Optimal Hashing-based Time–Space Trade-offs for Approximate Near Neighbors*

Alexandr Andoni
Columbia

Thijs Laarhoven
IBM Research Zürich

Ilya Razenshteyn
MIT CSAIL

Erik Waingarten
Columbia

March 21, 2017

Abstract

We show tight upper and lower bounds for time–space trade-offs for the c-Approximate Near Neighbor Search problem. For the d-dimensional Euclidean space and n-point datasets, we develop a data structure with space $n^{1+\rho_u+o(1)} + O(dn)$ and query time $n^{\rho_q+o(1)} + dn^{o(1)}$ for every $\rho_u, \rho_q \geq 0$ with:

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

For example, for the approximation $c = 2$ we can achieve:

- Space $n^{1.77\ldots}$ and query time $n^{o(1)}$, significantly improving upon known data structures that support very fast queries [IM98, KOR00];
- Space $n^{1.14\ldots}$ and query time $n^{0.14\ldots}$, matching the optimal data-dependent Locality-Sensitive Hashing (LSH) from [AR15];
- Space $n^{1+o(1)}$ and query time $n^{0.43\ldots}$, making significant progress in the regime of near-linear space, which is arguably of the most interest for practice [LJW$^+$07].

This is the first data structure that achieves sublinear query time and near-linear space for *every* approximation factor $c > 1$, improving upon [Kap15]. The data structure is a culmination of a long line of work on the problem for all space regimes; it builds on spherical Locality-Sensitive Filtering [BDGL16] and data-dependent hashing [AINR14, AR15].

Our matching lower bounds are of two types: conditional and unconditional. First, we prove tightness of the *whole* trade-off (1) in a restricted model of computation, which captures all known hashing-based approaches. We then show *unconditional* cell-probe lower bounds for one and two probes that match (1) for $\rho_q = 0$, improving upon the best known lower 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 the corresponding one-probe bound. To show the result for two probes, we establish and exploit a connection to *locally-decodable codes*.

*This paper merges two arXiv preprints: [Laa15c] (appeared online on November 24, 2015) and [ALRW16] (appeared online on May 9, 2016), and subsumes both of these articles. An extended abstract of this paper appeared in the proceedings of 28th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA '2017).

Contents

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 P of n points 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/Hamming $(\mathbb{R}^d, \ell_1)$ metric spaces have received the most attention. Besides 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 [MO15, Laa15a, Laa15b, BDGL16] and optimization [DRT11, HLM15, ZYS16].

The performance of an NNS data structure is primarily characterized by two key metrics:

- space: the amount of memory a data structure occupies, and
- query time: 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.

ANN does allow efficient data structures with a query time sublinear in n, and only polynomial dependence in d in all parameters [IM98, GIM99, KOR00, Ind01a, Ind01b, Cha02, CR04, DIIM04, Pan06, AI06, TT07, AC09, AINR14, Kap15, AR15, Pag16, BDGL16, ?, ?]. In practice, ANN algorithms are often successful 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 obtain tight time–space trade-offs for ANN in hashing-based models. Our upper bounds are stated in Section 1.5, and the lower bounds are stated in Section 1.6. We provide more background on the problem next.

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 dataset P according to the partition, and, given a query, retrieves all the data points which belong to the same part as the query. In order to return a near-neighbor with high probability of success, one maintains several partitions and checks all of them during the query stage. LSH yields data structures with space $O(n^{1+\rho} + dn)$ and query time $O(dn^{\rho})$, where ρ is the key quantity measuring the quality of the random space partition for a particular metric space and approximation $c \geq 1$. 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 Manhattan 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 random 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].

1.3 Time–space trade-offs

Since the early results on LSH, a natural question has been whether one can obtain query time vs. space trade-offs for a fixed approximation c. Indeed, data structures with *polynomial* space and *poly-logarithmic* query time were introduced [IM98, KOR00] simultaneously with LSH.

In practice, the most important regime is that of *near-linear* space, since space is usually a harder constraint than time: see, e.g., [LJW$^+$07]. The main question is whether it is possible to obtain near-linear space and sublinear query time. This regime has been studied since [Ind01a], with subsequent improvements in [Pan06, AI06, LJW$^+$07, Kap15, AIL$^+$15]. In particular, [LJW$^+$07, AIL$^+$15] introduce practical versions of the above theoretical results.

Despite significant progress in the near-linear space regime, no known algorithms obtain near-linear space and a sublinear query time for *all* approximations $c > 1$. For example, the best currently known algorithm of [Kap15] obtained query time of roughly $n^{4/(c^2+1)}$, which becomes trivial for $c < \sqrt{3}$.

1.4 Lower bounds

Lower bounds for NNS and ANN have also received considerable attention. Such lower bounds are ideally obtained in the *cell-probe* model [MNSW98, Mil99], where 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

[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.

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 algorithmic approaches.

Early lower bounds for NNS were obtained for data structures in *exact* or *deterministic* settings [BOR99, CCGL99, BR02, Liu04, JKKR04, CR04, PT06, Yin16]. [CR04, LPY16] obtained an almost tight cell-probe lower bound for the randomized Approximate *Nearest* Neighbor Search under the ℓ_1 distance. In that 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, so 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 ANN. 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 Manhattan distance and $n^{1+\Omega(1/tc^2)}$ for the Euclidean distance (for every $c > 1$).

Lower bounds have also been obtained for other metrics. For the ℓ_∞ distance, [ACP08] show a lower bound for deterministic ANN data structures. This lower bound was later generalized to randomized data structures [PTW10, KP12]. A recent result [AV15] adapts the framework of [PTW10] to Bregman divergences.

To prove higher lower bounds, researchers resorted to lower bounds for restricted models. These examples include: decision trees [ACP08] (the corresponding upper bound [Ind01b] is in the same model), LSH [MNP07, OWZ14, AIL$^+$15] and data-dependent LSH [AR16].

1.5 Our results: upper bounds

We give an algorithm obtaining the entire range of time–space tradeoffs, obtaining sublinear query time for all $c > 1$, for the entire space $\mathbb{R}^d$. Our main theorem is the following:

Theorem 1.1 (see Sections 3 and 4)**.** *For every $c > 1$, $r > 0$, $\rho_q \geq 0$ and $\rho_u \geq 0$ such that*

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

there exists a data structure for (c, r)-ANN for the Euclidean space $\mathbb{R}^d$, with space $n^{1+\rho_u+o(1)} + O(dn)$ and query time $n^{\rho_q+o(1)} + dn^{o(1)}$.

This algorithm has optimal exponents for all hashing-based algorithms, as well as one- and two-probe data structures, as we prove in later sections. In particular, Theorem 1.1 recovers or improves upon all earlier results on ANN in the entire time-space trade-off. For the near-linear

[2] The correct dependence on $1/\varepsilon$ requires the stronger Lopsided Set Disjointness lower bound from [Păt11].

space regime, setting $\rho_u = 0$, we obtain space $n^{1+o(1)}$ with query time $n^{\frac{2c^2-1}{c^4}+o(1)}$, which is sublinear for every $c > 1$. For $\rho_q = \rho_u$, we recover the best data-dependent LSH bound from [AR15], with space $n^{1+\frac{1}{2c^2-1}+o(1)}$ and query time $n^{\frac{1}{2c^2-1}+o(1)}$. Finally, setting $\rho_q = 0$, we obtain query time $n^{o(1)}$ and space $n^{\left(\frac{c^2}{c^2-1}\right)^2+o(1)}$, which, for $c = 1+\varepsilon$ with $\varepsilon \to 0$, becomes $n^{1/(4\varepsilon^2)+\cdots}$.

Using a reduction from [Ngu14], we obtain a similar trade-off for the ℓ_p spaces for $1 \le p < 2$ with c^2 replaced with c^p. In particular, for the ℓ_1 distance we get:

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

Our algorithms can support insertions/deletions with only logarithmic loss in space/query time, using the *dynamization* technique for decomposable search problems from [OvL81], achieving update time of $dn^{\rho_u+o(1)}$. To apply this technique, one needs to ensure that the preprocessing time is near-linear in the space used, which is the case for our data structure.

