm that handles the following random instances of ANN over Hamming space (also known as the *light bulb problem* in literature [Val88] in the *off-line* setting).

- The dataset P consists of n independent uniformly random points from $\{-1,1\}^d$, where $d = \omega(\log n)$;
- A query q is generated by choosing a uniformly random data point $p \in P$, and flipping each coordinate of p with probability $\frac{1}{2c}$ independently;
- The goal for a data structure is to recover the data point p from the query point q.

At a high level, the data structure from [AR15] proceeds in two steps:

- it designs a (data-independent) LSH family that handles the random instance, and
- it develops a reduction from a worst-case instance to several instances that essentially look like random instances.

Thus, random instances are the hardest for ANN. On the other hand, random instances have been used for the lower bounds on ANN (more on this below), since they must be handled by *any* data structure for $\left(c, \frac{d}{2c} + o(1)\right)$-ANN over ℓ_1.

1.4 Time–space trade-offs

LSH gives data structures with space around $n^{1+\rho}$ and query time around n^{ρ}. Since early results on LSH, the natural question has been whether one can trade space for time and vice versa. One can achieve *polynomial* space with *poly-logarithmic* query time [IM98, KOR00], as well as *near-linear* space with *sublinear* query time [Ind01a]. In the latter regime, [Pan06, Kap15] and, most recently, [Laa15c] gave subsequent improvements. We point out that the near-linear space regime is especially relevant for practice: e.g., see [LJW⁺07, AIL⁺15] for practical versions of the above theoretical results.

For random instances, the best known trade-off is from [Laa15c]:

Theorem 1.1 (Theorem 1 of [Laa15c]). *Let $c \in (1, \infty)$. One can solve $\left(c, \frac{\sqrt{2}}{c} + o(1)\right)$-ANN on the unit sphere $S^{d-1} \subset \mathbb{R}^d$ equipped with ℓ_2 norm with query time $O(d \cdot n^{\rho_q + o(1)})$, and space $O(n^{1+\rho_u+o(1)} + d \cdot n)$ where*

$$c^2\sqrt{\rho_q} + (c^2-1)\sqrt{\rho_u} = \sqrt{2c^2-1}. \tag{1}$$

This data structure can handle the random Hamming instances introduced in Section 1.3 via a standard reduction. The resulting time–space trade-off is:

$$c\sqrt{\rho_q} + (c-1)\sqrt{\rho_u} = \sqrt{2c-1}. \tag{2}$$

For the sake of illustration, consider the setting of the Hamming distance and approximation $c = 2$. The optimal data-dependent LSH from [AR15] gives space $n^{4/3+o(1)}$ and query time $n^{1/3+o(1)}$. For random instances, the above bound (2) gives the same bound as well as a smooth interpolation between the following extremes: space $n^{1+o(1)}$ and query time $n^{3/4+o(1)}$, and space $n^{4+o(1)}$ and query time $n^{o(1)}$.

The algorithm from [Laa15c] can be applied to the entire ℓ_2 sphere (and hence, via standard reductions à la [Val15, Algorithm 25], to the entire space $\mathbb{R}^d$). However, this direct extension degrades the quality of the (ρ_q, ρ_u) trade-off to essentially those corresponding to the classical LSH bounds (e.g., for $\rho_q = \rho_u$, obtaining $\rho_q = \rho_u = 1/c^2 + o(1)$, instead of the optimal $\rho_q = \rho_u = 1/(2c^2 - 1) + o(1)$). Nonetheless, it is possible to apply the worst-case–to–random-case reduction from [AR15] in order to extend Theorem 1.1 to the entire $\mathbb{R}^d$ with the same trade-off as (1) (see Appendices B and C for details).

Furthermore, we note that all algorithms for ℓ_2 extend to ℓ_p, for $p \in (1, 2)$, with c^2 being replaced with c^p in the expressions for the exponents (ρ_q, ρ_u). This follows from the reduction shown in [Ngu14, Section 5.5].

1.5 Lower bounds

Lower bounds for NNS and ANN have also received much attention. Such lower bounds are almost always obtained in the *cell-probe* model [MNSW98, Mil99]. In the cell-probe model one measures the *number of memory cells* the query algorithm accesses. Despite a number of success stories, high cell-probe lower bounds are notoriously hard to prove. In fact, there are few techniques for proving high cell-probe lower bounds, for any (static) data structure problem. For ANN in particular, we have no viable techniques to prove $\omega(\log n)$ query time lower bounds. Due to this state of affairs, one may rely on *restricted* models of computation, which nevertheless capture existing upper bounds.

Early lower bounds for NNS were shown for data structures in the *exact* or *deterministic* settings [BOR99, CCGL99, BR02, Liu04, JKKR04, CR04, PT06, Yin16]. In [CR04, LPY16] an almost tight cell-probe lower bound is shown for the randomized Approximate *Nearest* Neighbor Search under the ℓ_1 distance. In the latter problem, there is no distance threshold r, and instead the goal is to find a data point that is not much further than the *closest* data point. This twist is the main source of hardness, and the result is not applicable to the ANN problem as introduced above.

There are few results that show lower bounds for *randomized* data structures for the *approximate* near neighbor problem (the setting studied in the present paper). The first such result [AIP06] shows that any data structure that solves $(1+\varepsilon, r)$-ANN for ℓ_1 or ℓ_2 using t cell probes requires space $n^{\Omega(1/t\varepsilon^2)}$.[2] This result shows that the algorithms of [IM98, KOR00] are tight up to constants in the exponent for $t = O(1)$.

In [PTW10] (following up on [PTW08]), the authors introduce a general framework for proving lower bounds for ANN under any metric. They show that lower bounds for ANN are implied by the *robust expansion* of the underlying metric space. Using this framework, [PTW10] show that (c, r)-ANN using t cell probes requires space $n^{1+\Omega(1/tc)}$ for the Hamming distance and $n^{1+\Omega(1/tc^2)}$ for the Euclidean distance (for every $c > 1$).

Lower bounds were also shown for other metrics. For the ℓ_∞ distance, [ACP08] show a lower bound for deterministic ANN data structures, matching the upper bound of [Ind01b] for decision trees. This lower bound was later generalized to randomized data structures [PTW10, KP12]. A recent result [AV15] adapts the framework of [PTW10] to Bregman divergences. There are also lower bounds for restricted models: for LSH [MNP07, OWZ14, AIL$^+$15] and for data-dependent LSH [AR16]. We note that essentially all of the aforementioned lower bounds for ANN under ℓ_1 [AIP06, PTW10, MNP07, AIL$^+$15, AR16] use the *random instance* defined in Section 1.3 as a hard distribution.

1.6 Our results

In this paper, we show both new cell-probe and restricted lower bounds for (c, r)-ANN. In all cases our lower bounds match the upper bounds from [Laa15c]. Our lower bounds use the random instance from Section 1.3 as a hard distribution. Via a standard reduction, we obtain similar hardness results for ℓ_p with $1 < p \leq 2$ (with c being replaced by c^p).

1.6.1 One cell probe

First, we show a tight (up to $n^{o(1)}$ factors) lower bound on the space needed to solve ANN for a random instance, for query algorithms that use a *single* cell probe. More formally, we prove the following theorem:

Theorem 1.2 (Section 4)**.** *Any data structure that:*

- *solves (c, r)-ANN for the Hamming random instance (as defined in Section 1.3) with probability $2/3$,*
- *operates on memory cells of size $n^{o(1)}$,*
- *for each query, looks up a* single *cell,*

must use at least $n^{\left(\frac{c}{c-1}\right)^2 - o(1)}$ words of memory.

[2] The correct dependence on $1/\varepsilon$ requires a stronger LSD lower bound from [Pǎt11].

The space lower bound matches the upper bound from [Laa15c] (see also Appendix C) for $\rho_q = 0$. The previous best lower bound from [PTW10] for a single probe was weaker by a polynomial factor.

We prove Theorem 1.2 by computing tight bounds on the robust expansion of a hypercube $\{-1, 1\}^d$ as defined in [PTW10]. Then, we invoke a result from [PTW10], which yields the desired cell probe lower bound. We obtain estimates on the robust expansion via a combination of the hypercontractivity inequality and Hölder's inequality [O'D14]. Equivalently, one could obtain the same bounds by an application of the Generalized Small-Set Expansion Theorem of [O'D14].

1.6.2 Two cell probes

To state our results for two cell probes, we first define the *decision* version of ANN (first introduced in [PTW10]). Suppose that with every data point $p \in P$ we associate a bit $x_p \in \{0, 1\}$. A new goal is: given a query $q \in \{-1, 1\}^d$ which is at distance at most r from a data point $p \in P$, and assuming that $P \setminus \{p\}$ is at distance more than cr from q, return correct x_p with probability at least 2/3. It is easy to see that any algorithm for (c, r)-ANN would solve this decision version.

We prove the following lower bound for data structures making only two cell probes per query.

Theorem 1.3 (see Section 6)**.** *Any data structure that:*

- *solves the decision ANN for the random instance (Section 1.3) with probability* 2/3*,*
- *operates on memory cells of size* $o(\log n)$*,*
- *accesses at most two cells for each query,*

must use at least $n^{\left(\frac{c}{c-1}\right)^2 - o(1)}$ *words of memory.*

Informally speaking, we show that the second cell probe cannot improve the space bound by more than a subpolynomial factor. To the best of our knowledge, this is the first lower bound for the space of *any* static data structure problem without a polynomial gap between $t = 1$ and $t \geq 2$ cell-probes. Previously, the highest ANN lower bound for two queries was weaker by a polynomial factor [PTW10]. (This remains the case even if we plug the tight bound on the robust expansion into the framework of [PTW10].) Thus, in order to obtain a higher lower bound for $t = 2$, we need to depart from the framework of [PTW10].

Our proof establishes a connection between two-query data structures (for the decision version of ANN), and two-query locally-decodable codes (LDC). A possibility of such a connection was suggested in [PTW10]. In particular, we show that a data structure violating the lower bound from Theorem 1.3 implies an efficient two-query LDC, which contradicts known LDC lower bounds from [KdW04, BRdW08].

The first lower bound for unrestricted two-query LDCs was proved in [KdW04] via a quantum argument. Later, the argument was simplified and made *classical* in [BRdW08]. It turns out that for our lower bound, we need to resort to the original quantum argument of [KdW04] since it has a better dependence on the noise rate a code is able to tolerate. During the course of our proof, we do not obtain a full-fledged LDC, but rather an object which can be called an *LDC on average*. For this reason, we are unable to use [KdW04] as a black box but rather adapt their proof to the average case.

Finally, we point out an important difference with Theorem 1.2: in Theorem 1.3 we allow words to be merely of size $o(\log n)$ (as opposed to $n^{o(1)}$). Nevertheless, for the *decision version* of ANN the upper bounds from [Laa15c] hold even for such "tiny" words. In fact, our techniques do not allow us to handle words of size $\Omega(\log n)$ due to the weakness of known lower bounds for two-query LDC for *large alphabets*. In particular, our argument can not be pushed beyond word size $2^{\widetilde{\Theta}(\sqrt{\log n})}$ *in principle*, since this would contradict known constructions of two-query LDCs over large alphabets [DG15]!

1.6.3 The general time–space trade-off

Finally, we prove conditional lower bound on the entire time–space trade-off that is tight (up to $n^{o(1)}$ factors), matching the upper bound from [Laa15c] (see also Appendix C). Note that—since we show polynomial query time lower bounds—proving similar lower bounds *unconditionally* is far beyond the current reach of techniques, modulo major breakthrough in cell probe lower bounds.

Our lower bounds are proved in the following model, which can be loosely thought of comprising all hashing-based frameworks we are aware of:

Definition 1.4. *A* list-of-points data structure *for the ANN problem is defined as follows:*

- *We fix (possibly randomly) sets $A_i \subseteq \{0,1\}^d$, for $i = 1 \ldots m$; also, with each possible query point $q \in \{0,1\}^d$, we associate a (random) set of indices $I(q) \subseteq [m]$;*
- *For a given dataset P, the data structure maintains m lists of points $L_1, L_2, \ldots, L_m$, where $L_i = P \cap A_i$;*
- *On query q, we scan through each list L_i for $i \in I(q)$ and check whether there exists some $p \in L_i$ with $\|p - q\|_1 \le cr$. If it exists, return p.*

The total space is defined as $s = m + \sum_{i=1}^{m} |L_i|$ and the query time is $t = |I(q)| + \sum_{i \in I(q)} |L_i|$.

For this model, we prove the following theorem.

Theorem 1.5 (see Section 5)**.** *Consider any list-of-points data structure for (c,r)-ANN for random instances of n points in the d-dimensional Hamming space with $d = \omega(\log n)$, which achieves a total space of $n^{1+\rho_u+o(1)}$, and has query time $n^{\rho_q - o(1)}$, for $2/3$ success probability. Then it must hold that:*

$$c\sqrt{\rho_q} + (c-1)\sqrt{\rho_u} \ge \sqrt{2c-1}. \tag{3}$$

We note that our model captures the basic hashing-based algorithms, in particular most of the known algorithms for the high-dimensional ANN problem [KOR00, IM98, Ind01b, Ind01a, GIM99, Cha02, DIIM04, Pan06, AC09, AI06, Pag16, Kap15], including the recently proposed Locality-Sensitive Filters scheme from [BDGL16, Laa15c]. The only data structures not captured are the data-dependent schemes from [AINR14, Raz14, AR15]; we conjecture that the natural extension of the list-of-point model to data-dependent setting would yield the same lower bound. In particular, Theorem 1.5 uses the random instance as a hard distribution, for which being data-dependent seems to offer no advantage. Indeed, a data-dependent lower bound in the standard LSH regime (where $\rho_q = \rho_s$) has been recently shown in [AR16], and matches (3) for $\rho_s = \rho_q$.

1.7 Other related work

There has been a lot of recent algorithmic advances on high-dimensional similarity search, including better algorithms for the closest pair problem[3] [Val15, AW15, KKK16, KKKÓ16], locality-sensitive filters [BDGL16, Laa15c], LSH without false negatives [Pag16, PP16], to name just a few.

2 Preliminaries

We introduce a few definitions from [PTW10] to setup the nearest neighbor search problem for which we show lower bounds.

Definition 2.1. *The goal of the (c, r)-approximate nearest neighbor problem with failure probability δ is to construct a data structure over a set of points $P \subset \{0, 1\}^d$ supporting the following query: given any point q such that there exists some $p \in P$ with $\|q - p\|_1 \leq r$, report some $p' \in P$ where $\|q - p'\|_1 \leq cr$ with probability at least $1 - \delta$.*

Definition 2.2 ([PTW10]). *In the Graphical Neighbor Search problem (GNS), we are given a bipartite graph $G = (U, V, E)$ where the dataset comes from U and the queries come from V. The dataset consists of pairs $P = \{(p_i, x_i) \mid p_i \in U, x_i \in \{0, 1\}, i \in [n]\}$. On query $q \in V$, if there exists a unique p_i with $(p_i, q) \in E$, then we want to return x_i.*

We will sometimes use the GNS problem to prove lower bounds on (c, r)-ANN as follows: we build a GNS graph G by taking $U = V = \{0, 1\}^d$, and connecting two points $u \in U, v \in V$ iff they are at a distance at most r (see details in [PTW10]). We will also need to make sure that in our instances q is not closer than cr to other points except the near neighbor.

2.1 Robust Expansion

The following is the fundamental property of a metric space that [PTW10] use to prove lower bounds.

[3] These can be seen as the off-line version of NNS/ANN.