1.5.1 Techniques

We now describe the proof of Theorem 1.1 at a high level. It consists of two major stages. In the first stage, we give an algorithm for *random* Euclidean instances (introduced formally in Section 2). In the random Euclidean instances, we generate a dataset uniformly at random on a unit sphere $S^{d-1} \subset \mathbb{R}^d$ and plant a query at random within distance $\sqrt{2}/c$ from a randomly chosen data point. In the second stage, we show the claimed result for the *worst-case* instances by combining ideas from the first stage with data-dependent LSH from [AINR14, AR15].

Data-independent partitions. To handle random instances, we use a certain *data-independent* random process, which we briefly introduce below. It can be seen as a modification of spherical Locality-Sensitive Filtering from [BDGL16], and is related to a cell-probe *upper bound* from [PTW10]. While this data-independent approach can be extended to *worst case* instances, it gives a bound significantly worse than (2).

We now describe the random process which produces a decision tree to solve an instance of ANN on a *Euclidean unit sphere* $S^{d-1} \subset \mathbb{R}^d$. We take our initial dataset $P \subset S^{d-1}$ and sample T i.i.d. standard Gaussian d-dimensional vectors $z_1, z_2, \ldots, z_T$. The sets $P_i \subseteq P$ (not necessarily disjoint) are defined for each z_i as follows:

$$P_i = \{p \in P \mid \langle z_i, p\rangle \ge \eta_u\}.$$

We then recurse and repeat the above procedure for each non-empty P_i. We stop the recursion once we reach depth K. The above procedure generates a tree of depth K and degree at most T, where each leaf explicitly stores the corresponding subset of the dataset. To answer a query $q \in S^{d-1}$, we start at the root and descend into (potentially multiple) P_i's for which $\langle z_i, q\rangle \ge \eta_q$. When we eventually reach the K-th level, we iterate through all the points stored in the accessed leaves searching for a near neighbor.

The parameters T, K, η_u and η_q depend on the distance threshold r, the approximation factor c, as well as the desired space and query time exponents ρ_u and ρ_q. The special case of $\eta_u = \eta_q$ corresponds to the "LSH regime" $\rho_u = \rho_q$; $\eta_u < \eta_q$ corresponds to the "fast queries" regime $\rho_q < \rho_u$ (the query procedure is more selective); and $\eta_u > \eta_q$ corresponds to the "low memory" regime $\rho_u < \rho_q$. The analysis of this algorithm relies on bounds on the Gaussian area of certain two-dimensional sets [AR15, AIL+15], which are routinely needed for understanding "Gaussian" partitions.

This algorithm has two important consequences. First, we obtain the desired trade-off (2) for random instances by setting $r = \frac{\sqrt{2}}{c}$. Second, we obtain an inferior trade-off for *worst-case* instances of (c, r)-ANN over a unit sphere S^{d-1}. Namely, we get:

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

Even though it is inferior to the desired bound from (2)[3], it is already non-trivial. In particular, (3) is better than *all the prior work* on time–space trade-offs for ANN, including the most recent trade-off [Kap15]. Moreover, using a reduction from [Val15], we achieve the bound (3) for the whole $\mathbb{R}^d$ as opposed to just the unit sphere. Let us formally record it below:

Theorem 1.2. *For every $c > 1$, $r > 0$, $\rho_q \geq 0$ and $\rho_u \geq 0$ such that (3) holds, there exists a data structure for (c, r)-ANN for the* whole $\mathbb{R}^d$ *with space $n^{1+\rho_u+o(1)} + O(dn)$ and query time $n^{\rho_q+o(1)} + dn^{o(1)}$.*

Data-dependent partitions. We then improve Theorem 1.2 for worst-case instances and obtain the final result, Theorem 1.1. We build on the ideas of data-dependent LSH from [AINR14, AR15]. Using the reduction from [Val15], we may assume that the dataset and queries lie on a unit sphere S^{d-1}.

If pairwise distances between data points are distributed roughly like a random instance, we could apply the data-independent procedure. In absence of such a guarantee, we manipulate the dataset in order to reduce it to a random-looking case. Namely, we look for low-diameter clusters that contain many data points. We extract these clusters, and we enclose each of them in a ball of radius non-trivially smaller than one, and we recurse on each cluster. For the remaining points, which do not lie in any cluster, we perform one step of the data-independent algorithm: we sample T Gaussian vectors, form T subsets of the dataset, and recurse on each subset. Overall, we make progress in two ways: for the clusters, we make them a bit more isotropic after re-centering, which, after several re-centerings, makes the instance amenable to the data-independent algorithm, and for the remainder of the points, we can show that the absence of dense clusters makes the data-independent algorithm work for a single level of the tree (though, when recursing into P_i's, dense clusters may re-appear, which we will need to extract).

While the above intuition is very simple and, in hindsight, natural, the actual execution requires a good amount of work. For example, we need to formalize "low-diameter", "lots of points", "more

[3] See Figure 2 for comparison for the case $c = 2$.

isotropic", etc. However, compared to [AR15], we manage to simplify certain parts. For example, we do not need to analyze the behavior of Gaussian partitions on *triples* of points. While this was necessary in [AR15], we can avoid that analysis here, which makes the overall argument much cleaner. The algorithm still requires fine tuning of many moving parts, and we hope that it will be further simplified in the future.

Let us note that prior work suggested that time–space trade-offs might be possible with data-dependent partitions. To quote [Kap15]: "*It would be very interesting to see if similar [...to [AINR14] ...] analysis can be used to improve our tradeoffs*".

1.6 Our results: lower bounds

We show new *cell-probe* and *restricted* lower bounds for (c, r)-ANN matching our upper bounds. All our lower bounds rely on a certain canonical hard distribution for the Hamming space (defined later in Section 2). Via a standard reduction [LLR94], 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 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 1.3 (see Section 6.2)**.** *Any data structure that:*

- *solves (c, r)-ANN for the Hamming random instance (as defined in Section 2) with probability at least $2/3$,*
- *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.

The space lower bound matches:

- Our upper bound for *random instances* that can be made single-probe;
- Our upper bound for worst-case instances with query time $n^{o(1)}$.

The previous best lower bound from [PTW10] for a single probe are weaker by a polynomial factor.

We prove Theorem 1.3 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 for $\{-1, 1\}^d$ from [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 within distance r from a data point $p \in P$, if $P \setminus \{p\}$ is at distance at least cr from q, return 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 1.4 (see Section 8)**.** *Any data structure that:*

- *solves the decision ANN for the random instance (Section 2) 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, Theorem 1.4 shows 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 on 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 of a hypercube into the framework of [PTW10]. Thus, in order to obtain a higher lower bound for $t = 2$, we must 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 [PTW08]. In particular, we show that any data structure violating the lower bound from Theorem 1.4 implies a too-good-to-be-true 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 adjust their proof to the average case.

Finally, we point out an important difference with Theorem 1.3: in Theorem 1.4 we allow words to be merely of size $o(\log n)$ (as opposed to $n^{o(1)}$). Nevertheless, for the *decision version* of ANN for random instances our upper bounds 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 matching our upper bounds that up to $n^{o(1)}$ factors. Note that—since we show polynomial query time lower bounds—proving similar lower bounds *unconditionally* is far beyond the current reach of techniques. Any such statement would constitute a 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 1.5. *A* list-of-points data structure *for the ANN problem is defined as follows:*

- *We fix (possibly random) sets* $A_i \subseteq \{-1,1\}^d$*, for* $1 \le i \le m$*; also, with each possible query point* $q \in \{-1,1\}^d$*, 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 1.6 (see Section 7)**.** *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* $n^{1+\rho_u}$*, and has query time* $n^{\rho_q - o(1)}$*, for* $2/3$ *success probability. Then it must hold that:*

$$c\sqrt{\rho_q} + (c-1)\sqrt{\rho_u} \ge \sqrt{2c-1}. \tag{4}$$

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]. 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 1.6 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 $\rho_q = \rho_u$) has been recently shown in [AR16], and matches (4) for $\rho_q = \rho_u$.