Definition 2.3 (Robust Expansion [PTW10])**.** *For a GNS graph $G = (U, V, E)$, fix a distribution e on $E \subset U \times V$, and let μ be the marginal on U and η be the marginal on V. For $\delta, \gamma \in (0, 1]$, the robust expansion $\Phi_r(\delta, \gamma)$ is defined as follows:*

$$\Phi_r(\delta, \gamma) = \min_{A \subset V : \eta(A) \leq \delta} \min_{B \subset U : \frac{e(A \times B)}{e(A \times V)} \geq \gamma} \frac{\mu(B)}{\eta(A)}.$$

2.2 Locally Decodable Codes

Finally, our 2-cell lower bounds uses results on Locally Decodable Codes (LDCs). We present the standard definitions and results on LDCs below, although we will need a weaker definition (and stronger statement) for our 2-query lower bound in Section 6.

Definition 2.4. *A (t, δ, ε) locally decodable code (LDC) encodes n-bit strings $x \in \{0, 1\}^n$ into m-bit codewords $C(x) \in \{0, 1\}^m$ such that, for each $i \in [n]$, the bit x_i can be recovered with probability $\frac{1}{2} + \varepsilon$ while making only t queries into $C(x)$, even if the codeword is arbitrarily modified (corrupted) in δm bits.*

We will use the following lower bound on the size of the LDCs.

Theorem 2.5 (Theorem 4 from [KdW04])**.** *If $C : \{0, 1\}^n \to \{0, 1\}^m$ is a $(2, \delta, \varepsilon)$-LDC, then*

$$m \geq 2^{\Omega(\delta \varepsilon^2 n)}. \tag{4}$$

3 Robust Expansion of the Hamming Space

The goal of this section is to compute tight bounds for the robust expansion $\Phi_r(\delta, \gamma)$ in the Hamming space of dimension d, as defined in the preliminaries. We use these bounds for all of our lower bounds in the subsequent sections.

We use the following model for generating dataset points and queries (which is essentially the random instance from the introduction).

Definition 3.1. *For any $x \in \{-1, 1\}^n$, $N_\sigma(x)$ is a probability distribution over $\{-1, 1\}^n$ representing the neighborhood of x. We sample $y \sim N_\sigma(x)$ by choosing $y_i \in \{-1, 1\}$ for each coordinate $i \in [d]$. With probability σ, $y_i = x_i$. With probability $1 - \sigma$, y_i is set uniformly at random.*

Given any Boolean function $f : \{-1, 1\}^n \to \mathbb{R}$, the function $T_\sigma f : \{-1, 1\}^n \to \mathbb{R}$ is

$$T_\sigma f(x) = \mathop{\mathbb{E}}_{y \sim N_\sigma(x)} [f(y)] \tag{5}$$

In the remainder of this section, will work solely on the Hamming space $V = \{-1, 1\}^d$. We let

$$\sigma = 1 - \frac{1}{c} \qquad\qquad d = \omega(\log n)$$

and μ will refer to the uniform distribution over V.

The choice of σ allows us to make the following observations. A query is generated as follows: we sample a dataset point x uniformly at random and then generate the query y by sampling $y \sim N_\sigma(x)$. From the choice of σ, $d(x, y) \leq \frac{d}{2c}(1 + o(1))$ with high probability. In addition, for every other point in the dataset $x' \neq x$, the pair (x', y) is distributed as two uniformly random points (even though $y \sim N_\sigma(x)$, because x is randomly distributed). Therefore, by taking a union-bound over all dataset points, we can conclude that with high probability, $d(x', y) \geq \frac{d}{2}(1 - o(1))$ for each $x' \neq x$.

Given a query y generated as described above, we know there exists a dataset point x whose distance to the query is $d(x, y) \leq \frac{d}{2c}(1 + o(1))$. Every other dataset point lies at a distance $d(x', y) \geq \frac{d}{2}(1 - o(1))$. Therefore, the two distances are a factor of $c - o(1)$ away.

The following lemma is the main result of this section, and we will reference this lemma in subsequent sections.

Lemma 3.2 (Robust expansion). *In the Hamming space equipped with the Hamming norm, for any $p, q \in [1, \infty)$ where $(q - 1)(p - 1) = \sigma^2$, any $\gamma \in [0, 1]$ and $m \geq 1$,*

$$\Phi_r\left(\frac{1}{m}, \gamma\right) \geq \gamma^q m^{1+\frac{q}{p}-q} \tag{6}$$

The robust expansion comes from a straight forward application from small-set expansion. In fact, one can easily prove tight bounds on robust expansion via the following lemma:

Theorem 3.3 (Generalized Small-Set Expansion Theorem, [O'D14]). *Let $0 \leq \sigma \leq 1$. Let $A, B \subset \{-1, 1\}^n$ have volumes $\exp(-\frac{a^2}{2})$ and $\exp(-\frac{b^2}{2})$ and assume $0 \leq \sigma a \leq b \leq a$. Then*

$$\Pr_{\substack{(x,y) \\ \sigma-correlated}}[x \in A, y \in B] \leq \exp\left(-\frac{1}{2}\frac{a^2 - 2\sigma ab + b^2}{1 - \sigma^2}\right)$$

However, we compute the robust expansion via an application of the Bonami-Beckner Inequality and Hölder's inequality. This computation gives us a bit more flexibility with respect to parameters which will become useful in subsequent sections. We now recall the necessary tools.

Theorem 3.4 (Bonami-Beckner Inequality [O'D14]). *Fix $1 \leq p \leq q$ and $0 \leq \sigma \leq \sqrt{(p - 1)/(q - 1)}$. Any Boolean function $f : \{-1, 1\}^n \to \mathbb{R}$ satisfies*

$$\|T_\sigma f\|_q \leq \|f\|_p \tag{7}$$

Theorem 3.5 (Hölder's Inequality). *Let $f : \{-1, 1\}^n \to \mathbb{R}$ and $g : \{-1, 1\}^n \to \mathbb{R}$ be arbitrary Boolean functions. Fix $s, t \in [1, \infty)$ where $\frac{1}{s} + \frac{1}{t} = 1$. Then*

$$\langle f, g \rangle \leq \|f\|_s \|g\|_t \tag{8}$$

We will let f and g be indicator functions for two sets A and B and use a combination of the Bonami-Beckner Inequality and Hölder's Inequality to lower bound the robust expansion. The operator T_σ will applied to f will measure the neighborhood of set A. We will compute an upper bound on the correlation of the neighborhood of A and B (referred to as γ) with respect to the volumes of A and B, and the expression will give a lower bound on robust expansion.

We also need the following lemma.

Lemma 3.6. *Let $p, q \in [1, \infty)$, where $(p-1)(q-1) = \sigma^2$ and $f, g : \{-1,1\}^d \to \mathbb{R}$ be two Boolean functions. Then*

$$\langle T_\sigma f, g \rangle \leq \|f\|_p \|g\|_q$$

Proof. We first apply Hölder's Inequality to split the inner-product into two parts. Then we apply the Bonami-Beckner Inequality to each part.

$$\begin{aligned} \langle T_\sigma f, f \rangle &= \langle T_{\sqrt{\sigma}} f, T_{\sqrt{\sigma}} g \rangle && (9) \\ &\leq \|T_{\sqrt{\sigma}} f\|_s \|T_{\sqrt{\sigma}} g\|_t && (10) \end{aligned}$$

We pick the parameters $s = \dfrac{p-1}{\sigma} + 1$ and $t = \dfrac{s}{s-1}$, so $\frac{1}{s} + \frac{1}{t} = 1$. Note that $p \leq s$ because $\sigma < 1$ and $p \geq 1$ because $(p-1)(q-1) = \sigma^2 \leq \sigma$. We have

$$q \leq \frac{\sigma}{p-1} + 1 = t. \tag{11}$$

In addition,

$$\sqrt{\frac{p-1}{s-1}} = \sqrt{\sigma} \qquad \sqrt{\frac{q-1}{t-1}} = \sqrt{(q-1)(s-1)} \tag{12}$$

$$= \sqrt{\frac{(q-1)(p-1)}{\sigma}} = \sqrt{\sigma}. \tag{13}$$

So we can apply the Bonami-Beckner Inequality to both norms. We obtain

$$\|T_{\sqrt{\sigma}} f\|_s \|T_{\sqrt{\sigma}} g\|_t \leq \|f\|_p \|g\|_q \tag{14}$$

□

We are now ready to prove Lemma 3.2.

Proof of Lemma 3.2. We use Lemma 3.6 and the definition of robust expansion. For any two sets $A, B \subset V$, let $a = \frac{1}{2^d}|A|$ and $b = \frac{1}{2^d}|B|$ be the measure of set A and B with respect to the uniform distribution. We refer to $\mathbf{1}_A : \{-1,1\}^d \to \{0,1\}$ and $\mathbf{1}_B : \{-1,1\}^d \to \{0,1\}$ as the indicator functions for A and B.

$$\gamma = \Pr_{x\sim\mu, y\sim N_\sigma(x)}[x \in B \mid y \in A] \tag{15}$$

$$= \frac{1}{a}\langle T_\sigma \mathbf{1}_A, \mathbf{1}_B\rangle \tag{16}$$

$$\leq a^{\frac{1}{p}-1} b^{\frac{1}{q}} \tag{17}$$

Therefore, $\gamma^q a^{q-\frac{q}{p}} \leq b$. Let A and B be the minimizers of $\frac{b}{a}$ satisfying (15) and $a \leq \frac{1}{m}$.

$$\Phi_r\left(\frac{1}{m}, \gamma\right) = \frac{b}{a} \tag{18}$$

$$\geq \gamma^q a^{q-\frac{q}{p}-1} \tag{19}$$

$$\geq \gamma^q m^{1+\frac{q}{p}-q}. \tag{20}$$

□

4 Tight Lower Bounds for 1 Cell Probe Data Structures

In this section, we prove Theorem 1.2. Our proof relies on the main result of [PTW10] for the GNS problem:

Theorem 4.1 (Theorem 1.5 [PTW10])**.** *There exists an absolute constant γ such that the following holds. Any randomized algorithm for a weakly independent instance of GNS which is correct with probability greater than $\frac{1}{2}$ must satisfy*

$$\frac{m^t w}{n} \geq \Phi_r\left(\frac{1}{m^t}, \frac{\gamma}{t}\right) \tag{21}$$

Proof of Theorem 1.2. The bound comes from a direct application of the computation of $\Phi_r(\frac{1}{m}, \gamma)$ in Lemma 3.2 to the bound in Theorem 4.1. Setting $t = 1$ in Theorem 4.1, we obtain

$$mw \geq n \cdot \Phi_r\left(\frac{1}{m}, \gamma\right) \tag{22}$$

$$\geq n\gamma^q m^{1+\frac{q}{p}-q} \tag{23}$$

for some $p, q \in [1, \infty)$ and $(p-1)(q-1) = \sigma^2$. Rearranging the inequality, we obtain

$$m \geq \frac{\gamma^{\frac{p}{p-1}} n^{\frac{p}{pq-q}}}{w^{\frac{p}{pq-q}}} \tag{24}$$

Let $p = 1 + \frac{\log\log n}{\log n}$, and $q = 1 + \sigma^2 \frac{\log n}{\log\log n}$. Then

$$m \geq n^{\frac{1}{\sigma^2} - o(1)}. \tag{25}$$

Since $\sigma = 1 - \frac{1}{c}$ and $w = n^{o(1)}$, we obtain the desired result. □

Corollary 4.2. *Any 1 cell probe data structures with cell size $O(\log n)$ for c-approximate nearest neighbors on the sphere in ℓ_2 needs $n^{1 + \frac{2c^2 - 1}{(c^2-1)^2} - o(1)}$ many cells.*

Proof. Each point in the Hamming space $\{-1, 1\}^d$ (after scaling by $\frac{1}{\sqrt{d}}$) can be thought of as lying on the unit sphere. If two points are a distance r apart in the Hamming space, then they are $2\sqrt{r}$ apart on the sphere with ℓ_2 norm. Therefore a data structure for a c^2-approximation on the sphere gives a data structure for a c-approximation in the Hamming space. □

5 Lower Bounds for List-of-Points Data Structures

In this section we prove Theorem 1.5, i.e., a tight lower bound against data structure that fall inside the "list-of-points" model, as defined in Def. 1.4.

Recall that $A_i \subset V$ is the subset of dataset points which get placed in L_i. Let $B_i \subset V$ the subset of query points which query L_i, this is well defined, since $B_i = \{v \in V \mid i \in I(v)\}$. Suppose we sample a random dataset point $u \sim V$ and then a random query point v from the neighborhood of u. Let

$$\gamma_i = \Pr[v \in B_i \mid u \in A_i] \tag{26}$$

and let $s_i = \mu(A_i)$.

On instances where n dataset points $\{u_i\}_{i=1}^n$ are drawn randomly, and a query v is drawn from the neighborhood of a random dataset point, we can exactly characterize the query time.

$$T = \sum_{i=1}^m \mathbf{1}\{v \in B_i\} \left(1 + \sum_{j=1}^n \mathbf{1}\{u_j \in A_i\}\right) \tag{27}$$

$$\mathbb{E}[T] = \sum_{i=1}^m \mu(B_i) + \sum_{i=1}^m \gamma_i \mu(A_i) + (n-1) \sum_{i=1}^m \mu(B_i)\mu(A_i) \tag{28}$$

$$\geq \sum_{i=1}^m \Phi_r(s_i, \gamma_i) s_i + \sum_{i=1}^m s_i \gamma_i + (n-1) \sum_{i=1}^m \Phi_r(s_i, \gamma_i) s_i^2 \tag{29}$$

Since the data structure succeeds with probability γ, it must be the case that

$$\sum_{i=1}^{m} s_i \gamma_i \geq \gamma = \Pr_{j\sim[n], v\sim N(u_j)}[\exists i \in [m] : v \in B_i, u_j \in A_i] \tag{30}$$

And since we use at most space $O(s)$,

$$n \sum_{i=1}^{m} s_i \leq O(s) \tag{31}$$

From Lemma 3.2, for any $p, q \in [1, \infty)$ where $(p-1)(q-1) = \sigma^2$ where $\sigma = 1 - \frac{1}{c}$,

$$\mathbb{E}[T] \geq \sum_{i=1}^{m} s_i^{q-\frac{q}{p}} \gamma_i^q + (n-1) \sum_{i=1}^{m} s_i^{q-\frac{q}{p}+1} \gamma_i^q + \gamma \tag{32}$$

$$\gamma \leq \sum_{i=1}^{m} s_i \gamma_i \tag{33}$$

$$O\left(\frac{s}{n}\right) \geq \sum_{i=1}^{m} s_i \tag{34}$$

We set $S = \{i \in [m] : s_i \neq 0\}$ and for $i \in S$, $v_i = s_i \gamma_i$.

$$\mathbb{E}[T] \geq \sum_{i\in S} v_i^q \left(s_i^{-\frac{q}{p}} + (n-1) s_i^{-\frac{q}{p}+1} \right) \tag{35}$$

$$\geq \sum_{i\in S} \left(\frac{\gamma}{|S|}\right)^q \left(s_i^{-\frac{q}{p}} + (n-1) s_i^{-\frac{q}{p}+1} \right) \tag{36}$$

where we used the fact $q \geq 1$. Consider

$$F = \sum_{i\in S} \left(s_i^{-\frac{q}{p}} + (n-1) s_i^{-\frac{q}{p}+1} \right) \tag{37}$$

We analyze three cases separately:

- $0 < \rho_u \leq \frac{1}{2c-1}$
- $\frac{1}{2c-1} < \rho_u \leq \dfrac{2c-1}{(c-1)^2}$
- $\rho_u = 0$.