1.7 Related work: past and concurrent

≪If we want, we can cite LSH forest paper and symmetric norms here –Ilya≫ IR

There have been many recent algorithmic advances on high-dimensional similarity search. The closest pair problem, which can seen as the off-line version of NNS/ANN, has received much attention recently [Val15, AW15, KKK16, KKKÓ16, ACW16]. ANN solutions with $n^{1+\rho_u}$ space (and preprocessing), and n^{ρ_q} query time imply closest pair problem with $O(n^{1+\rho_u} + n^{1+\rho_q})$ time (implying that the balanced, LSH regime is most relevant). Other work includes locality-sensitive filters [BDGL16] and LSH without false negatives [GPY94, Ind00, AGK06, Pag16, PP16]. See the surveys [HIM12, AI08, And09].

Relation to the article of [Chr17]. The article of [Chr17] has significant intersection with this paper (and, in particular, with the arXiv preprints [Laa15c, ALRW16] that are now merged to give this paper), as we explain next. In November 2015, [Laa15c] announced the optimal trade-off (i.e., Theorem 1.1) for random instances. As mentioned earlier, it is possible to extend this result to the entire Euclidean space, albeit with an inferior trade-off, from Theorem 1.2; for this, one can use a standard reduction á la [Val15] (this extension was not discussed in [Laa15c]). On May 9, 2016, both [Chr17] and [ALRW16] have been announced on arXiv. In [Chr17], the author also obtains an upper bound similar to Theorem 1.2 (trade-offs for the entire $\mathbb{R}^d$, but which are suboptimal), using a different (data-*independent*) reduction from the worst-case to the spherical case. Besides the upper bound, the author of [Chr17] also proved a conditional lower bound, similar to our lower bound from Theorem 1.6. This lower bound of [Chr17] is independent of our work in [ALRW16] (which is now a part of the current paper).

1.8 Open problems

We compile a list of exciting open problems:

- While our upper bounds are optimal (at least, in the hashing framework), the most general algorithms are, unfortunately, impractical. Our trade-offs for random instances on the sphere may well be practical (see also [BDGL16, Laa15a] for an experimental comparison with e.g. [Cha02, AIL$^+$15] for $\rho_q = \rho_u$), but a specific bottleneck for the extension to worst-case instances in $\mathbb{R}^d$ is the clustering step inherited from [AR15]. Can one obtain simple and practical algorithms that achieve the optimal time–space trade-off for these instances as well?

- Our new algorithms for the Euclidean case come tantalizingly close to the best known data structure for the ℓ_∞ distance [Ind01b]. Can we unify them and extend in a smooth way to the ℓ_p spaces for $2 < p < \infty$?

- Can we improve the dependence on the word size in the reduction from ANN data structures to LDCs used in the two-probe lower bound? As discussed above, the word size can not be pushed beyond $2^{\widetilde{\Theta}(\sqrt{\log n})}$ due to known constructions [DG15].

- A more optimistic view is that LDCs may provide a way to avoid the barrier posed by hashing-based approaches. We have shown that ANN data structures can be used to build weak forms of LDCs, and an intriguing open question is whether known LDC constructions can help with designing even more efficient ANN data structures.

2 Random instances

In this section, we introduce the *random* instances of ANN for the Hamming and Euclidean spaces. These instances play a crucial role for both upper bounds (algorithms) and the lower bounds in all the subsequent sections (as well as some prior work). For upper bounds, we focus on the Euclidean space, since algorithms for ℓ_2 yield the algorithms for the Hamming space using standard reductions. For the lower bounds, we focus on the Hamming space, since these yield lower bounds for the Euclidean space.

Hamming distance. We now describe a distribution supported on dataset-query pairs (P, q), where $P \subset \{-1, 1\}^d$ and $q \in \{-1, 1\}^d$. Random instances of ANN for the Hamming space will be dataset-query pairs drawn from this distribution.

- A dataset $P \subset \{-1, 1\}^d$ is given by n points, where each point is drawn independently and uniformly from $\{-1, 1\}^d$, where $d = \omega(\log n)$;
- A query $q \in \{-1, 1\}^d$ is drawn by first picking a dataset point $p \in P$ uniformly at random, and then flipping each coordinate of p independently with probability $\frac{1}{2c}$.
- The goal of the data structure is to preprocess P in order to recover the data point p from the query point q.

The distribution defined above is similar to the classic distribution introduced for the *light bulb problem* in [Val88], which can be seen as the *off-line* setting of ANN. This distribution has served as the hard distribution in many of the lower bounds for ANN mentioned in Section 1.4.

Euclidean distance. Now, we describe the distribution supported on dataset-query pairs (P, q), where $P \subset S^{d-1}$ and $q \in S^{d-1}$. Random instances of ANN for Euclidean space will be instances drawn from this distribution.

- A dataset $P \subset S^{d-1}$ is given by n unit vectors, where each vector is drawn independently and uniformly at random from S^{d-1}. We assume that $d = \omega(\log n)$, so pairwise distances are sufficiently concentrated around $\sqrt{2}$.
- A query $q \in S^{d-1}$ is drawn 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 in order to recover the data point p from the query point q.

Any data structure for $\left(c + o(1), \frac{\sqrt{2}}{c}\right)$-ANN over ℓ_2 must handle this instance. [AR15] showed how to reduce *any* (c, r)-ANN instance to several *pseudo*-random instances without increasing query time and space too much. These pseudo-random instances have the necessary properties of the random instance above in order for the data-independent algorithms (which are designed with the random instance in mind) to achieve optimal bounds. Similarly to [AR15], a data structure for these instances will lie at the core of our algorithm.

3 Upper bounds: data-independent partitions

3.1 Setup

For $0 < s < 2$, let $\alpha(s) = 1 - \frac{s^2}{2}$ be the cosine of the angle between two points on a unit Euclidean sphere S^{d-1} with distance s between them, and $\beta(s) = \sqrt{1 - \alpha^2(s)}$ be the sine of the same angle.

We introduce two functions that will be useful later. First, for $\rho > 0$, let

$$F(\rho) = \Pr_{z \sim N(0,1)^d} \left[\langle z, u \rangle \geq \rho\right],$$

where $u \in S^{d-1}$ is an arbitrary point on the unit sphere, and $N(0,1)^d$ is a distribution over $\mathbb{R}^d$, where coordinates of a vector are distributed as i.i.d. standard Gaussians. Note that $F(\rho)$ does not depend on the specific choice of u due to the spherical symmetry of Gaussians.

Second, for $0 < s < 2$ and $\rho, \sigma > 0$, let

$$G(s, \rho, \sigma) = \Pr_{z \sim N(0,1)^d} \left[\langle z, u \rangle \geq \rho \text{ and } \langle z, v \rangle \geq \sigma\right],$$

where $u, v \in S^{d-1}$ are arbitrary points from the unit sphere with $\|u - v\|_2 = s$. As with F, the value of $G(s, \rho, \sigma)$ does not depend on the specific points u and v; it only depends on the distance $\|u - v\|_2$ between them. Clearly, $G(s, \rho, \sigma)$ is non-increasing in s, for fixed ρ and σ.

We state two useful bounds on $F(\cdot)$ and $G(\cdot, \cdot, \cdot)$. The first is a standard tail bound for $N(0, 1)$ and the second follows from a standard computation (see the appendix of [AIL+15] for a proof).

Lemma 3.1. *For $\rho \to \infty$,*

$$F(\rho) = e^{-(1+o(1)) \cdot \frac{\rho^2}{2}}.$$

Lemma 3.2. *If $\rho, \sigma \to \infty$, then, for every s, one has:*

$$G(s, \rho, \sigma) = e^{-(1+o(1)) \cdot \frac{\rho^2 + \sigma^2 - 2\alpha(s)\rho\sigma}{2\beta^2(s)}}.$$

Finally, by using the Johnson–Lindenstrauss lemma [JL84, DG99] we can assume that $d = \Theta(\log n \cdot \log \log n)$ incurring distortion at most $1 + \frac{1}{\log^{\Omega(1)} \log n}$.

3.2 Results

Now we formulate the main result of Section 3, which we later significantly improve in Section 4.

Theorem 3.3. *For every $c > 1$, $r > 0$, $\rho_q \geq 0$ and $\rho_u \geq 0$ such that $cr < 2$ and*

$$(1 - \alpha(r)\alpha(cr))\sqrt{\rho_q} + (\alpha(r) - \alpha(cr))\sqrt{\rho_u} \geq \beta(r)\beta(cr), \tag{5}$$

there exists a data structure for (c, r)-ANN on a unit sphere $S^{d-1} \subset \mathbb{R}^d$ with space $n^{1+\rho_u+o(1)}$ and query time $n^{\rho_q+o(1)}$.

We instantiate Theorem 3.3 for two important cases. First, we get a single trade-off between ρ_q and ρ_u *for all $r > 0$ at the same time* by observing that (5) is the worst when $r \to 0$. Thus, we get a bound on ρ_q and ρ_u that depends on the approximation c only, which then can easily be translated to a result for the *whole* $\mathbb{R}^d$ using a reduction from [Val15].

Corollary 3.4. *For every $c > 1$, $r > 0$, $\rho_q \geq 0$ and $\rho_u \geq 0$ such that*

$$(c^2 + 1)\sqrt{\rho_q} + (c^2 - 1)\sqrt{\rho_u} \geq 2c, \tag{6}$$

there exists a data structure for (c, r)-ANN for the whole *$\mathbb{R}^d$ with space $n^{1+\rho_u+o(1)}$ and query time $n^{\rho_q+o(1)}$.*

Proof. We will show that we may transform an instance of (c, r)-ANN on $\mathbb{R}^d$ to an instance of $(c + o(1), r')$-ANN on the sphere with $r' \to 0$. When $r' \to 0$, we have:

$$\begin{aligned} 1 - \alpha(r')\alpha(cr') &= \frac{(c^2+1)r'^2}{2} + O_c(r'^4), \\ \alpha(r') - \alpha(cr') &= \frac{(c^2-1)r'^2}{2} + O_c(r'^4), \\ \beta(r')\beta(cr') &= cr'^2 + O_c(r'^4). \end{aligned}$$

Substituting these estimates into (5), we get (6).

Now let us show how to reduce ANN over $\mathbb{R}^d$ to the case, when all the points and queries lie on a unit sphere.

We first rescale all coordinates so as to assume $r = 1$. Now let us partition the whole space $\mathbb{R}^d$ into randomly shifted cubes with the side length $s = 10 \cdot \sqrt{d}$ and consider each cube separately. For any query $q \in \mathbb{R}^d$, with near neighbor $p \in P$,

$$\Pr[p \text{ and } q \text{ are in different cubes}] \leq \sum_{i=1}^{d} \frac{|p_i - q_i|}{s} = \frac{\|p - q\|_1}{s} \leq \frac{\sqrt{d} \cdot \|p - q\|_2}{s} \leq \frac{1}{10}.$$

The ℓ_2 diameter of a single cube is d. Consider one particular cube C, where we first translate points so $x \in C$ have $\|x\|_2 \leq d$. We let $\pi\colon C \to \mathbb{R}^{d+1}$ where

$$\pi(x) = (x, R),$$

where we append coordinate $R \gg d$ as the $(d+1)$-th coordinate. For any point $x \in C$,

$$\left\| \pi(x) - \left(\frac{R}{\|\pi(x)\|_2} \right) \cdot \pi(x) \right\|_2 \leq \frac{\|x\|_2^2}{2R}$$

and for any two points $x, y \in C$, $\|x-y\|_2 = \|\pi(x)-\pi(y)\|_2$; thus, $\left\| \left(\frac{R}{\|\pi(x)\|_2} \right) \pi(x) - \left(\frac{R}{\|\pi(y)\|_2} \right) \pi(y) \right\|_2 \leq \frac{d^2}{R} + \|x-y\|_2$. In addition, since $\left(\frac{R}{\|\pi(x)\|_2} \right) \pi(x)$ lies in a sphere of radius R for each point $x \in C$. Thus, letting $R = d^2 \cdot \log\log n \leq O(\log n^2 \log^3 \log n)$ (which is without loss of generality by the Johnson–Lindenstrauss Lemma), we get that an instance of (c, r)-ANN on $\mathbb{R}^d$ corresponds to an instance of $(c + o(1), \frac{1}{d^2 \log\log n})$-ANN on the surface of the unit sphere $S^d \subset \mathbb{R}^{d+1}$, where we lose $\frac{1}{10}$ in the success probability due to the division into disjoint cubes. Applying Theorem 3.3, we obtain the desired bound. □

If we instantiate Theorem 3.3 with inputs (dataset and query) drawn from the *random* instances defined in Section 2 (corresponding to the case $r = \frac{\sqrt{2}}{c}$), we obtain a significantly better tradeoff than (6). By simply applying Theorem 3.3, we give a trade-off for random instances matching the trade-off promised in Theorem 1.1.

Corollary 3.5. *For every $c > 1$, $\rho_q \geq 0$ and $\rho_u \geq 0$ such that*

$$c^2 \sqrt{\rho_q} + (c^2 - 1)\sqrt{\rho_u} \geq \sqrt{2c^2 - 1}, \tag{7}$$

there exists a data structure for $\left(c, \frac{\sqrt{2}}{c}\right)$-ANN on a unit sphere $S^{d-1} \subset \mathbb{R}^d$ with space $n^{1+\rho_u+o(1)}$ and query time $n^{\rho_q+o(1)}$. In particular, this data structure is able to handle random instances as defined in Section 2.

Proof. Follows from (5) and that $\alpha(\sqrt{2}) = 0$ and $\beta(\sqrt{2}) = 1$. □

Figure 2 plots the time-space trade-off in (6) and (7) for $c = 2$. Note that (7) is much better than (6), especially when $\rho_q = 0$, where (6) gives space $n^{2.77\ldots}$, while (7) gives much better space $n^{1.77\ldots}$. In Section 4, we show how to get best of both worlds: we obtain the trade-off (7) for *worst-case* instances. The remainder of the section is devoted to proving Theorem 3.3.

3.3 Data structure

3.3.1 Description

Fix K and T to be positive integers, we determine their exact value later. Our data structure is a *single* rooted tree where each node corresponds to a spherical cap. The tree consists of $K+1$ levels of nodes where each node has out-degree at most T. We will index the levels by 0, 1, ..., K, where the 0-th level consists of the root denoted by v_0, and each node up to the $(K-1)$-th level has at most T children. Therefore, there are at most T^K nodes at the K-th level.

For every node v in the tree, let $\mathcal{L}_v$ be the set of nodes on the path from v to the root v_0 excluding the root (but including v). Each node v, except for the root, stores a random Gaussian

```
function Build($P'$, $l$, $z$)
    create a tree node $v$
    store $l$ as $v.l$
    store $z$ as $v.z$
    if $l = K$ then
        store $P'$ as $v.P$
    else
        for $i \leftarrow 1 \ldots T$ do
            sample a Gaussian vector $z' \sim N(0,1)^d$
            $P'' \leftarrow \{p \in P' \mid \langle z', p\rangle \geq \eta_u\}$
            if $P'' \neq \emptyset$ then
                add Build($P''$, $l+1$, $z'$) as a child of $v$
    return $v$
```

```
function Query($q$, $v$)
    if $v.l = K$ then
        for $p \in v.P$ do
            if $\|p - q\| \leq cr$ then
                return $p$
    else
        for $v'$ : $v'$ is a child of $v$ do
            if $\langle v'.z, q\rangle \geq \eta_q$ then
                $p \leftarrow$ Query($q, v'$)
                if $p \neq \perp$ then
                    return $p$
    return $\perp$
```

Figure 1: Pseudocode for data-independent partitions

vector $z_v \sim N(0,1)^d$. For each node v, we define the following subset of the dataset $P_v \subseteq P$:

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

where $\eta_u > 0$ is a parameter to be chosen later.

At the root node v_0, $P_{v_0} = P$, since $\mathcal{L}_{v_0} = \emptyset$. Intuitively, each set P_v corresponds to a subset of the dataset lying in the intersection of spherical caps centered around $z_{v'}$ for all $v' \in \mathcal{L}_v$. Every leaf ℓ at the level K stores the subset P_ℓ explicitly.