For the first two cases, we let

$$q = 1 - \sigma^2 + \sigma\beta \qquad p = \frac{\beta}{\beta - \sigma} \qquad \beta = \sqrt{\frac{1-\sigma^2}{\rho_u}} \tag{38}$$

Since $0 < \rho_u \leq \dfrac{2c-1}{(c-1)^2}$, one can verify $\beta > \sigma$ and both p and q are at least 1.

Lemma 5.1. *When $\rho_u \leq \frac{1}{2c-1}$, and $s = n^{1+\rho_u}$,*

$$\mathbb{E}[T] \geq \Omega(n^{\rho_q})$$

where ρ_q and ρ_u satisfy Equation 3.

Proof. In this setting, p and q are constants, and $q \geq p$. Therefore, $\frac{q}{p} \geq 1$, so F is convex in all s_i's in Equation 37. So we minimize the sum by taking $s_i = O(\frac{s}{n|S|})$ and substituting in (36),

$$\begin{aligned} \mathbb{E}[T] &\geq \Omega\left(\frac{\gamma^q s^{-q/p+1} n^{q/p}}{|S|^{q-q/p}}\right) && (39) \\ &\geq \Omega(\gamma^q s^{1-q} n^{q/p}) && (40) \end{aligned}$$

since $q - q/p > 0$ and $|S| \leq s$. In addition, p, q and γ are constants, $\mathbb{E}[T] \geq \Omega(n^{\rho_q})$ where

$$\begin{aligned} \rho_q &= (1+\rho_u)(1-q) + \frac{q}{p} && (41) \\ &= (1+\rho_u)(\sigma^2 - \sigma\beta) + \frac{(1-\sigma^2+\sigma\beta)(\beta-\sigma)}{\beta} && (42) \\ &= \left(\sqrt{1-\sigma^2} - \sqrt{\rho_u}\sigma\right)^2 && (43) \\ &= \left(\frac{\sqrt{2c-1}}{c} - \sqrt{\rho_u} \cdot \frac{(c-1)}{c}\right)^2 && (44) \end{aligned}$$

□

Lemma 5.2. *When $\rho_u > \frac{1}{2c-1}$,*

$$\mathbb{E}[T] \geq \Omega(n^{\rho_q})$$

where ρ_q and ρ_u satisfy Equation 3.

Proof. We follow a similar pattern to Lemma 5.1. However, we may no longer assert that F is convex in all s_i's.

$$\frac{\partial F}{\partial s_i} = \left(-\frac{q}{p}\right) s_i^{-\frac{q}{p}-1} + \left(-\frac{q}{p}+1\right)(n-1)s_i^{-\frac{q}{p}} \tag{45}$$

The gradient is zero when each $s_i = \dfrac{q}{(p-q)(n-1)}$. Since $q < p$, this value is positive and $\sum_{i \in S} s_i \leq O\left(\frac{m}{n}\right)$ for large enough n. F is continuous, so it is minimized exactly at that point. So $\mathbb{E}[T] \geq \left(\frac{\gamma}{|S|}\right)^q |S| \left(\frac{q}{(p-q)(n-1)}\right)^{-\frac{q}{p}}$. Again, we maximize $|S|$ to minimize this sum since $q \geq 1$.

Therefore

$$\mathbb{E}[T] \geq \left(\frac{\gamma}{s}\right)^{q} s \left(\frac{q}{(p-q)(n-1)}\right)^{-\frac{q}{p}} \tag{46}$$

Since p, q and γ are constants, $\mathbb{E}[T] \geq \Omega(n^{\rho_q})$ where

$$\rho_q = (1+\rho_u)(1-q) + \frac{q}{p}$$

which is the same expression for ρ_q as in Lemma 5.1. □

Lemma 5.3. *When $\rho_u = 0$ (so $s = O(n)$),*

$$\mathbb{E}[T] \geq n^{\rho_q - o(1)}$$

where $\rho_q = \dfrac{2c-1}{c^2} = 1 - \sigma^2$.

Proof. In this case, although we cannot set p and q as in Equation 38, we let

$$q = 1 + \sigma^2 \cdot \frac{\log n}{\log\log n} \qquad p = 1 + \frac{\log\log n}{\log n}.$$

Since $q > p$, we have

$$\begin{aligned} \mathbb{E}[T] &= \Omega(\gamma^q s^{1-q} n^{\frac{q}{p}}) && (47)\\ &= n^{1-\sigma^2-o(1)} && (48) \end{aligned}$$

giving the desired expression. □

6 Tight Lower Bounds for 2 Cell Probe Data Structures

In this section we prove a cell probe lower bound for ANN for $t = 2$ cell probes as stated in Theorem 1.3.

As in [PTW10], we will prove lower bounds for GNS when $U = V$ with measure μ (see Def. 2.2). We assume there is an underlying graph G with vertex set V. For any particular point $p \in V$, its neighborhood $N(p)$ is the set of points with an edge to p in the graph G.

In the 2-query GNS problem, we have a dataset $P = \{p_i\}_{i=1}^{n} \subset V$ of n points as well as a bit-string $x \in \{0,1\}^n$. We let D denote a data structure with m cells of w bits each. We can think of D as a map $[m] \to \{0,1\}^w$ which holds w bits in each cell. D will depend on the dataset P as well as the bit-string x. The problem says that: given a query point $q \in V$, if there exists a unique neighbor $p_i \in N(q)$ in the dataset, we should return x_i with probability at least $\frac{2}{3}$ after making two cell-probes to D.

Theorem 6.1. *There exists a constant $\gamma > 0$ such that any non-adaptive GNS data structure holding a dataset of $n \geq 1$ points which succeeds with probability $\frac{2}{3}$ using two cell probes and m cells of w bits satisfies*

$$\frac{m \log m \cdot 2^{O(w)}}{n} \geq \Omega\left(\Phi_r\left(\frac{1}{m}, \gamma\right)\right).$$

Theorem 1.3 will follow from Theorem 6.1 together with the robust expansion bound from Lemma 3.2 for the special case when probes to the data structure are non-adaptive. For the rest of this section, we prove Theorem 6.1. We will later show how to reduce adaptive algorithms losing a sub-polynomial factor in the space for $w = o(\frac{\log n}{\log \log n})$ in Section 6.6.3.

At a high-level, we will show that with a "too-good-to-be-true" data structure with small space we can construct a weaker notion of 2-query locally-decodable code (LDC) with small noise rate using the same amount of space[4]. Even though we our notion of LDC is weaker than Def. 2.4, we can use most of the tools for showing 2-query LDC lower bounds from [KdW04]. These arguments use quantum information theory arguments, which are very robust and still work with the 2-query weak LDC we construct.

We note that [PTW10] was the first to suggest the connection between nearest neighbor search and locally-decodable codes. This work represents the first concrete connection which gives rise to better lower bounds.

Proof structure. The proof of Theorem 6.1 proceeds in six steps.

1. First we will use Yao's principle to reduce to the case of deterministic non-adaptive data structures for GNS with two cell-probes. We will give distributions over n-point datasets P, as well as bit-strings x and a query q. After defining these distributions, we will assume the existence of a deterministic data structure which makes two cell-probes non-adaptively and succeeds with probability at least $\frac{2}{3}$ when the inputs are sampled according to the three distributions.

2. We will modify the deterministic data structure in order to get "low-contention" data structures. These are data structures which do not rely on any single cell too much similar to Def. 6.1 in [PTW10]. This will be a simple argument where we increase the space bound by a constant factor to achieve this guarantee.

3. In the third step, we will take a closer look at how the low-contention data structure probes the cells. We will use ideas from [PTW10] to understand how queries neighboring particular dataset points probe various cells of the data structure. We will conclude with finding a fixed n-point dataset P. A constant fraction of the points in the dataset will satisfy the

[4] A 2-query LDC corresponds to LDCs which make two probes to their memory contents. Even though there is a slight ambiguity with the data structure notion of query, we say "2-query LDCs" in order to be consistent with the LDC literature.

following condition: many queries in the neighborhood of these points probe disjoint pairs of cells. Intuitively, this means information about these dataset points must be spread out over various cells.

4. We will show that for the fixed dataset P, we could still recover a constant fraction bits with significant probability even if we corrupt the contents of some cells. This will be the crucial connection between nearest neighbor data structures and LDCs.

5. We will reduce to the case of 1-bit words in order to apply the LDC arguments from [KdW04]. We will increase the number of cells by a factor of 2^w and decrease the probability of success from $\frac{1}{2} + \eta$ to $\frac{1}{2} + \frac{\eta}{2^{2w}}$.

6. Finally, we will design an LDC with weaker guarantees and use the arguments in [KdW04] to prove lower bounds on the space of the weak LDC.

6.1 Deterministic Data Structure

Definition 6.2. *A non-adaptive randomized algorithm R for the GNS problem with two cell-probes is an algorithm specified by the following three components. The data structure preprocesses a dataset $P = \{p_i\}_{i=1}^n$ consisting of n points, as well as a bit-string $x \in \{0,1\}^n$, in order to produce a data structure $D : [m] \to \{0,1\}^w$ which depends on P and x. On a query q, $R(q)$ chooses two indices $(i,j) \in [m]^2$, and specifies a function $f_q : \{0,1\}^w \times \{0,1\}^w \to \{0,1\}$. The output is given as $f_q(D_j, D_k)$. We require that*

$$\Pr_{R,D}[f_q(D_j, D_k) = x_i] \geq \frac{2}{3}$$

whenever $q \in N(p_i)$ and p_i is the unique such neighbor.

Note that the indices (i,j) which R generates to probe the data structure as well as the function f_q is independent of P and x.

Definition 6.3. *We define the following distributions:*

- *Let $\mathcal{P}$ be the distribution over n-point datasets given by sampling n times from our space V uniformly at random.*
- *Let $\mathcal{X}$ be the uniform distribution over $\{0,1\}^n$.*
- *Let $\mathcal{Q}(P)$ be the distribution over queries given by first picking a dataset point $p \in P$ uniformly at random and then picking $q \in N(p)$ uniformly at random.*

Lemma 6.4. *Assume R is a non-adaptive randomized algorithm for GNS using two cell-probes. Then there exists a non-adaptive deterministic algorithm A for GNS using two cell-probes which also produces a data structure $D : [m] \to \{0,1\}^w$ and on query q chooses two indices $j, k \in [m]$*

(again, independently of P and x) to probe in D as well as a function $f_q : \{0,1\}^w \times \{0,1\}^w \to \{0,1\}$ where

$$\Pr_{P \sim \mathcal{P}, x \sim \mathcal{X}, q \sim \mathcal{Q}(P)}[f_q(D_j, D_k) = x_i] \geq \frac{2}{3}.$$

Proof. The following is a direct application of Yao's principle to the success probability of the algorithm. By assumption, there exists a distribution over algorithms which can achieve probability of success at least $\frac{2}{3}$ for any single query. Therefore, for the fixed distributions $\mathcal{P}, \mathcal{X}$, and $\mathcal{Q}$, there exists a deterministic algorithm achieving at least the same success probability. □

In order to simplify notation, for any algorithm A, we let $A^D(q)$ denote output of the algorithm. When we write $A^D(q)$, we assume that $A(q)$ outputs a pair of indices (j,k) as well as the function $f_q : \{0,1\}^w \times \{0,1\}^w \to \{0,1\}$, and the algorithm outputs $f_q(D_j, D_k)$. For any fixed dataset $P = \{p_i\}_{i=1}^n$ and bit-string $x \in \{0,1\}^n$, we have

$$\Pr_{q \sim N(p_i)}[A^D(q) = x_i] = \Pr_{q \sim N(p_i)}[f_q(D_j, D_k) = x_i]$$

by definition. This allows us to succinctly state the probability of correctness when the query is a neighbor of p_i without caring about the specific cells the algorithm probes or the function f_q the algorithm uses to make its decision.

The important thing to note is that the contents of the data structure D may depend on the dataset P and the bit-string x. However, the algorithm A which produces D as well as the indexes for the probes to D for any query point is deterministic.

From now on, we will assume the existence of a non-adaptive deterministic algorithm A with success probability at least $\frac{2}{3}$ using m cells of width w. The success probability is taken over the random choice of the dataset $P \sim \mathcal{P}$, $x \sim \mathcal{X}$ and $q \sim \mathcal{Q}(P)$.

6.2 Making Low-Contention Data Structures

For any $t \in \{1,2\}$ and $j \in [m]$, let $A_{t,j}$ be the set of queries which probe cell j at the t-th probe of algorithm A. These sets are well defined independently of the dataset P and the bit-string x. In particular, we could write

$$A_{t,j} = \{q \in V \mid A \text{ probes cell } j \text{ in probe } t \text{ when querying } q\ \}$$

by running the "probing" portion of the algorithm without the need to specify a dataset P or bit-string x. We could write down $A_{t,j}$ by simply trying every query point q and seeing which cells the algorithm probes.

In other words, since the algorithm is deterministic, the probing portion of algorithm A is completely specified by two collections $\mathcal{A}_1 = \{A_{1,j}\}_{j \in [m]}$ and $\mathcal{A}_2 = \{A_{2,j}\}_{j \in [m]}$ as well as the function f_q. $\mathcal{A}_1$ and $\mathcal{A}_2$ are two partitions of the query space V. On query q, if $q \in A_{t,j}$, we make the t-th probe to cell j. We output the value of f_q after observing the contents of the cells.

We now define the notion of low-contention data structures, which informally requires the data structure not rely on any one particular cell too much, namely no $A_{t,j}$ is too large.

Definition 6.5. *A deterministic non-adaptive algorithm A using m cells has* low contention *if every set $\mu(A_{t,j}) \leq \frac{1}{m}$ for $t \in \{1, 2\}$ and $j \in [m]$.*

We now use the following lemma to argue that up to a small increase in space, a data structure can be made low-contention.

Lemma 6.6. *Suppose A is a deterministic non-adaptive algorithm for GNS with two cell-probes using m cells, then there exists an deterministic non-adaptive algorithm A' for GNS with two cell-probes using $3m$ cells which succeeds with the same probability and has low contention.*

Proof. We first handle $\mathcal{A}_1$ and then $\mathcal{A}_2$.

Suppose $\mu(A_{1,j}) \geq \frac{1}{m}$, then we partition $A_{1,j}$ into enough parts $\{A_{1,k}^{(j)}\}_k$ of size $\frac{1}{m}$. There will be at most one set with measure between 0 and $\frac{1}{m}$. For each of part $A_{1,k}^{(j)}$ of the partition, we make a new cell j_k with the same contents as cell j. When a query lies inside $A_{1,k}^{(j)}$ we probe the new cell j_k. From the data structure side, the cell contents are replicated for all additional cells.

The number of cells in this data structure is at most $2m$, since there can be at most m cells of size $\frac{1}{m}$ and for each original cell, we have only one cell with small measure. Also, keep in mind that we have not modified the sets in $\mathcal{A}_2$, and thus there is at most m cells for which $\mu(A_{2,j}) \geq \frac{1}{m}$.

We do the same procedure for the second collection $\mathcal{A}_2$. If some $\mu(A_{2,j}) \geq \frac{1}{m}$, we partition that cell into multiple cells of size exactly $\frac{1}{m}$, with one extra small cell. Again, the total number of cells will be m for dividing the heavy cells in the second probe, and at most m for the lighter cells in the second probe.

We have added m cells in having $\mu(A_{1,j}) \leq \frac{1}{m}$ for all $j \in [m]$, and added at most m cells in order to make $\mu(A_{2,j}) \leq \frac{1}{m}$ for all $j \in [m]$. Therefore, we have at most $3m$ cells. Additionally, the contents of the cells remain the same, so the algorithm succeeds with the same probability. □

Given Lemma 6.6, we will assume that A is a deterministic non-adaptive algorithm for GNS with two cell-probes using m cells which has low contention. The extra factor of 3 in the number of cells will be pushed into the asymptotic notation.

6.3 Datasets which shatter

We fix some $\gamma > 0$ which can be thought of as a sufficiently small constant.

Definition 6.7 (Weak-shattering [PTW10]). *We say a partition $A_1, \dots, A_m$ of V (K, γ)-weakly shatters a point p if*

$$\sum_{i \in [m]} \left(\mu(A_i \cap N(p)) - \frac{1}{K} \right)^+ \leq \gamma$$

where the operator $(\cdot)^+$ takes only the non-negative part.

For a fixed dataset point $p \in P$, we refer to γ as the "slack" in the shattering. The slack corresponds to the total measure which is leftover after we remove an arbitrary subset of $A_{t,j} \cap N(p)$ of measure at least $\frac{1}{K}$.

Lemma 6.8 (Shattering [PTW10]). *Let $A_1, \dots, A_k$ collection of disjoint subsets of measure at most $\frac{1}{m}$. Then*

$$\Pr_{p \sim \mu}[p \textit{ is } (K, \gamma)\textit{-weakly shattered}] \geq 1 - \gamma$$

for $K = \Phi_r \left(\frac{1}{m}, \frac{\gamma^2}{4} \right) \cdot \frac{\gamma^3}{16}$.

For the remainder of the section, we let

$$K = \Phi_r \left(\frac{1}{m}, \frac{\gamma^2}{4} \right) \cdot \frac{\gamma^3}{16}.$$

We are interested in the shattering of dataset points with respect to the collections $\mathcal{A}_1$ and $\mathcal{A}_2$. The dataset points which get shattered will probe many cells in the data structure. Intuitively, a bit x_i corresponding to a dataset point p_i which is weakly-shattered should be stored across various cells.

So for each point p which is (K, γ) weakly-shattered we define subsets $\beta_1, \beta_2 \subset N(p)$ which hold the "slack" of the shattering of p with respect to $\mathcal{A}_1$ and $\mathcal{A}_2$.

Definition 6.9. *Let $p \in V$ be a dataset point which is (K, γ)-weakly shattered by $\mathcal{A}_1$ and $\mathcal{A}_2$. Let $\beta_1, \beta_2 \subset N(p)$ be arbitrary subsets where each $j \in [m]$ satisfies*

$$\mu(A_{1,j} \cap N(p) \setminus \beta_1) \leq \frac{1}{K}$$

and

$$\mu(A_{2,j} \cap N(p) \setminus \beta_2) \leq \frac{1}{K}$$

Since p is (K, γ)-weakly shattered, we can pick β_1 and β_2 with measure at most γ each. We will refer to $\beta(p) = \beta_1 \cup \beta_2$.

For a given collection $\mathcal{A}$, let $S(\mathcal{A}, p)$ be the event that the collection $\mathcal{A}$ (K, γ)-weakly shatters p. Note that Lemma 6.8 implies that $\Pr_{p \sim \mu}[S(\mathcal{A}, p)] \geq 1 - \gamma$.

Lemma 6.10. *With high probability over the choice of n point dataset, at most* $4\gamma n$ *points do not satisfy* $S(\mathcal{A}_1, p)$ *and* $S(\mathcal{A}_2, p)$.

Proof. This is a simple Chernoff bound. The expected number of points p which do not satisfy $S(\mathcal{A}_1, p)$ and $S(\mathcal{A}_2, p)$ is at most $2\gamma n$. Therefore, the probability that more than $4\gamma n$ points do not satisfy $S(\mathcal{A}_1, p)$ and $S(\mathcal{A}_2, p)$ is at most $\exp\left(-\frac{2\gamma n}{3}\right)$. □

We call a dataset *good* if there are at most $4\gamma n$ dataset points which are not (K, γ)-weakly shattered by $\mathcal{A}_1$ and $\mathcal{A}_2$.

Lemma 6.11. *There exists a good dataset* $P = \{p_i\}_{i=1}^n$ *where*

$$\Pr_{x \sim \mathcal{X}, q \sim \mathcal{Q}(P)}[A^D(q) = x_i] \geq \frac{2}{3} - o(1)$$

Proof. This follows via a simple argument. For any fixed dataset $P = \{p_i\}_{i=1}^n$, let

$$\mathbf{P} = \Pr_{x \sim \mathcal{X}, q \sim Q(p)}[A^D(q) = x_i]$$

to simplify notation.

$$\frac{2}{3} \leq \mathop{\mathbb{E}}_{P \sim \mathcal{P}}[\mathbf{P}] \tag{49}$$

$$= (1 - o(1)) \cdot \mathop{\mathbb{E}}_{P \sim \mathcal{P}}[\mathbf{P} \mid \ P \text{ is good}] + o(1) \cdot \mathop{\mathbb{E}}_{P \sim \mathcal{P}}[\mathbf{P} \mid \ P \text{ is not good}] \tag{50}$$

$$\frac{2}{3} - o(1) \leq (1 - o(1)) \cdot \mathop{\mathbb{E}}_{P \sim \mathcal{P}}[\mathbf{P} \mid P \text{ is good}] \tag{51}$$

Therefore, there exists a dataset which is not shattered by at most $4\gamma n$ and $\Pr_{x \sim \mathcal{X}, q \sim \mathcal{Q}(P)}[A^D(y) = x_i] \geq \frac{2}{3} - o(1)$. □

6.4 Corrupting some cell contents of shattered points

In the rest of the proof, we fix the dataset $P = \{p_i\}_{i=1}^n$ satisfying the conditions of Lemma 6.11, i.e., such that

$$\Pr_{x \sim \mathcal{X}, q \sim \mathcal{Q}(P)}[A^D(q) = x_i] \geq \frac{2}{3} - o(1).$$

We now introduce the notion of corruption of the data structure cells D, which parallels the notion of noise in locally-decodable codes. Remember that, after fixing some bit-string x, the algorithm A produces some data structure $D : [m] \to \{0, 1\}^w$.

Definition 6.12. *We call* $D' : [m] \to \{0, 1\}^w$ *a* corrupted *version of* D *at* k *cells if they differ on at most* k *cells, i.e., if* $|\{i \in [m] : D(i) \neq D'(i)\}| \leq k$.

In this section, we will show there exist a dataset P of n points and a set $S \subset [n]$ of size $\Omega(n)$ with good recovery probability, even if the algorithm has access to a corrupted version of data structure.

Definition 6.13. *For a fixed $x \in \{0,1\}^n$, let*

$$c_x(i) = \Pr_{q \sim N(p_i)}[A^D(q) = x_i].$$

Note that from the definitions of $\mathcal{Q}(P)$, $\mathbb{E}_{x \sim \mathcal{X}, i \in [n]}[c_x(i)] \geq \frac{2}{3} - o(1)$.

Lemma 6.14. *Fix $\varepsilon > 0$, vector $x \in \{0,1\}^n$, and let $D : [m] \to \{0,1\}^w$ be the data structure the algorithm produces on dataset P with bit-string x. Let D' be a corruption of D at εK cells. For every $i \in [n]$ where events $S(\mathcal{A}_1, p_i)$ and $S(\mathcal{A}_2, p_i)$ occur, we have*

$$\Pr_{q \sim N(p_i)}[A^{D'}(q) = x_i] \geq c_x(i) - 2\gamma - 2\varepsilon.$$

Proof. Note that $c_x(i)$ represents the probability mass of queries in the neighborhood of p_i for which the algorithm returns x_i. We want to understand how much of that probability mass we remove when we avoid probing the corrupted cells.

Since the dataset point p_i is (K, γ)-weakly shattered by $\mathcal{A}_1$ and $\mathcal{A}_2$, at most 2γ probability mass of $c_i(x)$ will come from the slack of the shattering. In more detail, if $q \sim N(p_i)$, we have probability $c_i(x)$ that the algorithm returns x_i. If we query $q \sim N(p_i) \setminus \beta(p_i)$, in the worst case, every query $q \in \beta(p_i)$ returns x_i; thus, after removing $\beta(p_i)$, we have removed at most 2γ probability mass over queries that the algorithm returns correctly.

The remaining probability mass is distributed across various cells, where each cell has at most $\frac{1}{K}$ mass for being probing in the first probe, and at most $\frac{1}{K}$ mass for being probe in the second probe. Therefore, if we remove εK cells, the first or second probe will probe those cells with probability at most 2ε. If we avoid the εK corrupted cells, the algorithm has the same output as it did with the uncorrupted data structure D. Therefore, the probability mass which returns x_i on query q in the corrupted data structure D' is at least $c_x(i) - 2\gamma - 2\varepsilon$. □

Lemma 6.15. *Fix $\gamma > 0$ to be a small enough constant. There exists a set $S \subset [n]$ of size $|S| = \Omega(n)$, such that whenever $i \in S$, we have that: events $S(\mathcal{A}_1, p_i)$ and $S(\mathcal{A}_2, p_i)$ occur, and*

$$\mathbb{E}_{x \sim \mathcal{X}}[c_x(i)] \geq \frac{1}{2} + \nu,$$

where ν can be taken to be some small constant like $\frac{1}{10}$.

Proof. There is at most a 4γ-fraction of the dataset points which are not shattered. For simplifying the notation, let $\mathbf{P} = \Pr_{i \in [n]}[\mathbb{E}_{x \sim \mathcal{X}}[c_x(i)] \geq \frac{1}{2} + \nu, S(\mathcal{A}_1, p_i) \wedge S(\mathcal{A}_2, p_i)]$. We need to show that

$\mathbf{P} = \Omega(1)$, since we will set $S \subset [n]$ as

$$S = \left\{ i \in [n] \mid \mathop{\mathbb{E}}_{x \sim \mathcal{X}}[c_x(i)] \geq \frac{1}{2} + \nu, S(\mathcal{A}_1, p_i) \wedge S(\mathcal{A}_2, p_i) \right\}.$$

The argument is a straight-forward averaging argument.

$$\begin{aligned} \frac{2}{3} - o(1) &\leq \mathop{\mathbb{E}}_{x \sim \mathcal{X}, i \in [n]}[c_x(i)] && (52) \\ &\leq 1 \cdot 4\gamma + 1 \cdot \mathbf{P} + \left(\frac{1}{2} + \nu\right) \cdot (1 - \mathbf{P}) && (53) \\ \frac{1}{6} - o(1) - 4\gamma - \nu &\leq \mathbf{P} \cdot \left(\frac{1}{2} - \nu\right). && (54) \end{aligned}$$

□

We combine Lemma 6.14 and Lemma 6.15 to obtain the following condition on the dataset.

Lemma 6.16. *Fix small enough $\gamma > 0$ and $\varepsilon > 0$. There exists a set $S \subset [n]$ where $|S| = \Omega(n)$, such that whenever $i \in S$,*

$$\mathop{\mathbb{E}}_{x \sim \mathcal{X}}\left[\Pr_{q \sim N(p_i)}[A^{D'}(q) = x_i]\right] \geq \frac{1}{2} + \eta$$

where $\eta = \nu - 2\gamma - 2\varepsilon$ and the algorithm probes a corrupted version of the data structure D.

Proof. Consider the set $S \subset [n]$ satisfying the conditions of Lemma 6.15. Whenever $i \in S$, p_i gets (K, γ)-weakly shattered and on average over x, A will recover x_i with probability $\frac{1}{2} + \nu$ when probing the data structure D on input $q \sim N(p_i)$, i.e

$$\mathop{\mathbb{E}}_{x \sim \mathcal{X}}\left[\Pr_{q \sim N(p_i)}[A^{D}(q) = x_i]\right] \geq \frac{1}{2} + \nu.$$

Therefore, from Lemma 6.14, if A probes D' which is a corruption of D in any εK cells, A will recover x_i with probability at least $\frac{1}{2} + \nu - 2\gamma - 2\varepsilon$ averaged over all $x \sim \mathcal{X}$ where $q \sim N(p_i)$. In other words,

$$\mathop{\mathbb{E}}_{x \sim \mathcal{X}}\left[\Pr_{q \sim N(p_i)}[A^{D'}(q) = x_i]\right] \geq \frac{1}{2} + \nu - 2\gamma - 2\varepsilon.$$

□

Theorem 6.17. *There exists an algorithm A and a subset $S \subseteq [n]$ of size $S = \Omega(n)$, where A makes only 2 cell probes to D. Furthermore, for any corruption of D at εK cells, A can recover x_i with probability at least $\frac{1}{2} + \eta$ over the random choice of $x \sim \mathcal{X}$.*

Proof. In order to extract x_i, we generate a random query $q \sim N(p_i)$ and we probe the data structure at the cells assuming the data structure is uncorrupted. From Lemma 6.16, there exists a set $S \subset [n]$ of size $\Omega(n)$ for which this algorithm recovers x_i with probability at least $\frac{1}{2} + \eta$, where the probability is taken on average over all possible $x \in \{0,1\}^n$. □

We fix the algorithm A and subset $S \subset [n]$ satisfying the conditions of Theorem 6.17. Since we fixed the dataset $P = \{p_i\}_{i=1}^n$ satisfying the conditions of Lemma 6.11, we say that $x \in \{0,1\}^n$ is an input to algorithm A in order to initialize the data structure with dataset $P = \{p_i\}_{i=1}^n$ and x_i is the bit associated with p_i.

6.5 Decreasing the word size

We now reduce to the case when the word size is $w = 1$ bit.

Lemma 6.18. *There exists a deterministic non-adaptive algorithm A' which on input $x \in \{0,1\}^n$ builds a data structure D' using $m2^w$ cells of width 1 bit. Any $i \in S$ as well as any corruption C to D' in at most εK positions satisfies*

$$\mathop{\mathbb{E}}_{x \in \{0,1\}^n}\left[\Pr_{q \sim N(p_i)}[A'^{C}(q) = x_i]\right] \geq \frac{1}{2} + \frac{\eta}{2^{2w}}$$

Proof. Given algorithm A which constructs the data structure $D : [m] \to \{0,1\}^w$ on input $x \in \{0,1\}^n$, construct the following data structure $D' : [m \cdot 2^w] \to \{0,1\}$. For each cell $D_j \in \{0,1\}^w$, make 2^w cells which contain all the parities of the w bits in D_j. This blows up the size of the data structure by 2^w.

Fix $i \in S$ and $q \in N(p_i)$ if algorithm A produces a function $f_q : \{0,1\}^w \times \{0,1\}^w \to \{0,1\}$ which succeeds with probability at least $\frac{1}{2} + \zeta$ over $x \in \{0,1\}^n$, then there exists a signed parity on some input bits which equals f_q in at least $\frac{1}{2} + \frac{\zeta}{2^{2w}}$ inputs $x \in \{0,1\}^n$. Let S_j be the parity of the bits of cell j and S_k be the parity of the bits of cell k. Let $f'_q : \{0,1\} \times \{0,1\} \to \{0,1\}$ denote the parity or the negation of the parity which equals f_q on $\frac{1}{2} + \frac{\zeta}{2^{2w}}$ possible input strings $x \in \{0,1\}^n$.