We build the tree recursively. For a given node v in levels $0, \ldots, K-1$, we first sample T i.i.d. Gaussian vectors $g_1, g_2, \ldots, g_T \sim N(0,1)^d$. Then, for every i such that $\{p \in P_v \mid \langle g_i, p\rangle \geq \eta_u\}$ is non-empty, we create a new child v' with $z_{v'} = g_i$ and recursively process v'. At the K-th level, each node v stores P_v as a list of points.

In order to process a query $q \in S^{d-1}$, we start from the root v_0 and descend down the tree. We consider every child v of the root for which $\langle z_v, q\rangle \geq \eta_q$, where $\eta_q > 0$ is another parameter to be chosen later[4]. After identifying all the children, we proceed down the children recursively. If we reach leaf ℓ at level K, we scan through all the points in P_ℓ and compute their distance to the query q. If a point lies at a distance at most cr from the query, we return it and stop.

We provide pseudocode for the data structure above in Figure 1. The procedure Build(P, 0, $\perp$) builds the data structure for dataset P and returns the root of the tree, v_0. The procedure Query(q, v_0) queries the data structure with root v_0 at point q.

3.3.2 Analysis

Probability of success We first analyze the probability of success of the data structure. We assume that a query q has some $p \in P$ where $\|p - q\|_2 \leq r$. The data structure succeeds when Query(q, v_0) returns some point $p' \in P$ with $\|q - p'\|_2 \leq cr$.

[4] Note that η_u may not be equal to η_q. It is exactly this discrepancy that will govern the time–space trade-off.

Lemma 3.6. *If*

$$T \geq \frac{100}{G\left(r, \eta_u, \eta_q\right)},$$

then with probability at least 0.9*,* Query$(q,\, v_0)$ *finds some point within distance cr from q.*

Proof. We prove the lemma by induction on the depth of the tree. Let $q \in S^{d-1}$ be a query point and $p \in P$ its near neighbor. Suppose we are within the recursive call Query$(q,\, v)$ for some node v in the tree. Suppose we have not yet failed, that is, $p \in P_v$. We would like to prove that—if the condition of the lemma is met—the probability that this call returns *some* point within distance cr is at least 0.9.

When v is a node in the last level K, the algorithm enumerates P_v and, since we assume $p \in P_v$, some good point will be discovered (though not necessarily p itself). Therefore, this case is trivial. Now suppose that v is not from the K-th level. Using the inductive assumption, suppose that the statement of the lemma is true for all T potential children of v, i.e., if $p \in P_{v'}$, then with probability 0.9, Query(q, v') returns some point within distance cr from q. Then,

$$\begin{aligned} \Pr[\text{failure}] &\leq \prod_{i=1}^{T} \left(1 - \Pr_{z_{v_i}}\left[\langle z_{v_i}, p\rangle \geq \eta_u \text{ and } \langle z_{v_i}, q\rangle \geq \eta_q\right] \cdot 0.9\right) \\ &\leq \left(1 - G\left(r, \eta_u, \eta_q\right) \cdot 0.9\right)^T \leq 0.1, \end{aligned}$$

where the first step follows from the inductive assumption and independence between the children of v during the preprocessing phase. The second step follows by monotonicity of $G(s, \rho, \sigma)$ in s, and the third step is due to the assumption of the lemma. □

Space We now analyze the space consumption of the data structure.

Lemma 3.7. *The expected space consumption of the data structure is at most*

$$n^{1+o(1)} \cdot K \cdot \left(T \cdot F(\eta_u)\right)^K.$$

Proof. We compute the expected total size of the sets P_ℓ for leaves ℓ at K-th level. There are at most T^K such nodes, and for a fixed point $p \in P$ and a fixed leaf ℓ the probability that $p \in P_\ell$ is equal to $F(\eta_u)^K$. Thus, the expected total size is at most $n \cdot \left(T \cdot F(\eta_u)\right)^K$. Since we only store a node v if P_v is non-empty, the number of nodes stored is at most $K + 1$ times the number of points stored at the leaves. The Gaussian vectors stored at each node require space d, which is at most $n^{o(1)}$. □

Query time Finally, we analyze the query time.

Lemma 3.8. *The expected query time is at most*

$$n^{o(1)} \cdot T \cdot \left(T \cdot F(\eta_q)\right)^K + n^{1+o(1)} \cdot \left(T \cdot G(cr, \eta_u, \eta_q)\right)^K. \tag{8}$$

Proof. First, we compute the expected query time spent going down the tree, without scanning the leaves. The expected number of *nodes* the query procedure reaches is:

$$1 + T \cdot F(\eta_q) + (T \cdot F(\eta_q))^2 + \ldots + (T \cdot F(\eta_q))^K = O(1) \cdot (T \cdot F(\eta_q))^K ,$$

since we will set T so $T \cdot F(\eta_q) \geq 100$. In each of node, we spend time $n^{o(1)} \cdot T$. The product of the two expressions gives the first term in (8).

The expected time spent scanning points in the leaves is at most $n^{o(1)}$ times the number of points scanned at the leaves reached. The number of points scanned is always at most one more than the number of *far* points, i.e., lying a distance greater than cr from q, that reached the same leaf. There are at most $n - 1$ far points and T^K leaves. For each far point p' and each leaf ℓ the probability that both p' and q end up in P_ℓ is at most $G(cr, \eta_u, \eta_q)^K$. For each such pair, we spend time at most $n^{o(1)}$ processing the corresponding p'. This gives the second term in (8). □

3.3.3 Setting parameters

We end the section by describing how to set parameters T, K, η_u and η_q to prove Theorem 3.3.

First, we set $K \sim \sqrt{\ln n}$. In order to satisfy the requirement of Lemma 3.6, we set

$$T = \frac{100}{G(r, \eta_u, \eta_q)}. \tag{9}$$

Second, we (approximately) balance the terms in the query time (8). Toward this goal, we aim to have

$$F(\eta_q)^K = n \cdot G(cr, \eta_u, \eta_q)^K. \tag{10}$$

If we manage to satisfy these conditions, then we obtain space $n^{1+o(1)} \cdot (T \cdot F(\eta_u))^K$ and query time[5] $n^{o(1)} \cdot (T \cdot F(\eta_q))^K$.

Let $F(\eta_u)^K = n^{-\sigma}$ and $F(\eta_q)^K = n^{-\tau}$. By Lemma 3.1, Lemma 3.2 and (10), we have that, up to $o(1)$ terms,

$$\tau = \frac{\sigma + \tau - 2\alpha(cr) \cdot \sqrt{\sigma\tau}}{\beta^2(cr)} - 1,$$

which can be rewritten as

$$\left|\sqrt{\sigma} - \alpha(cr)\sqrt{\tau}\right| = \beta(cr), \tag{11}$$

since $\alpha^2(cr) + \beta^2(cr) = 1$. We have, by Lemma 3.1, Lemma 3.2 and (9),

$$T^K = n^{\frac{\sigma+\tau-2\alpha(r)\sqrt{\sigma\tau}}{\beta^2(r)} + o(1)}.$$

Thus, the space bound is

$$n^{1+o(1)} \cdot (T \cdot F(\eta_u))^K = n^{1 + \frac{\sigma+\tau-2\alpha(r)\sqrt{\sigma\tau}}{\beta^2(r)} - \sigma + o(1)} = n^{1 + \frac{(\alpha(r)\sqrt{\sigma}-\sqrt{\tau})^2}{\beta^2(r)} + o(1)}$$

[5] Other terms from the query time are absorbed into $n^{o(1)}$ due to our choice of K.

and query time is

$$n^{o(1)} \cdot (T \cdot F(\eta_q))^K = n^{\frac{\sigma+\tau-2\alpha(r)\sqrt{\sigma\tau}}{\beta^2(r)} - \tau + o(1)} = n^{\frac{(\sqrt{\sigma}-\alpha(r)\sqrt{\tau})^2}{\beta^2(r)} + o(1)}.$$

In other words,

$$\rho_q = \frac{\left(\sqrt{\sigma} - \alpha(r)\sqrt{\tau}\right)^2}{\beta^2(r)},$$

and

$$\rho_u = \frac{\left(\alpha(r)\sqrt{\sigma} - \sqrt{\tau}\right)^2}{\beta^2(r)}$$

where τ is set so that (11) is satisfied. Combining these identities, we obtain (5).