Algorithm A' will evaluate $f_{q'}$ at the cell containing the parity of the S_j bits in cell j and the parity of S_k bits in cell k. Let $I_{S_j}, I_{S_k} \in [m \cdot 2^w]$ be the indices of these cells. Since we can find such function for each fixed $q \in N(p_i)$, any two cell probes to $j, k \in [m]$, and any corrupted version of D, the algorithm A' satisfies

$$\mathop{\mathbb{E}}_{x \in \{0,1\}^n}\left[\Pr_{q \sim N(p_i)}[f'_q(C'_{I_{S_j}}, C'_{I_{S_k}}) = x_i]\right] \geq \frac{1}{2} + \frac{\eta}{2^{2w}}$$

whenever $i \in S$. □

For the remainder of the section, we will prove a version of Theorem 6.1 for algorithms with 1-bit words. Given Lemma 6.18, we will modify the space to $m \cdot 2^w$ and the probability to $\frac{1}{2} + \frac{\eta}{2^{2w}}$ to obtain the answer. So for the remainder of the section, assume algorithm A has 1 bit words.

6.6 Connecting to Locally-Decodable Codes

To complete the proof of Theorem 6.1, it remains to prove the following lemma.

Lemma 6.19. *Let A be a non-adaptive deterministic algorithm which makes 2 cell probes to a data structure D of m cells of width 1 bit which can handle εK corruptions and recover x_i with probability $\frac{1}{2} + \eta$ on random input $x \in \{0,1\}^n$ whenever $i \in S$ for some fixed S of size $\Omega(n)$. Then the following must hold*

$$\frac{m \log m}{n} \geq \Omega\left(\varepsilon K \eta^2\right).$$

The proof of the lemma uses [KdW04] and relies heavily on notions from quantum computing, in particular quantum information theory as applied to LDC lower bounds.

6.6.1 Crash Course in Quantum Computing

We introduce a few concepts from quantum computing that are necessary in our subsequent arguments. A *qubit* is a unit-length vector in $\mathbb{C}^2$. We write a qubit as a linear combination of the basis states $\binom{1}{0} = |0\rangle$ and $\binom{0}{1} = |1\rangle$. The qubit $\alpha = \binom{\alpha_1}{\alpha_2}$ can be written

$$|\alpha\rangle = \alpha_1 |0\rangle + \alpha_2 |1\rangle$$

where we refer to α_1 and α_2 as *amplitudes* and $|\alpha_1|^2 + |\alpha_2|^2 = 1$. An *$m$-qubit system* is a vector in the tensor product $\mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2$ of dimension 2^m. The basis states correspond to all 2^m bit-strings of length m. For $j \in [2^m]$, we write $|j\rangle$ as the basis state $|j_1\rangle \otimes |j_2\rangle \otimes \cdots \otimes |j_m\rangle$ where $j = j_1 j_2 \ldots j_m$ is the binary representation of j. We will write the m-qubit *quantum state* $|\phi\rangle$ as unit-vector given by linear combination over all 2^m basis states. So $|\phi\rangle = \sum_{j \in [2^m]} \phi_j |j\rangle$. As a shorthand, $\langle\phi|$ corresponds to the conjugate transpose of a quantum state.

A *mixed state* $\{p_i, |\phi_i\rangle\}$ is a probability distribution over quantum states. In this case, we the quantum system is in state $|\phi_i\rangle$ with probability p_i. We represent mixed states by a density matrix $\sum p_i |\phi_i\rangle \langle\phi_i|$.

A measurement is given by a family of positive semi-definite operators which sum to the identity operator. Given a quantum state $|\phi\rangle$ and a measurement corresponding to the family of operators $\{M_i^* M_i\}_i$, the measurement yields outcome i with probability $\|M_i |\phi\rangle\|^2$ and results in state $\frac{M_i|\phi\rangle}{\|M_i|\phi\rangle\|^2}$, where the norm $\|\cdot\|$ is the ℓ_2 norm. We say the measurement makes the *observation* M_i.

Finally, a quantum algorithm makes a query to some bit-string $y \in \{0,1\}^m$ by starting with the state $|c\rangle |j\rangle$ and returning $(-1)^{c \cdot y_j} |c\rangle |j\rangle$. One can think of c as the control qubit taking values 0 or 1; if $c = 0$, the state remains unchanged by the query, and if $c = 1$ the state receives a $(-1)^{y_j}$ in its amplitude. The queries may be made in superposition to a state, so the state $\sum_{c \in \{0,1\}, j \in [m]} \alpha_{cj} |c\rangle |j\rangle$ becomes $\sum_{c \in \{0,1\}, j \in [m]} (-1)^{c \cdot y_j} \alpha_{cj} |c\rangle |j\rangle$.

6.6.2 Weak quantum random access codes from GNS algorithms

Definition 6.20. *$C : \{0,1\}^n \to \{0,1\}^m$ is a $(2, \delta, \eta)$-LDC if there exists a randomized decoding algorithm making at most 2 queries to an m-bit string y non-adaptively, and for all $x \in \{0,1\}^n$, $i \in [n]$, and $y \in \{0,1\}^m$ where $d(y, C(x)) \leq \delta m$, the algorithm can recover x_i from the two queries to y with probability at least $\frac{1}{2} + \eta$.*

In their paper, [KdW04] prove the following result about 2-query LDCs.

Theorem 6.21 (Theorem 4 in [KdW04])**.** *If $C : \{0,1\}^n \to \{0,1\}^m$ is a $(2, \delta, \eta)$-LDC, then $m \geq 2^{\Omega(\delta\eta^2 n)}$.*

The proof of Theorem 6.21 proceeds as follows. They show how to construct a 1-query quantum-LDC from a classical 2-query LDC. From a 1-query quantum-LDC, [KdW04] constructs a quantum random access code which encodes n-bit strings in $O(\log m)$ qubits. Then they apply a quantum information theory lower bound due to Nayak [Nay99]:

Theorem 6.22 (Theorem 2 stated in [KdW04] from Nayak [Nay99])**.** *For any encoding $x \to \rho_x$ of n-bit strings into m-qubit states, such that a quantum algorithm, given query access to ρ_x, can decode any fixed x_i with probability at least $1/2 + \eta$, it must hold that $m \geq (1 - H(1/2 + \eta))n$.*

Our proof will follow a pattern similar to the proof of Theorem 6.21. We assume the existence of a GNS algorithm A which builds a data structure $D : [m] \to \{0,1\}$. We can think of D as a length m binary string encoding x; in particular let $D_j \in \{0,1\}$ be the jth bit of D.

Our algorithm A from Theorem 6.17 does not satisfy the strong properties of an LDC, preventing us from applying 6.21 directly. However, it does have some LDC-*ish* guarantees. In particular, we can support εK corruptions to D. In the LDC language, this means that we can tolerate a noise rate of $\delta = \frac{\varepsilon K}{m}$. Additionally, we cannot necessarily recover *every* coordinate x_i, but we can recover x_i for $i \in S$, where $|S| = \Omega(n)$. Also, our success probability is $\frac{1}{2} + \eta$ over the random choice of $i \in S$ and the random choice of the bit-string $x \in \{0,1\}^n$. Our proof follows by adapting the arguments of [KdW04] to this weaker setting.

Lemma 6.23. *Let $r = \frac{2}{\delta a^2}$ where $\delta = \dfrac{\varepsilon K}{m}$ and $a \leq 1$ is a constant. Let D be the data structure from above (i.e., satisfying the hypothesis of Lemma 6.19). Then there exists a quantum algorithm that, starting from the $r(\log m + 1)$-qubit state with r copies of $|U(x)\rangle$, where*

$$|U(x)\rangle = \frac{1}{\sqrt{2m}} \sum_{c \in \{0,1\}, j \in [m]} (-1)^{c \cdot D_j} |c\rangle |j\rangle$$

can recover x_i for any $i \in S$ with probability $\frac{1}{2} + \Omega(\eta)$ (over a random choice of x).

Assuming Lemma 6.23, we can complete the proof of Lemma 6.19.

Proof of Lemma 6.19. The proof is similar to the proof of Theorem 2 of [KdW04]. Let ρ_x represent the s-qubit system consisting of the r copies of the state $|U(x)\rangle$, where $s = r(\log m + 1)$; ρ_x is an encoding of x. Using Lemma 6.23, we can assume we have a quantum algorithm that, given ρ_x, can recover x_i for any $i \in S$ with probability $\alpha = \frac{1}{2} + \Omega(\eta)$ over the random choice of $x \in \{0,1\}^n$.

We will let $H(A)$ be the Von Neumann entropy of A, and $H(A|B)$ be the conditional entropy and $H(A:B)$ the mutual information.

Let XM be the $(n+s)$-qubit system

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} |x\rangle \langle x| \otimes \rho_x.$$

The system corresponds to the uniform superposition of all 2^n strings concatenated with their encoding ρ_x. Let X be the first subsystem corresponding to the first n qubits and M be the second subsystem corresponding to the s qubits. We have

$$H(XM) = n + \frac{1}{2^n} \sum_{x \in \{0,1\}^n} H(\rho_x) \geq n = H(X) \tag{55}$$

$$H(M) \leq s, \tag{56}$$

since M has s qubits. Therefore, the mutual information $H(X:M) = H(X) + H(M) - H(XM) \leq s$. Note that $H(X|M) \leq \sum_{i=1}^{n} H(X_i|M)$. By Fano's inequality, if $i \in S$,

$$H(X_i|M) \leq H(\alpha)$$

where we are using the fact that Fano's inequality works even if we can recover x_i with probability α averaged over all x's. Additionally, if $i \notin S$, $H(X_i|M) \leq 1$. Therefore,

$$\begin{aligned} s \geq H(X:M) &= H(X) - H(X|M) && (57)\\ &\geq H(X) - \sum_{i=1}^{n} H(X_i|M) && (58)\\ &\geq n - |S|H(\alpha) - (n - |S|) && (59)\\ &= |S|(1 - H(\alpha)). && (60) \end{aligned}$$

Furthermore, $1 - H(\alpha) \geq \Omega(\eta^2)$ since, and $|S| = \Omega(n)$, we have

$$\frac{2m}{a^2 \varepsilon K}(\log m + 1) \geq \Omega\left(n\eta^2\right) \tag{61}$$

$$\frac{m \log m}{n} \geq \Omega\left(\varepsilon K \eta^2\right). \tag{62}$$

□

It remains to prove Lemma 6.23, which we proceed to do in the rest of the section. We first show that we can simulate our GNS algorithm with a 1-query quantum algorithm.

Lemma 6.24. *Fix an $x \in \{0,1\}^n$ and $i \in [n]$. Let $D : [m] \to \{0,1\}$ be the data structure produced by algorithm A on input x. Suppose $\Pr_{q \sim N(p_i)}[A^D(q) = x_i] = \frac{1}{2} + b$ for $b > 0$. Then there exists a quantum algorithm which makes one quantum query (to D) and succeeds with probability $\frac{1}{2} + \frac{4b}{7}$ to output x_i.*

Proof. We use the procedure in Lemma 1 of [KdW04] to determine the output algorithm A on input x at index i. The procedure simulates two classical queries with one quantum query. □

All quantum algorithms which make 1-query to D can be specified in the following manner: there is a quantum state $|Q_i\rangle$, where

$$|Q_i\rangle = \sum_{c \in \{0,1\}, j \in [m]} \alpha_{cj} |c\rangle |j\rangle$$

which queries D. After querying D, the resulting quantum state is $|Q_i(x)\rangle$, where

$$|Q_i(x)\rangle = \sum_{c \in \{0,1\}, j \in [m]} (-1)^{c \cdot D_j} \alpha_{cj} |c\rangle |j\rangle .$$

There is also a quantum measurement $\{R, I - R\}$ such that, after the algorithm obtains the state $|Q_i(x)\rangle$, it performs the measurement $\{R, I - R\}$. If the algorithm observes R, it outputs 1 and if the algorithm observes $I - R$, it outputs 0.

From Lemma 6.24, we know there must exist a state $|Q_i\rangle$ and $\{R, I - R\}$ where if algorithm A succeeds with probability $\frac{1}{2} + \eta$ on random $x \sim \{0,1\}^n$, then the quantum algorithm succeeds with probability $\frac{1}{2} + \frac{4\eta}{7}$ on random $x \sim \{0,1\}^n$.

In order to simplify notation, we write $p(\phi)$ as the probability of making observation R from state $|\phi\rangle$. Since R is a positive semi-definite matrix, $R = M^* M$ and so $p(\phi) = \| M |\phi\rangle \|^2$.

In exactly the same way as [KdW04], we can remove parts of the quantum state $|Q_i(x)\rangle$ where $\alpha_{cj} > \frac{1}{\sqrt{\delta m}} = \frac{1}{\sqrt{\varepsilon K}}$. If we let $L = \{(c,j) \mid \alpha_{cj} \leq \frac{1}{\sqrt{\varepsilon K}}\}$, after keeping only the amplitudes in L, we obtain the quantum state $\frac{1}{a} |A_i(x)\rangle$, where

$$|A_i(x)\rangle = \sum_{(c,j) \in L} (-1)^{c \cdot D_j} \alpha_{cj} |c\rangle |j\rangle \qquad a = \sqrt{\sum_{(c,j) \in L} \alpha_{cj}^2}$$

Lemma 6.25. *Fix $i \in S$. The quantum state $|A_i(x)\rangle$ satisfies*

$$\mathop{\mathbb{E}}_{x \in \{0,1\}^n} \left[p\left(\frac{1}{a} A_i(x) \right) \mid x_i = 1 \right] - \mathop{\mathbb{E}}_{x \in \{0,1\}^n} \left[p\left(\frac{1}{a} A_i(x) \right) \mid x_i = 0 \right] \geq \frac{8\eta}{7a^2}.$$

Proof. Note that since $|Q_i(x)\rangle$ and $\{R, I-R\}$ simulate A and succeed with probability at least $\frac{1}{2}+\frac{4\eta}{7}$ on a random $x \in \{0,1\}^n$, we have that

$$\frac{1}{2} \underset{x\in\{0,1\}^n}{\mathbb{E}} \left[p\left(Q_i(x)\right) \mid x_i = 1\right] + \frac{1}{2} \underset{x\in\{0,1\}^n}{\mathbb{E}} \left[1 - p\left(Q_i(x)\right) \mid x_i = 0\right] \geq \frac{1}{2} + \frac{4\eta}{7}, \tag{63}$$

which we can simplify to say

$$\underset{x\in\{0,1\}^n}{\mathbb{E}} \left[p\left(Q_i(x)\right) \mid x_i = 1\right] + \underset{x\in\{0,1\}^n}{\mathbb{E}} \left[p\left(Q_i(x)\right) \mid x_i = 0\right] \geq \frac{8\eta}{7}. \tag{64}$$

Since $|Q_i(x)\rangle = |A_i(x)\rangle + |B_i(x)\rangle$ and $|B_i(x)\rangle$ contains at most εK parts, if all probes to D in $|B_i(x)\rangle$ had corrupted values, the algorithm should still succeed with the same probability on random inputs x. Therefore, the following two inequalities hold:

$$\underset{x\in\{0,1\}^n}{\mathbb{E}} \left[p\left(A_i(x) + B(x)\right) \mid x_i = 1\right] + \underset{x\in\{0,1\}^n}{\mathbb{E}} \left[p\left(A_i(x) + B(x)\right) \mid x_i = 0\right] \geq \frac{8\eta}{7} \tag{65}$$

$$\underset{x\in\{0,1\}^n}{\mathbb{E}} \left[p\left(A_i(x) - B(x)\right) \mid x_i = 1\right] + \underset{x\in\{0,1\}^n}{\mathbb{E}} \left[p\left(A_i(x) - B(x)\right) \mid x_i = 0\right] \geq \frac{8\eta}{7} \tag{66}$$

Note that $p(\phi \pm \psi) = p(\phi) + p(\psi) \pm (\langle\phi| R |\psi\rangle + \langle\psi| D |\phi\rangle)$ and $p(\frac{1}{c}\phi) = \frac{p(\phi)}{c^2}$. One can verify by averaging the two inequalities (65) and (66) that we get the desired expression. □

Lemma 6.26. *Fix $i \in S$. There exists a quantum algorithm that starting from the quantum state $\frac{1}{a}|A_i(x)\rangle$, can recover the value of x_i with probability $\frac{1}{2}+\frac{2\eta}{7a^2}$ over random $x \in \{0,1\}^n$.*

Proof. The algorithm and argument are almost identical to Theorem 3 in [KdW04], we just check that it works under the weaker assumptions. Let

$$q_1 = \underset{x\in\{0,1\}^n}{\mathbb{E}} \left[p\left(\frac{1}{a}A_i(x)\right) \mid x_i = 1\right] \qquad q_0 = \underset{x\in\{0,1\}^n}{\mathbb{E}} \left[p\left(\frac{1}{a}A_i(x)\right) \mid x_i = 0\right].$$

From Lemma 6.25, we know $q_1 - q_0 \geq \frac{8\eta}{7a^2}$. In order to simplify notation, let $b = \frac{4\eta}{7a^2}$. So we want a quantum algorithm which starting from state $\frac{1}{a}|A_i(x)\rangle$ can recover x_i with probability $\frac{1}{2}+\frac{b}{2}$ on random $x \in \{0,1\}^n$. Assume $q_1 \geq \frac{1}{2}+b$, since otherwise $q_0 \leq \frac{1}{2}-b$ and the same argument will work for 0 and 1 flipped. Also, assume $q_1 + q_0 \geq 1$, since otherwise simply outputting 1 on observation R and 0 on observation $I-R$ will work.

The algorithm works in the following way: it outputs 0 with probability $1-\frac{1}{q_1+q_0}$ and otherwise makes the measurement $\{R, I-R\}$ on state $\frac{1}{a}|A_i(x)\rangle$. If the observation made is R, then the algorithm outputs 1, otherwise, it outputs 0. The probability of success over random input $x \in$

$\{0,1\}^n$ is

$$\underset{x\in\{0,1\}^n}{\mathbb{E}}[\Pr[\text{returns correctly}]] \\ = \frac{1}{2}\underset{x\in\{0,1\}^n}{\mathbb{E}}[\Pr[\text{returns } 1] \mid x_i = 1] + \frac{1}{2}\underset{x\in\{0,1\}^n}{\mathbb{E}}[\Pr[\text{returns } 0] \mid x_i = 0]\,. \tag{67}$$

When $x_i = 1$, the probability the algorithm returns correctly is $(1-q)p\left(\frac{1}{a}A_i(x)\right)$ and when $x_i = 0$, the probability the algorithm returns correctly is $q + (1-q)(1 - p(\frac{1}{a}A_i(x)))$. So simplifying (67),

$$\underset{x\in\{0,1\}^n}{\mathbb{E}}[\Pr[\text{returns correctly}]] = \frac{1}{2}(1-q)q_1 + \frac{1}{2}(q + (1-q)(1-q_0)) \tag{68}$$

$$\geq \frac{1}{2} + \frac{b}{2}. \tag{69}$$

□

Now we can finally complete the proof of Lemma 6.23.

Proof of Lemma 6.23. Again, the proof is exactly the same as the finishing arguments of Theorem 3 in [KdW04], and we simply check the weaker conditions give the desired outcome. On input $i \in [n]$ and access to r copies of the state $|U(x)\rangle$, the algorithm applies the measurement $\{M_i^* M_i, I - M_i^* M_i\}$ where

$$M_i = \sqrt{\varepsilon K} \sum_{(c,j)\in L} \alpha_{cj} |c,j\rangle \langle c,j|\,.$$

This measurement is designed in order to yield the state $\frac{1}{a}|A_i(x)\rangle$ on $|U(x)\rangle$ if the measurement makes the observation $M_i^* M_i$. The fact that the amplitudes of $|A_i(x)\rangle$ are not too large makes $\{M_i^* M_i, I - M_i^* M_i\}$ a valid measurement.

The probability of observing $M_i^* M_i$ is $\langle U(x)| M_i^* M_i |U(x)\rangle = \frac{\delta a^2}{2}$, where we used that $\delta = \frac{\varepsilon K}{m}$. So the algorithm repeatedly applies the measurement until observing outcome $M_i^* M_i$. If it never makes the observation, the algorithm outputs 0 or 1 uniformly at random. If the algorithm does observe $M_i^* M_i$, it runs the output of the algorithm of Lemma 6.26. The following simple calculation (done in [KdW04]) gives the desired probability of success on random input,

$$\underset{x\in\{0,1\}^n}{\mathbb{E}}[\Pr[\text{returns correctly}]] \geq \left(1 - (1 - \delta a^2/2)^r\right)\left(\frac{1}{2} + \frac{2\eta}{7a^2}\right) + (1 - \delta a^2/2)^r \cdot \frac{1}{2} \tag{70}$$

$$\geq \frac{1}{2} + \frac{\eta}{7a^2}. \tag{71}$$

□

6.6.3 On adaptivity

We can extend our lower bounds from the non-adaptive to the adaptive setting.

Lemma 6.27. *If there exists a deterministic data structure which makes two queries adaptively and succeeds with probability at least* $\frac{1}{2}+\eta$*, there exists a deterministic data structure which makes the two queries non-adaptively and succeeds with probability at least* $\frac{1}{2}+\frac{\eta}{2^w}$*.*

Proof. The algorithm guesses the outcome of the first cell probe and simulates the adaptive algorithm with the guess. After knowing which two probes to make, we probe the data structure non-adaptively. If the algorithm guessed the contents of the first cell-probe correctly, then we output the value of the non-adaptive algorithm. Otherwise, we output a random value. This algorithm is non-adaptive and succeeds with probability at least $\left(1-\frac{1}{2^w}\right)\cdot\frac{1}{2}+\frac{1}{2^w}\left(\frac{1}{2}+\eta\right)=\frac{1}{2}+\frac{\eta}{2^w}$. □

Applying this theorem, from an adaptive algorithm succeeding with probability $\frac{2}{3}$, we obtain a non-adaptive algorithm which succeeds with probability $\frac{1}{2}+\Omega(2^{-w})$. This value is lower than the intended $\frac{2}{3}$, but we the reduction to a weak LDC still goes through when let $\gamma=\Theta(2^{-w})$, $\varepsilon=\Theta(2^{-w})$. Another consequence is that $|S|=\Omega(2^{-w}n)$.

One can easily verify that for small enough $\gamma=\Omega(2^{-w})$,

$$\frac{m\log m\cdot 2^{\Theta(w)}}{n}\geq\Omega\left(\Phi_r\left(\frac{1}{m},\gamma\right)\right)$$

Which yields tight lower bounds (up to sub-polynomial factors) for the Hamming space when $w=o(\log n)$.

In the case of the Hamming space, we can compute robust expansion in a similar fashion to Theorem 1.2. In particular, for any $p,q\in[1,\infty)$ where $(p-1)(q-1)=\sigma^2$, we have

$$\frac{m\log m\cdot 2^{O(w)}}{n}\geq\Omega(\gamma^q m^{1+q/p-q}) \tag{72}$$

$$m^{q-q/p+o(1)}\geq n^{1-o(1)}\gamma^q \tag{73}$$

$$m\geq n^{\frac{1-o(1)}{q-q/p+o(1)}}\gamma^{\frac{q}{q-q/p+o(1)}} \tag{74}$$

$$=n^{\frac{p}{pq-q}-o(1)}\gamma^{\frac{p}{p-1}-o(1)} \tag{75}$$

Let $p=1+\frac{wf(n)}{\log n}$ and $q=1+\sigma^2\frac{\log n}{wf(n)}$ where we require that $wf(n)=o(\log n)$ and $f(n)\to\infty$ as $n\to\infty$.

$$m\geq n^{\frac{1}{\sigma^2}-o(1)}2^{\frac{\log n}{\log\log n}} \tag{76}$$

$$\geq n^{\frac{1}{\sigma^2}-o(1)} \tag{77}$$

7 Acknowledgments

We would like to thank Jop Briët for helping us to navigate literature about LDCs. We thank Omri Weinstein for useful discussions.

References

[AC09] Nir Ailon and Bernard Chazelle. The fast Johnson–Lindenstrauss transform and approximate nearest neighbors. *SIAM J. Comput.*, 39(1):302–322, 2009.

[ACP08] Alexandr Andoni, Dorian Croitoru, and Mihai Pătraşcu. Hardness of nearest neighbor under L-infinity. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 424–433, 2008.

[ADI$^+$06] Alexandr Andoni, Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. *Nearest Neighbor Methods for Learning and Vision: Theory and Practice, Neural Processing Information Series, MIT Press*, 2006.

[AI06] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 459–468, 2006.

[AI08] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[AIL$^+$15] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal LSH for angular distance. In *NIPS*, 2015. Full version available at `http://arxiv.org/abs/1509.02897`.

[AINR14] Alexandr Andoni, Piotr Indyk, Huy L. Nguyen, and Ilya Razenshteyn. Beyond locality-sensitive hashing. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2014. Full version at `http://arxiv.org/abs/1306.1547`.

[AIP06] Alexandr Andoni, Piotr Indyk, and Mihai Pătraşcu. On the optimality of the dimensionality reduction method. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 449–458, 2006.

[And09] Alexandr Andoni. *Nearest Neighbor Search: the Old, the New, and the Impossible*. PhD thesis, MIT, 2009. Available at `http://www.mit.edu/~andoni/thesis/main.pdf`.

[AR15] Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the Symposium on Theory of Computing (STOC)*, 2015. Full version at `http://arxiv.org/abs/1501.01062`.

[AR16] Alexandr Andoni and Ilya Razenshteyn. Tight lower bounds for data-dependent locality-sensitive hashing. In *Proceedings of the ACM Symposium on Computational Geometry (SoCG)*, 2016. Available at `http://arxiv.org/abs/1507.04299`.

[AV15] Amirali Abdullah and Suresh Venkatasubramanian. A directed isoperimetric inequality with application to bregman near neighbor lower bounds. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 509–518, 2015.

[AW15] Josh Alman and Ryan Williams. Probabilistic polynomials and hamming nearest neighbors. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2015.

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2016.

[BOR99] Allan Borodin, Rafail Ostrovsky, and Yuval Rabani. Lower bounds for high dimensional nearest neighbor search and related problems. *Proceedings of the Symposium on Theory of Computing*, 1999.

[BR02] Omer Barkol and Yuval Rabani. Tighter bounds for nearest neighbor search and related problems in the cell probe model. *J. Comput. Syst. Sci.*, 64(4):873–896, 2002. Previously appeared in STOC'00.

[BRdW08] Avraham Ben-Aroya, Oded Regev, and Ronald de Wolf. A hypercontractive inequality for matrix-valued functions with applications to quantum computing and ldcs. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 477–486, 2008.

[CCGL99] Amit Chakrabarti, Bernard Chazelle, Benjamin Gum, and Alexey Lvov. A lower bound on the complexity of approximate nearest-neighbor searching on the Hamming cube. *Proceedings of the Symposium on Theory of Computing (STOC)*, 1999.

[Cha02] Moses Charikar. Similarity estimation techniques from rounding. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 380–388, 2002.

[Cla88] Ken Clarkson. A randomized algorithm for closest-point queries. *SIAM Journal on Computing*, 17:830–847, 1988.

[CR04] Amit Chakrabarti and Oded Regev. An optimal randomised cell probe lower bounds for approximate nearest neighbor searching. *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2004.

[DG03] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures Algorithms*, 22(1):60–65, 2003.

[DG15] Zeev Dvir and Sivakanth Gopi. 2-server PIR with sub-polynomial communication. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 577–584, 2015.

[DIIM04] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the ACM Symposium on Computational Geometry (SoCG)*, 2004.

[DRT11] Inderjit S. Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Nearest neighbor based greedy coordinate descent. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain.*, pages 2160–2168, 2011.

[GIM99] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB)*, 1999.

[HIM12] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of Computing*, 1(8):321–350, 2012.

[HLM15] Thomas Hofmann, Aurélien Lucchi, and Brian McWilliams. Neighborhood watch: Stochastic gradient descent with neighbors. *CoRR*, abs/1506.03662, 2015.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbor: towards removing the curse of dimensionality. *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 604–613, 1998.

[Ind01a] Piotr Indyk. *High-dimensional computational geometry.* Ph.D. Thesis. Department of Computer Science, Stanford University, 2001.

[Ind01b] Piotr Indyk. On approximate nearest neighbors in ℓ_∞ norm. *J. Comput. Syst. Sci.*, 63(4):627–638, 2001. Preliminary version appeared in FOCS'98.

[JKKR04] T. S. Jayram, Subhash Khot, Ravi Kumar, and Yuval Rabani. Cell-probe lower bounds for the partial match problem. *Journal of Computer and Systems Sciences*, 69(3):435–447, 2004. See also STOC'03.

[JL84] William B. Johnson and Joram Lindenstrauss. Extensions of lipshitz mapping into hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[Kap15] Michael Kapralov. Smooth tradeoffs between insert and query complexity in nearest neighbor search. In *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, pages 329–342, New York, NY, USA, 2015. ACM.

[KdW04] Iordanis Kerenidis and Ronald de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. *Journal of Computer and System Sciences*, 69(3):395–420, 2004.

[KKK16] Matti Karppa, Petteri Kaski, and Jukka Kohonen. A faster subquadratic algorithm for finding outlier correlations. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2016. Available at `http://arxiv.org/abs/1510.03895`.

[KKKÓ16] Matti Karppa, Petteri Kaski, Jukka Kohonen, and Padraig Ó Catháin. Explicit correlation amplifiers for finding outlier correlations in deterministic subquadratic time. In *Proceedings of the 24th European Symposium Of Algorithms (ESA '2016)*, 2016. To appear.

[KOR00] Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.*, 30(2):457–474, 2000. Preliminary version appeared in STOC'98.

[KP12] Michael Kapralov and Rina Panigrahy. NNS lower bounds via metric expansion for ℓ_∞ and EMD. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP)*, pages 545–556, 2012.

[Laa15a] Thijs Laarhoven. *Search problems in cryptography: From fingerprinting to lattice sieving.* PhD thesis, Eindhoven University of Technology, 2015.

[Laa15b] Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 3–22, 2015.

[Laa15c] Thijs Laarhoven. Tradeoffs for nearest neighbors on the sphere. *CoRR*, abs/1511.07527, 2015.

[Liu04] Ding Liu. A strong lower bound for approximate nearest neighbor searching in the cell probe model. *Information Processing Letters*, 92:23–29, 2004.

[LJW$^+$07] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-probe LSH: efficient indexing for high-dimensional similarity search. In *VLDB*, 2007.

[LPY16] Mingmou Liu, Xiaoyin Pan, and Yitong Yin. Randomized approximate nearest neighbor search with limited adaptivity. *CoRR*, abs/1602.04421, 2016.

[Mei93] Stefan Meiser. Point location in arrangements of hyperplanes. *Information and Computation*, 106:286–303, 1993.

[Mil99] Peter Bro Miltersen. Cell probe complexity-a survey. *Proceedings of the 19th Conference on the Foundations of Software Technology and Theoretical Computer Science, Advances in Data Structures Workshop*, page 2, 1999.

[MNP07] Rajeev Motwani, Assaf Naor, and Rina Panigrahy. Lower bounds on locality sensitive hashing. *SIAM Journal on Discrete Mathematics*, 21(4):930–935, 2007. Previously in SoCG'06.

[MNSW98] Peter B. Miltersen, Noam Nisan, Shmuel Safra, and Avi Wigderson. Data structures and asymmetric communication complexity. *Journal of Computer and System Sciences*, 1998.