Namely, we set $\sqrt{\sigma} = \alpha(cr)\sqrt{\tau} + \beta(cr)$ to satisfy (11). Then, $\sqrt{\tau}$ can vary between:

$$\frac{\alpha(r)\beta(cr)}{1 - \alpha(r)\alpha(cr)},$$

which corresponds to $\rho_u = 0$ and

$$\frac{\beta(cr)}{\alpha(r) - \alpha(cr)},$$

which corresponds to $\rho_q = 0$.

This gives a relation:

$$\sqrt{\tau} = \frac{\beta(cr) - \beta(r)\sqrt{\rho_q}}{\alpha(r) - \alpha(cr)} = \frac{\alpha(r)\beta(cr) + \beta(r)\sqrt{\rho_u}}{1 - \alpha(r)\alpha(cr)},$$

which gives the desired trade-off (5).

3.4 An algorithm based on Locality-Sensitive Filtering (LSF)

We will now briefly describe an alternative algorithm to the one above, which is based on the *Spherical Locality-Sensitive Filtering* introduced in [BDGL16].[6] While it achieves the same bounds, it has a couple of potential advantages: 1) it may be more practical and 2) it naturally extends to the $d = O(\log n)$ case with somewhat better trade-offs between ρ_q, ρ_u than in (2) (such better exponents were already obtained in [BDGL16] for the "LSH regime" of $\rho_u = \rho_q$).

For spherical LSF, in the notation of the construction described above, partitions are formed by first dividing $\mathbb{R}^d$ into K blocks ($\mathbb{R}^d = \mathbb{R}^{d/K} \times \cdots \times \mathbb{R}^{d/K}$), and then generating a spherical code $C \subset S^{d/K-1} \subset \mathbb{R}^{d/K}$ of vectors sampled uniformly at random from the lower-dimensional unit sphere $S^{d/K-1}$. For any vector $p \in \mathbb{R}^d$, we write $p^{(1)}, \ldots, p^{(K)}$ for the K blocks of d/K coordinates in the vector p. For simplicity, let us assume that d is a multiple of K.

Similar to the tree-based construction above, we then generate a tree of vectors and subsets as follows. The tree consists of K levels, and the $|C|$ children of a node v at level ℓ are defined by the

[6] As a historical note, we remark that the algorithm from this section was the one to inspire the tree-based algorithm.

vectors $(0, \ldots, 0, z_i, 0, \ldots, 0)$, where only the ℓ-th block of d/K entries is potentially non-zero and is formed by one of the $|C|$ code words. The subset P'' of a child then corresponds to the subset P' of the parent, intersected with the spherical cap corresponding to the child. In other words, at the lowest level K a leaf v typically contains a subset $P' \subset P$ satisfying

$$P' = \{p \in P : \langle z_{i_1}, p^{(1)} \rangle \geq \eta_u, \ldots, \langle z_{i_K}, p^{(K)} \rangle \geq \eta_u\}, \tag{12}$$

where the (indices of the) code words $z_{i_1}, \ldots, z_{i_K}$ depend on the path to the root of the tree. It was then shown in [BDGL16] that this approach of intersecting spherical caps is asymptotically equivalent to the following, slightly different definition of the subsets associated to the leaves:

$$\hat{P}' = \{p \in P : \langle z_{i_1}, p^{(1)} \rangle + \cdots + \langle z_{i_K}, p^{(K)} \rangle \geq K \cdot \eta_u\}. \tag{13}$$

In other words, decoding each of the K blocks separately with threshold η_u was shown to be asymptotically equivalent to decoding the entire vector with threshold $K \cdot \eta_u$, as long as K does not grow too fast as a function of d and n. The latter joint decoding method based on the sum of the partial inner products is then used as the actual decoding method.

Let us highlight the difference between the previous tree-based algorithm and the algorithm in this section. Besides the difference between using Gaussian vectors and spherical (unit) vectors (a difference which is asymptotically negligible), the method for partitioning the sphere and the efficient decoding algorithm are different for these two methods. In short, both methods would like to use uniformly random, small spherical caps for the partitioning of the sphere, but clearly the decoding costs for such a method would be too high. Both methods then make a different concession as follows:

- The concession made in the Gaussian tree-based solution is to let leaves correspond to intersections of spherical caps, so that many of the leaves can be discarded as soon as a vector is not included in one of the spherical caps higher up the tree. This further guarantees that all z_i can still be chosen at random. However, one would prefer to use small *single* spherical caps instead of intersections of a few larger spherical caps.

- In the LSF-based solution, the leaves in the tree still correspond to single (small) spherical caps, but for decoding efficiently, additional structure is introduced in the vectors defining these spherical caps. These vectors are no longer all randomly sampled from a Gaussian or from the sphere, but can be seen as code words from a random product code [BDGL16, Section 5]. In this case, no concession is done in terms of the shape of the region, but a concession is made in terms of the randomness of the spherical-caps-defining vectors z_i.

Although both constructions achieve the same asymptotic performance, it was already argued in [BDGL16] that the combined decoding approach instead of decoding blocks separately (i.e. single small spherical caps vs. intersections of several larger spherical caps) seems to lead to much-improved results in practice.