[Nay99] Ashwin Nayak. Optimal lower bounds for quantum automata and random access codes. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 369–376. IEEE, 1999.

[Ngu14] Huy L. Nguyên. *Algorithms for High Dimensional Data.* PhD thesis, Princeton University, 2014. Available at `http://arks.princeton.edu/ark:/88435/dsp01b8515q61f`.

[O'D14] Ryan O'Donnell. *Analysis of boolean functions.* Cambridge University Press, 2014.

[OWZ14] Ryan O'Donnell, Yi Wu, and Yuan Zhou. Optimal lower bounds for locality sensitive hashing (except when q is tiny). *Transactions on Computation Theory*, 6(1):5, 2014. Previously in ICS'11.

[Pag16] Rasmus Pagh. Locality-sensitive hashing without false negatives. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2016. Available at `http://arxiv.org/abs/1507.03225`.

[Pan06] Rina Panigrahy. Entropy-based nearest neighbor algorithm in high dimensions. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006.

[Păt11] Mihai Pătraşcu. Unifying the landscape of cell-probe lower bounds. *SIAM Journal on Computing*, 40(3):827–847, 2011. See also FOCS'08, arXiv:1010.3783.

[PP16] Ninh Pham and Rasmus Pagh. Scalability and total recall with fast CoveringLSH. *CoRR*, abs/1602.02620, 2016.

[PT06] Mihai Pătraşcu and Mikkel Thorup. Higher lower bounds for near-neighbor and further rich problems. *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2006.

[PTW08] Rina Panigrahy, Kunal Talwar, and Udi Wieder. A geometric approach to lower bounds for approximate near-neighbor search and partial match. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 414–423, 2008.

[PTW10] Rina Panigrahy, Kunal Talwar, and Udi Wieder. Lower bounds on near neighbor search via metric expansion. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 805–814, 2010.

[Raz14] Ilya Razenshteyn. Beyond Locality-Sensitive Hashing. Master's thesis, MIT, 2014.

[SDI06] Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk, editors. *Nearest Neighbor Methods in Learning and Vision.* Neural Processing Information Series, MIT Press, 2006.

[TT07] Tengo Terasawa and Yuzuru Tanaka. Spherical LSH for approximate nearest neighbor search on unit hypersphere. *Workshop on Algorithms and Data Structures*, 2007.

[Val88] Leslie G Valiant. Functionality in neural nets. In *First Workshop on Computational Learning Theory*, pages 28–39, 1988.

[Val15] Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *J. ACM*, 62(2):13, 2015. Previously in FOCS'12.

[WLKC15] Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. Learning to hash for indexing big data — a survey. Available at `http://arxiv.org/abs/1509.05472`, 2015.

[WSSJ14] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. Hashing for similarity search: A survey. *CoRR*, abs/1408.2927, 2014.

[Yin16] Yitong Yin. Simple average-case lower bounds for approximate near-neighbor from isoperimetric inequalities. *CoRR*, abs/1602.05391, 2016.

[ZYS16] Zeyuan Allen Zhu, Yang Yuan, and Karthik Sridharan. Exploiting the structure: Stochastic gradient methods using raw clusters. *CoRR*, abs/1602.02151, 2016.

A Random instances for ℓ_2

We first introduce the equivalent notion of the "random instance" (from Section 1.3) for ℓ_2. This instance is what lies at the core of the optimal data-dependent LSH from [AR15].

- All points and queries lie on a unit sphere $S^{d-1} \subset \mathbb{R}^d$.
- The dataset P is generated by sampling n unit vectors in S^{d-1} independently and uniformly at random.
- A query q is generated by first choosing a dataset point $p \in P$ uniformly at random, and then choosing q uniformly at random from all points in S^{d-1} within distance $\frac{\sqrt{2}}{c}$ from p.
- The goal of the data structure is to preprocess P so given a query q generated as above, can recover the corresponding data point p.

This instance must be handled by any data structure for $\left(c + o(1), \frac{\sqrt{2}}{c}\right)$-ANN over ℓ_2. In fact, [AR15] show how to reduce any (c, r)-ANN instance into several (pseudo-)random instances from above without increasing the time and space complexity by a polynomial factor. The resulting instances are *pseudo*-random because they are not exactly the random instance described above, but do have roughly the same distribution over distances from q to the data points.

Following the strategy from [AR15], we first analyze the random instance, and then reduce the case for general subsets of $\mathbb{R}^d$ to pseudo-random instances.

B Spherical case

We describe how to solve a random instance of ANN on a unit sphere $S^{d-1} \subseteq \mathbb{R}^d$, where near neighbors are planted within distance $\frac{\sqrt{2}}{c}$ (as defined in Appendix A). We obtain the same time-space tradeoff as in [Laa15c], namely (1). In Appendix C, we extend this algorithm to the entire space $\mathbb{R}^d$ using the techniques from [AR15].

Below we assume that $d = \widetilde{O}(\log n)$ [JL84, DG03].

B.1 The data structure description

The data structure is a *single* rooted T-ary tree consisting of $K + 1$ levels. The zeroth level holds the root r, and each node up to the K-th level has T children, so there are T^K leaves. For every node v, let $\mathcal{P}_v$ be the set of nodes on the path from v to the root except the root itself. Each node v except the root holds a random Gaussian vector $z_v \sim N(0,1)^d$ is stored . For each node v, we define the subset of the dataset $P_v \subset P$:

$$P_v = \left\{ p \in P \mid \forall v' \in \mathcal{P}_v \;\; \langle z_{v'}, p \rangle \geq \eta \right\},$$

where $\eta > 0$ is a parameter to be chosen later. For instance, $P_r = P$, since $\mathcal{P}_r = \emptyset$. Intuitively, each set P_v corresponds to a subset of the dataset which lies in the intersection of sphere caps centered around $z_{v'}$ for all $v' \in \mathcal{P}_v$. Every *leaf* v of the tree stores the subset P_v.

To process the query $q \in S^{d-1}$, we start with the root and make our way down the tree. We consider all the children of the root v with $\langle z_v, q \rangle \geq \eta'$, where $\eta' > 0$ is a parameter to be chosen later, and recurse on them. If we end up in a leaf v, we try all the points from P_v until we find a near neighbor. If we don't end up in a leaf, or we do not find a neighbor, we fail.

B.2 Analysis

First, let us analyze the probability of success. Let $q \in S^{d-1}$ and $p \in P$ be the near neighbor ($\|p - q\| \leq \frac{\sqrt{2}}{c}$).

Lemma B.1. *If*

$$T \geq \frac{100}{\Pr_{z \sim N(0,1)^d}\left[\langle z, p \rangle \geq \eta \textit{ and } \langle z, q \rangle \geq \eta'\right]},$$

then the probability of successfully finding p *on query* q *is at least* 0.9*.*

Proof. We prove this by induction. Suppose the querying algorithm is at node v, where $p \in P_v$. We would like to prove that—if the conditions of the lemma are met—the probability of success is 0.9.

When v is a leaf, the statement is obvious. Suppose it is true for all the children of a node v, then

$$\Pr[\text{failure}] \leq \prod_{v' \text{ child of } v} \left(1 - \Pr_{z_{v'}}\left[\langle z_{v'}, p \rangle \geq \eta \text{ and } \langle z_{v'}, q \rangle \geq \eta'\right] \cdot 0.9\right)$$
$$= \left(1 - \Pr_{z \sim N(0,1)^d}\left[\langle z, p \rangle \geq \eta \text{ and } \langle z, q \rangle \geq \eta'\right] \cdot 0.9\right)^T \leq 0.1.$$

□

Now let us understand how much space the data structure occupies. In the lemma below, $u \in S^{d-1}$ is an arbitrary point.

Lemma B.2. *The expected space consumption of the data structure is at most*

$$n^{o(1)} \cdot T^K \left(1 + n \cdot \Pr_{z \sim N(0,1)^d}\left[\langle z, u \rangle \geq \eta\right]^K\right).$$

Proof. The total space the tree nodes occupy is $n^{o(1)} \cdot T^K$.

At the same time, every point u participates on average in $T^K \cdot \Pr_{z\sim N(0,1)^d}\left[\langle z, u\rangle \geq \eta\right]^K$ leaves, hence the desired bound. □

Finally, let us analyze the expected query time. As before, $u \in S^{d-1}$ is an arbitrary point.

Lemma B.3. *The expected query time is at most*

$$n^{o(1)} \cdot T^{K+1} \cdot \Pr_{z\sim N(0,1)^d}\left[\langle z, u\rangle \geq \eta'\right]^K \cdot \left(1 + n \cdot \Pr_{z\sim N(0,1)^d}\left[\langle z, u\rangle \geq \eta\right]^K\right).$$

Proof. First, a query touches at most $T^K \cdot \Pr_{z\sim N(0,1)^d}\left[\langle z, u\rangle \geq \eta'\right]^K$ tree nodes on average .

If a node is not a leaf, the time spent on it is at most $n^{o(1)} \cdot T$.

For a fixed leaf and a fixed dataset point, the probability that they end up in the leaf together with the query point is

$$\Pr_{z\sim N(0,1)^d}\left[\langle z, u\rangle \geq \eta'\right]^K \cdot \Pr_{z\sim N(0,1)^d}\left[\langle z, u\rangle \geq \eta\right]^K,$$

hence we obtain the desired bound. □

B.3 Setting parameters

First, we set $K = \sqrt{\log n}$. Second, we set $\eta > 0$ such that

$$\Pr_{z\sim N(0,1)^d}\left[\langle z, u\rangle \geq \eta\right] = n^{-1/K} = 2^{-\sqrt{\log n}}.$$

We can simply substitute the parameter setting of Lemma B.2 and Lemma B.3 This gives an expected space of

$$n^{o(1)} \cdot T^K$$

,

and an expected query time

$$n^{o(1)} \cdot T^{K+1} \cdot \Pr_{z\sim N(0,1)^d}\left[\langle z, u\rangle \geq \eta'\right]^K.$$

As discussed above in Lemma B.1, by setting

$$T \geq \frac{100}{\Pr_{z\sim N(0,1)^d}\left[\langle z, p\rangle \geq \eta \text{ and } \langle z, q\rangle \geq \eta'\right]},$$

the probability of success is 0.9.

In order to get the desired tradeoff, we can vary T and η'. Suppose we want space to be $n^{\rho_s + o(1)}$

for $\rho_s \geq 1$. Then we let

$$T = n^{\frac{\rho_s + o(1)}{K}} = 2^{(1+o(1))\cdot\rho_s\sqrt{\log n}},$$

and $\eta' > 0$ to be the largest number such that for every $p, q \in S^{d-1}$ with $\|p - q\| \leq \frac{\sqrt{2}}{c}$, we have

$$\mathrm{Pr}_{z\sim N(0,1)^d}\left[\langle z, p\rangle \geq \eta \text{ and } \langle z, q\rangle \geq \eta'\right] \geq \frac{100}{T} = 2^{-(1+o(1))\cdot\rho_s\sqrt{\log n}}.$$

Again, substituting in values of Lemma B.3, the query time is

$$n^{o(1)} \cdot T^{K+1} \cdot \mathrm{Pr}_{z\sim N(0,1)^d}\left[\langle z, u\rangle \geq \eta'\right]^K = n^{\rho_s + o(1)} \cdot \mathrm{Pr}_{z\sim N(0,1)^d}\left[\langle z, u\rangle \geq \eta'\right]^{\sqrt{\log n}}.$$

The trade-off between ρ_s and ρ_q follows from a standard computation of

$$\mathrm{Pr}_{z\sim N(0,1)^d}\left[\langle z, u\rangle \geq \eta'\right]$$

given that

$$\mathrm{Pr}_{z\sim N(0,1)^d}\left[\langle z, u\rangle \geq \eta\right] = 2^{-\sqrt{\log n}}$$

and

$$\mathrm{Pr}_{z\sim N(0,1)^d}\left[\langle z, p\rangle \geq \eta \text{ and } \langle z, q\rangle \geq \eta'\right] \geq 2^{-(1+o(1))\cdot\rho_s\sqrt{\log n}}.$$

The computation is relatively standard: see [AIL+15]. We verify next that the resulting trade-off is the same as (1) obtained in [Laa15c].

Denote α, β to be real numbers such that $\|(1,0) - (\alpha,\beta)\|_2 = \frac{\sqrt{2}}{c}$ and $\|(\alpha,\beta)\|_2 = 1$. Namely, $\alpha = 1 - \frac{1}{c^2}$ and $\beta = \sqrt{1-\alpha^2}$.

Lemma B.4. *Suppose that $\eta, \eta' > 0$ are such that $\eta, \eta' \to \infty$ and $\frac{\eta^2 + \eta'^2 - 2\alpha\eta\eta'}{\beta^2} \to \infty$. Then, for every $p, q \in S^{d-1}$ with $\|p - q\|_2 \leq \frac{\sqrt{2}}{c}$ one has:*

$$\mathrm{Pr}_{z\sim N(0,1)^d}\left[\langle z, p\rangle \geq \eta \textit{ and } \langle z, q\rangle \geq \eta'\right] = e^{-(1+o(1))\cdot\frac{\eta^2+\eta'^2-2\alpha\eta\eta'}{2\beta^2}},$$

and,

$$\mathrm{Pr}_{z\sim N(0,1)^d}\left[\langle z, p\rangle \geq \eta\right] = e^{-(1+o(1))\cdot\frac{\eta^2}{2}}.$$

Proof. Using spherical symmetry of Gaussians, we can reduce the computation to computing the Gaussian measure of the following *two-dimensional* set:

$$\{(x,y) \mid x \geq \eta' \text{ and } \alpha x + \beta y \geq \eta\}.$$

The squared distance from zero to the set is:

$$\frac{\eta^2 + \eta'^2 - 2\alpha\eta\eta'}{\beta^2}.$$

The result follows from the Appendix A of [AIL+15]. □

From the discussion above, we conclude that one can achieve the following trade-off between space $n^{\rho_s+o(1)}$ and query time $n^{\rho_q+o(1)}$:

$$1+\alpha^2\rho_s-\rho_q-2\alpha\sqrt{\rho_s-\rho_q}=0. \tag{78}$$

We now show that this is equivalent to the tradeoff of [Laa15c], i.e., (1), where $\rho_s=1+\rho_u$. Indeed, squaring (78) and replacing $\rho_s=1+\rho_u$, we get:

$$\big((1+\alpha^2)+\alpha^2\rho_u-\rho_q\big)^2=4\alpha^2(1+\rho_u-\rho_q), \tag{79}$$

or

$$(1+\alpha^2)^2+\alpha^4\rho_u^2+\rho_q^2+2\cdot((1+\alpha^2)\alpha^2\rho_u-(1+\alpha^2)\rho_q-\alpha^2\rho_u\rho_q)=4\alpha^2+4\alpha^2\rho_u-4\alpha^2\rho_q. \tag{80}$$

Simplifying the equation, we get

$$(1-\alpha^2)^2+\alpha^4\rho_u^2+\rho_q^2+2\cdot((\alpha^2-1)\alpha^2\rho_u-(1-\alpha^2)\rho_q-\alpha^2\rho_u\rho_q)=0. \tag{81}$$

Remember that we have $\alpha=1-1/c^2$, and hence $\alpha^2=\frac{(c^2-1)^2}{c^4}$ and $1-\alpha^2=\frac{2c^2-1}{c^4}$. We further obtain:

$$\tfrac{(2c^2-1)^2}{c^8}+\tfrac{(c^2-1)^4}{c^8}\rho_u^2+\rho_q^2-2\tfrac{(2c^2-1)(c^2-1)^2}{c^8}\rho_u-2\tfrac{2c^2-1}{c^4}\rho_q-2\tfrac{(c^2-1)^2}{c^4}\rho_u\rho_q=0, \tag{82}$$

or, multiplying by c^8,

$$(2c^2-1)^2+(c^2-1)^4\rho_u^2+c^8\rho_q^2-2(2c^2-1)(c^2-1)^2\rho_u-2(2c^2-1)c^4\rho_q-2(c^2-1)^2c^4\rho_u\rho_q=0. \tag{83}$$

In a similar fashion, squaring (1), we obtain:

$$c^4\rho_q+(c^2-1)^2\rho_u+2c^2(c^2-1)\sqrt{\rho_q\rho_u}=2c^2-1, \tag{84}$$

or equivalently,

$$2c^2(c^2-1)\sqrt{\rho_q\rho_u}=2c^2-1-c^4\rho_q-(c^2-1)^2\rho_u. \tag{85}$$

Squaring again, we obtain

$$\begin{aligned}4c^4(c^2-1)^2\rho_q\rho_u&=(2c^2-1)^2+c^8\rho_q^2+(c^2-1)^2\rho_u^2\\&\quad+2\cdot\left(c^4(c^2-1)^2\rho_q\rho_u-(2c^2-1)c^4\rho_q-(2c^2-1)(c^2-1)^2\rho_u\right),\end{aligned} \tag{86}$$

or, simplifying,

$$(2c^2-1)^2+c^8\rho_q^2+(c^2-1)^2\rho_u^2-2c^4(c^2-1)^2\rho_q\rho_u-2(2c^2-1)c^4\rho_q-2(2c^2-1)(c^2-1)^2\rho_u=0 \quad (87)$$

We now observe that we obtain the same equation as (83) and hence we are done proving that (78) is equivalent to (1).

C Upper Bound: General case

We show how to extend the result of [Laa15c] (and Appendix B) to the general case using the techniques of [AR15]. In particular, we show how to reduce a worst-case instance to several instances that are *random-like.* Overall the algorithm from below gives a data structure that solves the (c, r)-ANN problem in the d-dimensional Euclidean space, using space $O(n^{1+\rho_u+o(1)} + dn)$, and query time $O(dn^{\rho_q+o(1)})$ for any $\rho_u, \rho_q > 0$ that satisfy:

$$c^2\sqrt{\rho_q}+(c^2-1)\sqrt{\rho_u}=\sqrt{2c^2-1}. \quad (88)$$

As in [AR15], our data structure is a decision tree. However, there are several notable differences from [AR15]:

- The whole data structure is a *single* decision tree, while in [AR15] we consider a *collection* of $n^{\Theta(1)}$ trees.
- Instead of Spherical LSH used in [AR15], we use the partitioning procedure from Section B.
- In [AR15], one proceeds with partitioning a dataset until all parts contain less than $n^{o(1)}$ points. We change the stopping criterion slightly to ensure the number of "non-cluster" nodes[5] on any root-leaf branch is the same.
- Unlike [AR15], we do not use a "three-point property" of a random space partition in the analysis. This is related to the fact that the probability success of a single tree is constant, unlike [AR15], where it is polynomially small.
- In [AR15] we reduce the general case to the "bounded ball" case using LSH from [DIIM04]. Now we cannot quite do this, since we are aiming at getting a full time-space trade-off. Instead, we use a standard trick of imposing a randomly shifted grid, which reduces an arbitrary dataset to a dataset of diameter $\widetilde{O}(\sqrt{\log n})$ [IM98]. Then, we invoke an upper bound from [Laa15c] together with a reduction from [Val15], which for this case is enough to proceed.

[5] think $K = O(\sqrt{\log n})$ as in Section B.

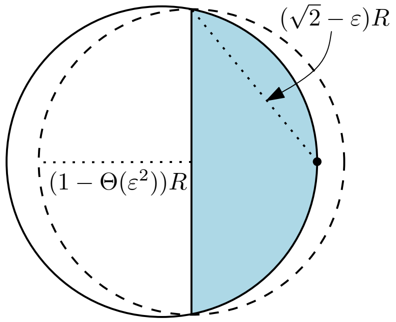

Figure 1: Covering a spherical cap of radius $(\sqrt{2}-\varepsilon)R$

C.1 Overview

We start with a high-level overview. Consider a dataset P_0 of n points. We can assume that $r = 1$ by rescaling. We may also assume that the dataset lies in the Euclidean space of dimension $d = \Theta(\log n \cdot \log \log n)$: one can always reduce the dimension to d by applying Johnson-Lindenstrauss lemma [JL84, DG03] while incurring distortion at most $1 + 1/(\log \log n)^{\Omega(1)}$ with high probability.

For simplicity, suppose that the entire dataset P_0 and a query lie on a sphere $\partial B(0, R)$ of radius $R = O_c(1)$. If $R \le c/\sqrt{2}$, we are done: this case corresponds to the "random instance" of points and we can apply the data structure from Section B.

Now suppose that $R > c/\sqrt{2}$. We split P_0 into a number of disjoint components: l *dense* components, termed $C_1, C_2, \ldots, C_l$, and one *pseudo-random* component, termed $\widetilde{P}$. The properties of these components are as follows. For each dense component C_i we require that $|C_i| \ge \tau n$ and that C_i can be covered by a spherical cap of radius $(\sqrt{2}-\varepsilon)R$ (see Fig. 1). Here $\tau, \varepsilon > 0$ are small quantities to be chosen later. The pseudo-random component $\widetilde{P}$ contains no more dense components inside.

We proceed separately for each C_i and $\widetilde{P}$. We enclose every dense component C_i in slightly smaller ball E_i of radius $(1-\Theta(\varepsilon^2))R$ (see Figure 1). For simplicity, let us first ignore the fact that C_i does not necessarily lie on the boundary ∂E_i. Once we enclose each dense cluster in a smaller ball, we recurse on each resulting spherical instance of radius $(1-\Theta(\varepsilon^2))R$. We treat the pseudo-random part $\widetilde{P}$ as described in Section B. we sample T Gaussian vectors $z_1, z_2, \ldots, z_T \sim N(0,1)^d$, where T is a parameter to be chosen later (for each pseudo-random remainder separately), and form T subsets of $\widetilde{P}$ as follows:

$$\widetilde{P}_i = \{p \in \widetilde{P} \mid \langle z_i, p \rangle \ge \eta R\},$$

where $\eta > 0$ is a parameter to be chosen later (for each pseudo-random remainder separately). Then we recurse on each $\widetilde{P}_i$. Note that after we recurse, there may appear new dense clusters in some sets $\widetilde{P}_i$ (e.g., since it may become easier to satisfy the minimum size constraint).

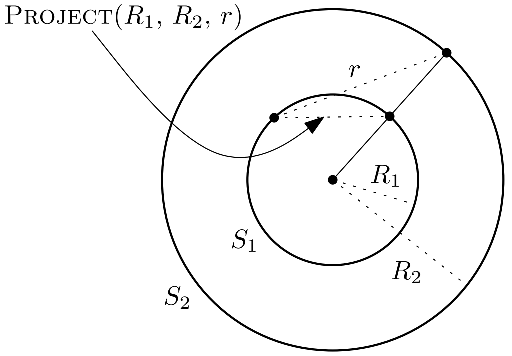

Figure 2: The definition of PROJECT

During the query procedure, we recursively query *each* C_i with the query point q. For the pseudo-random component $\widetilde{P}$, we identify all i's such that $\langle z_i, q\rangle \geq \eta' R$, and query all corresponding children recursively. Here $\eta' > 0$ is a parameter to be chosen later (for each pseudo-random remainder separately).

To analyze our algorithm, we show that we make progress in two ways. First, for dense clusters we reduce the radius of a sphere by a factor of $(1 - \Theta(\varepsilon^2))$. Hence, in $O_c(1/\varepsilon^2)$ iterations we must arrive to the case of $R \leq c/\sqrt{2}$, which is easy (as argued above). Second, for the pseudo-random component $\widetilde{P}$, we argue that most points lie at a distance $\geq (\sqrt{2} - \varepsilon)R$ from each other. In particular, the ratio of R to a typical inter-point distance is $\approx 1/\sqrt{2}$, exactly like in a random case. This is the reason we call $\widetilde{P}$ pseudo-random. This setting is where the data structure from Section B performs well.

We now address the issue deferred in the above high-level description: namely, that a dense component C_i does not generally lie on ∂E_i, but rather can occupy the interior of E_i. In this case, we partitioning E_i into very thin annuli of carefully chosen width δ and treat each annulus as a sphere. This discretization of a ball adds to the complexity of the analysis, but is not fundamental from the conceptual point of view.

C.2 Formal description

We are now ready to describe the data structure formally. It depends on the (small positive) parameters τ, ε and δ, as well as an integer parameter $K \sim \sqrt{\log n}$. We also need to choose parameters T, $\eta > 0$, $\eta' > 0$ for each pseudo-random remainder separately.

Preprocessing. Our preprocessing algorithm consists of the following functions:

- PROCESSSPHERE(P, r_1, r_2, o, R, k) builds the data structure for a dataset P that lies on a sphere $\partial B(o, R)$, assuming we need to solve ANN with distance thresholds r_1 and r_2.

Moreover, we are guaranteed that queries will lie on $\partial B(o, R)$. The parameter k is a counter which, in some sense, measures how far are we from being done.

- ProcessBall(P, r_1, r_2, o, R, k) builds the data structure for a dataset P that lies inside the ball $B(o, R)$, assuming we need to solve ANN with distance thresholds r_1 and r_2. Unlike ProcessSphere, here queries can be arbitrary. The parameter k has the same meaning as above.
- Process(P) builds the data structure for a dataset P to solve the general $(c, 1)$-ANN;
- Project(R_1, R_2, r) is an auxiliary function computing the following projection. Suppose we have two spheres S_1 and S_2 with a common center and radii R_1 and R_2. Suppose there are points $p_1 \in S_1$ and $p_2 \in S_2$ with $\|p_1 - p_2\| = r$. Project(R_1, R_2, r) returns the distance between p_1 and the point $\widetilde{p_2}$ that lies on S_1 and is the closest to p_2 (see Figure 2).

We now elaborate on algorithms in each of the above functions.

ProcessSphere. Function ProcessSphere follows the exposition from Section C.1. We consider three base cases. First, if $k = K$, then we stop and store the whole P. Second, if $r_2 \geq 2R$, then the goal can be achieved trivially, since any point from P works as an answer for any valid query. Third, if an algorithm from Section B would give a desired point on the time-space trade-off (in particular, if $r_2 \geq \sqrt{2}R$), then we just choose $\eta, \eta' > 0$ and T appropriately (in particular, we set $\eta > 0$ such that for any u, v with $\|u - v\| = r_2$ one has $\Pr_{z \sim N(0,1)^d}\left[\langle z, u\rangle \geq \eta R \text{ and } \langle z, v\rangle \geq \eta R\right] = n^{-1/K} = 2^{-\sqrt{\log n}}$) and make a single step .

Otherwise, we find dense clusters, i.e., non-trivially smaller balls, of radius $(\sqrt{2} - \varepsilon)R$, with centers on $\partial B(o, R)$ that contain many data points (at least $\tau|P|$). These balls can be enclosed into balls (with unconstrained center) of radius $\widetilde{R} \leq (1 - \Omega(\varepsilon^2))R$. For these balls we invoke ProcessBall with the same k. Then, for the remaining points we perform a single step of the algorithm from Section B with appropriate $\eta, \eta' > 0$ and T (in particular, we set $\eta > 0$ as above for the distance $\sqrt{2}R$), and recurse on each part with k increased by 1.

ProcessBall. First, we consider the following simple base case. If $r_1 + 2R \leq r_2$, then any point from $B(o, R)$ could serve as a valid answer to any query.

In general, we reduce to the spherical case via a discretization of the ball $B(o, R)$. First, we round all the distances to o up to a multiple of δ, which can change distance between any pair of points by at most 2δ (by the triangle inequality). Then, for every possible distance δi from o to a data point and every possible distance δj from o to a query (for admissible integers i, j), we build a separate data structure via ProcessSphere (we also need to check that $|\delta(i - j)| \leq r_1 + 2\delta$ to ensure that the corresponding pair (i, j) does not yield a trivial instance). We compute the new distance thresholds $\widetilde{r}_1$ and $\widetilde{r}_2$ for this data structure as follows. After rounding, the new thresholds for the ball instance should be $r_1 + 2\delta$ and $r_2 - 2\delta$, since distances can change by at most 2δ. To compute the final thresholds (after projecting the query to the sphere of radius δi), we just invoke Project (see the definition above).

Process. Process reduces the general case to the ball case. We proceed similarly to ProcessSphere, with two modifications. First, we apply a randomized partition using cubes with side $O_c(\sqrt{d}) = \widetilde{O}_c(\sqrt{\log n})$, and solve each part separately. Second, we seek to find dense clusters of radius $O_c(1)$. After there are no such clusters, we apply the reduction to unit-norm case from [Val15, Algorithm 25], and then (a single iteration of) the algorithm from Section B.

Project. This is implemented by a formula as in [AR15] (see Figure 2).

Overall, the preprocessing creates a decision tree, where the nodes correspond to procedures ProcessSphere, ProcessBall, Process. We refer to the tree nodes correspondingly, using the labels in the description of the query algorithm from below.

Query algorithm. Consider a query point $q \in \mathbb{R}^d$. We run the query on the decision tree, starting with the root, and applying the following algorithms depending on the label of the nodes:

- In Process we first recursively query the data structures corresponding to the clusters. Second, we locate q in the spherical caps, and query the data structure we built for the corresponding subsets of P.
- In ProcessBall, we first consider the base case, where we just return the stored point if it is close enough. In general, we check if $\|q - o\| \le R + r_1$. If not, we can return. Otherwise, we round q so that the distance from o to q is a multiple of δ. Next, we enumerate the distances from o to the potential near neighbor we are looking for, and query the corresponding ProcessSphere children after projecting q on the sphere with a tentative near neighbor (using, naturally, Project).
- In ProcessSphere, we proceed exactly the same way as Process modulo the base cases.
- In all the cases we try all the points if we store them explicitly (which happens when $k = K$).

C.3 How to set parameters

Here we briefly state how one sets the parameters of the data structure.

Recall that the dimension is $d = \Theta(\log n \cdot \log\log n)$. We set $\varepsilon, \delta, \tau$ as follows:

- $\varepsilon = \frac{1}{\log\log\log n}$;
- $\delta = \exp\bigl(-(\log\log\log n)^C\bigr)$;

- $\tau = \exp\bigl(-\log^{2/3} n\bigr)$,

where C is a sufficiently large positive constant.

Now we need to specify how to set $\eta, \eta' > 0$ and T for each pseudo-random remainder. The idea is to set η, η' and T such that

$$\mathrm{Pr}_{z \sim N(0,1)^d}\left[\langle z, u\rangle\right] = n^{-1/K} = 2^{-\sqrt{\log n}}$$

while, at the same time, for every u and v at distance at most r_1

$$T \sim \frac{100}{\mathrm{Pr}_{z \sim N(0,1)^d}\left[\langle z, u\rangle \ge \eta, \langle z, v\rangle \ge \eta'\right]}.$$

Finally, we choose T such that $T^K \sim n^{\rho_s + o(1)}$ where $\rho_s \ge 1$ is a parameter that governs the memory consumption.

This gives us a unique value of $\eta' > 0$, which governs the query time.

A crucial relation between parameters is that τ should be much smaller than $n^{-1/K} = 2^{-\sqrt{\log n}}$. This implies that the "large distance" is effectively equal to $\sqrt{2}R$, at least for the sake of a single step of the random partition.