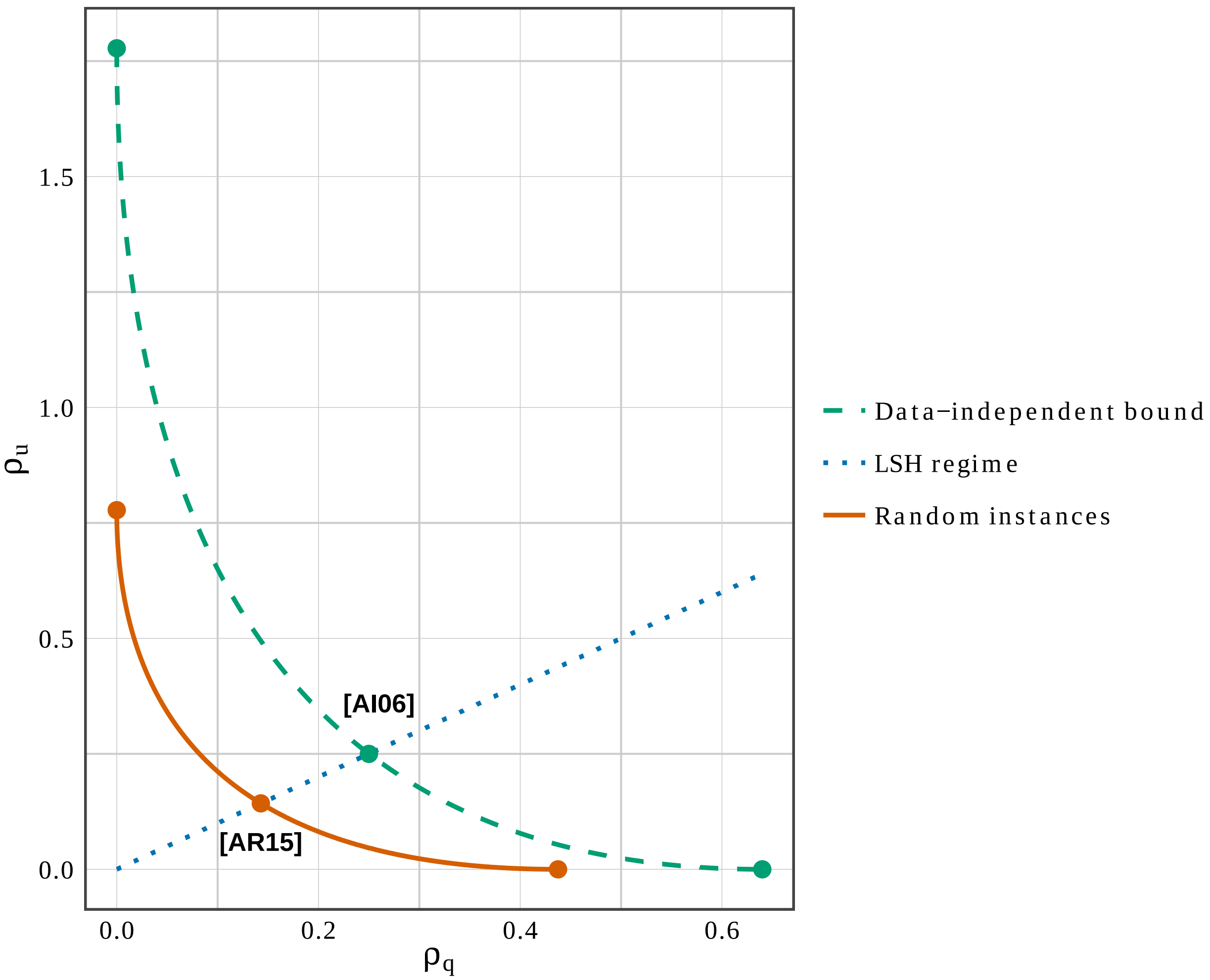

Figure 2: Trade-offs between query time $n^{\rho_q+o(1)}$ and space $n^{1+\rho_u+o(1)}$ for the Euclidean distance and approximation $c = 2$. The green dashed line corresponds to the simple data-independent bound for *worst-case* instances from Corollary 3.4. The red solid line corresponds to the bound for *random* instances from Corollary 3.5, which we later extend to *worst-case* instances in Section 4. The blue dotted line is $\rho_q = \rho_u$, which corresponds to the "LSH regime". In particular, the intersection of the dotted and the dashed lines matches the best *data-independent* LSH from [AI06], while the intersection with the solid line matches the best *data-dependent* LSH from [AR15].

4 Upper bounds: data-dependent partitions

In this section we prove the main upper bound theorem, Theorem 1.1, which we restate below:

Theorem 4.1. *For every $c > 1$, $r > 0$, $\rho_q \geq 0$ and $\rho_u \geq 0$ such that*

$$c^2\sqrt{\rho_q} + (c^2 - 1)\sqrt{\rho_u} \geq \sqrt{2c^2 - 1}, \tag{14}$$

there exists a data structure for (c, r)-ANN for the whole $\mathbb{R}^d$ *with space $n^{1+\rho_u+o(1)} + O(dn)$ and query time $n^{\rho_q+o(1)} + dn^{o(1)}$.*

This theorem achieves "the best of both worlds" in Corollary 3.4 and Corollary 3.5. Like Corollary 3.4, our data structure works for worst-case datasets; however, we improve upon the trade-off between time and space complexity from Corollary 3.4 to that of random instances in Corollary 3.5. See Figure 2 for a comparison of both trade-offs for $c = 2$. We achieve the improvement by combining the result of Section 3 with the techniques from [AR15].

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

- The whole data structure is a *single* decision tree, while [AR15] considers a *collection* of $n^{\Theta(1)}$ trees.
- Instead of Spherical LSH used in [AR15], we use the partitioning procedure from Section 3.
- In [AR15], the algorithm continues partitioning the 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 on any root-leaf branch is the same (this value will be around $\sqrt{\ln n}$ to reflect the setting of K in Section 3).
- Unlike [AR15], our analysis does not require the "three-point property", which is necessary in [AR15]. 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], the algorithm reduces the general case to the "bounded ball" case using LSH from [DIIM04]. While the cost associated with this procedure is negligible in the LSH regime, the cost becomes too high in certain parts of the 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 $\widetilde{O}(\log n)$ (see the proof of Corollary 6 and [IM98]). Then, we invoke an upper bound from Section 3 together with a reduction from [Val15] which happens to be enough for this case.

4.1 Overview

We start with a high-level overview. Consider a dataset P_0 of n points. We may assume $r = 1$ by rescaling. We may further assume the dataset lies in the Euclidean space of dimension $d =$

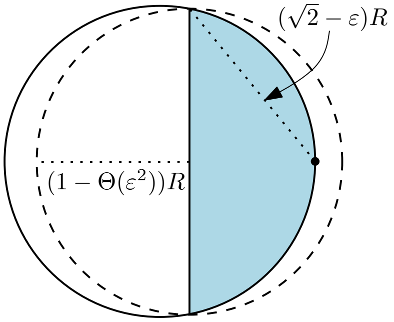

Figure 3: Covering a spherical cap of radius $(\sqrt{2}-\varepsilon)R$

$\Theta(\log n \cdot \log\log n)$; one can always reduce the dimension to d by applying the Johnson–Lindenstrauss lemma [JL84, DG03] which reduces the dimension and distorts pairwise distances by at most $1 \pm 1/(\log\log n)^{\Omega(1)}$ with high probability. We may also assume the entire dataset P_0 and a query lie on a sphere $\partial B(0,R)$ of radius $R = \widetilde{O}(\log^2 n)$ (see the proof of Corollary 6).

We partition P_0 into various components: s *dense* components, denoted by C_1, C_2, …, C_s, and one *pseudo-random* component, denoted by $\widetilde{P}$. The partition is designed to satisfy the following properties. Each dense component C_i satisfies $|C_i| \geq \tau n$ and can be covered by a spherical cap of radius $(\sqrt{2}-\varepsilon)R$ (see Figure 3). Here $\tau, \varepsilon > 0$ are small quantities to be chosen later. One should think of C_i as clusters consisting of $n^{1-o(1)}$ points which are closer than random points would be. The pseudo-random component $\widetilde{P}$ consists of the remaining points without any dense clusters inside.

We proceed separately for each C_i and $\widetilde{P}$. We enclose every dense component C_i in a smaller ball E_i of radius $(1-\Omega(\varepsilon^2))R$ (see Figure 3). For simplicity, we first ignore the fact that C_i does not necessarily lie on the boundary ∂E_i. Once we enclose each dense cluster with a smaller ball, we recurse on each resulting spherical instance of radius $(1-\Omega(\varepsilon^2))R$. We treat the pseudo-random component $\widetilde{P}$ similarly to the random instance from Section 2 described in Section 3. Namely, we sample T Gaussian vectors $z_1, z_2, \ldots, z_T \sim N(0,1)^d$, and form T subsets of $\widetilde{P}$:

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

where $\eta_u > 0$ 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, new dense clusters may appear in some $\widetilde{P}_i$ since it becomes easier to satisfy the minimum size constraint.

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 $\langle z_i, q\rangle \geq \eta_q R$, and query all corresponding children recursively. Here T, $\eta_u > 0$ and $\eta_q > 0$ are parameters that need to be chosen carefully (for each pseudo-random remainder separately).

Our algorithm makes progress in two ways. For dense clusters, we reduce the radius of the enclosing sphere by a factor of $(1-\Omega(\varepsilon^2))$. Ideally, we have that initially $R = \widetilde{O}(\log^2 n)$, so in $O(\log\log n/\varepsilon^2)$ iterations of removing dense clusters, we arrive at the case of $R \leq c/\sqrt{2}$, where

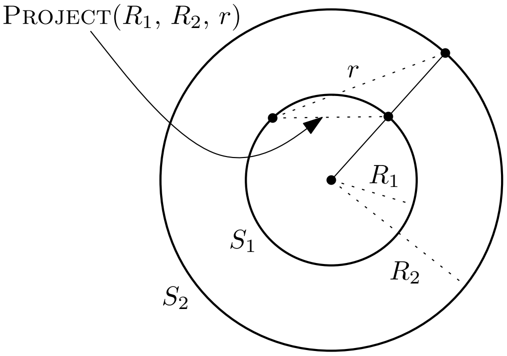

Figure 4: The definition of PROJECT

Corollary 3.5 begins to apply. For the pseudo-random component $\widetilde{P}$, most points will lie a distance at least $(\sqrt{2} - \varepsilon)R$ from each other. In particular, the ratio of R to a typical inter-point distance is approximately $1/\sqrt{2}$, exactly like in a random case. For this reasonm we call $\widetilde{P}$ pseudo-random. In this setting, the data structure from Section 3 performs well.

We now address the issue deferred in the above high-level description: 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 partition E_i into very thin annuli of carefully chosen width 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.

4.2 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 $K \sim \sqrt{\ln n}$. We also need to choose parameters T, $\eta_u > 0$, $\eta_q > 0$ for each pseudo-random remainder separately. Figure 5 provides pseudocode for the algorithm.

Preprocessing. Our preprocessing algorithm consists of the following functions:

- PROCESS(P) does the initial preprocessing. In particular, it performs the rescaling so that $r_1 = 1$ as well as the dimension reduction to $d = \Theta(\log n \log\log n)$ with the Johnson–Lindenstrauss lemma [JL84, DG03]. In addition, we partition into randomly shifted cubes, translate the points, and think of them as lying on a sphere of radius $R = \widetilde{O}(\log^2 n)$ (see the proof of Corollary 6 for details). Then we call PROCESSSPHERE.

- PROCESSSPHERE(P, r_1, r_2, o, R, l) builds the data structure for a dataset P lying 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 l counts the number of non-cluster nodes in the recursion stack we have encountered so far. Recall that we stop as soon as we encounter K of them.

- ProcessBall(P, r_1, r_2, o, R, l) builds the data structure for a dataset P lying 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 l has the same meaning as in ProcessSphere.
- Project(R_1, R_2, r) is an auxiliary function allowing us to project points on a ball to very thin annuli. 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 4). This is implemented by a formula as in [AR15].

We now elaborate on the above descriptions of ProcessSphere and ProcessBall, since these are the crucial components of our analysis. We will refer to line number of the pseudocode from Figure 5.

ProcessSphere. We consider three base cases.

1. If $l = K$, we stop and store P explicitly. This corresponds to having reached a leaf in the algorithm from Section 3. This case is handled in lines 2-4 of Figure 5.
2. If $r_2 \geq 2R$, then we may only store one point, since any point in P is a valid answer to any query made on a sphere of radius R containing P. This trivial instance is checked in lines 5-7 of Figure 5.
3. The last case occurs when the algorithm from Section 3 can give the desired point on the time–space trade-off. In this case, we may simply proceed as in the algorithm from Section 3. We choose $\eta_u, \eta_q > 0$ and T appropriately and build a single level of the tree from Section 3 with l increased by 1. We check for this last condition using (5) in line 9 of Figure 5, and if so, we may skip lines 10-16 of Figure 5.

If none of the above three cases apply, we proceed in lines 10-16 of Figure 5 by removing the dense components and then handling the pseudo-random remainder. The dense components are clusters of at least $\tau|P|$ points lying in a ball of radius $(\sqrt{2} - \varepsilon)R$ with its center on $\partial B(o, R)$. These balls can be enclosed by smaller balls of radius $\widetilde{R} \leq (1 - \Omega(\varepsilon^2))R$. In each of these smaller balls, we invoke ProcessBall with the same l. Finally, we build a single level of the tree in Section 3 for the remaining pseudo-random points. We pick the appropriate $\eta_u, \eta_q > 0$ and T and recurse on each part with ProcessSphere with l increased by 1.

ProcessBall. Similarly to ProcessSphere, if $r_1 + 2R \leq r_2$, then any point from $B(o, R)$ is a valid answer to any query in $B(o, R + r_2)$. We handle this trivial instance in lines 25-27 of Figure 5.

If we are not in the trivial setting above, we reduce the ball to the spherical case via a discretization of the ball $B(o, R)$ into thin annuli of width δr_1. First, we round all distances from points to o to a multiple of δr_1 in line 28 of Figure 5. This rounding can change the distance between any pair of points by at most $2\delta r_1$ by the triangle inequality.

Then, we handle each non-empty annuli separately. In particular, for a fixed annuli at distance $\delta i r_1$ from o, a possible query can lie at most a distance $\delta j r_1$ from o, where $\delta r_1 |i - j| \leq r_1 + 2\delta r_1$. For each such case, we recursively build a data structure with ProcessSphere. However, when projecting points, the distance thresholds of r_1 and r_2 change, and this change is computed using Project in lines 34 and 35 of Figure 5.

Overall, the preprocessing creates a decision tree. The root corresponds to the procedure Process, and subsequent nodes correspond to procedures ProcessSphere and ProcessBall. 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 which executes Process, and applying the following algorithms depending on the label of the nodes:

- In ProcessSphere we first recursively query the data structures corresponding to the clusters. Then, we locate q in the spherical caps (with threshold η_q, like in Section 3), and query data structures we built for the corresponding subsets of P. When we encounter a node with points stored explicitly, we simply scan the list of points for a possible near neighbor. This happens when $l = K$.

- In ProcessBall, we first consider the base case, where we just return the stored point if it is close enough. In general, we check whether $\|q - o\|_2 \leq R + r_1$. If not, we return with no neighbor, since each dataset point lies within a ball of radius R from o, but the query point is at least $R + r_1$ away from o. If $\|q - o\|_2 \leq R + r_1$, we round q so the distance from o to q is a multiple of δr_1 and enumerate all possible distances from o to the potential near neighbor we are looking for. For each possible distance, we query the corresponding ProcessSphere children after projecting q on the sphere with a tentative near neighbor using, Project.

4.3 Setting parameters

We complete the description of the data structure by setting the remaining parameters. 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 specify how to set $\eta_u, \eta_q > 0$ and T for each pseudo-random remainder. The idea will be to try to replicate the parameter settings of Section 3.3.3 corresponding to the random instance. The important parameter will be r^*, which acts as the "effective" r_2. In the case that $r_2 \geq \sqrt{2}R$, then we have more flexibility than in the random setting, so we let $r^* = r_2$. In the case that $r_2 < \sqrt{2}R$, then we let $r^* = \sqrt{2}R$. In particular, we let

$$T = \frac{100}{G(r_1/R, \eta_u, \eta_q)}$$

in order to achieve a constant probability of success. Then we let η_u and η_q such that

- $F(\eta_u)/G(r_1/R, \eta_u, \eta_q) \leq n^{(\rho_u + o(1))/K}$
- $F(\eta_q)/G(r_1/R, \eta_u, \eta_q) \leq n^{(\rho_q + o(1))/K}$
- $G(r^*/R, \eta_u, \eta_q)/G(r_1/R, \eta_u, \eta_q) \leq n^{(\rho_q - 1 + o(1))/K}$

which correspond to the parameter settings achieving the tradeoff of Section 3.3.3.

A crucial relation between the 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.

We collect some basic facts from the data structure which will be useful for the analysis. These facts follow trivially from the pseudocode in Figure 5.

- Process is called once at the beginning and has one child corresponding to one call to ProcessSphere. In the analysis, we will disregard this node. Process does not take up any significant space or time. Thus, we refer to the root of the tree as the first call to ProcessSphere.
- The children to ProcessSphere may contain at most $\frac{1}{\tau}$ many calls to ProcessBall, corresponding to cluster nodes, and T calls to ProcessSphere. Each ProcessBall call of ProcessSphere handles a disjoint subset of the dataset. Points can be replicated in the pseudo-random remainder, when a point lies in the intersection of two or more caps.
- ProcessBall has many children, all of which are ProcessSphere which do not increment l. Each of these children corresponds to a call on a specific annulus of width δr_1 around the center as well as possible distance for a query. For each annulus, there are at most $\frac{2}{\delta} + 4$ notable distances; after rounding by δr_1, a valid query can be at most $r_1 + 2\delta r_1$ away from a particular annulus in both directions, thus, each point gets duplicated at most $\frac{2}{\delta} + 4$ many times.
- For each possible point $p \in P$, we may consider the subtree of nodes which process that particular point. We make the distinction between two kinds of calls to ProcessSphere: calls where p lies in a dense cluster, and calls where p lies in a pseudo-random remainder. If p lies in a dense cluster, l will not be incremented; if p lies in the pseudo-random remainder, l will

be incremented. The point p may be processed by various rounds of calls to PROCESSBALL and PROCESSSPHERE without incrementing l; however, there will be a moment when p is not in a dense cluster and will be part of the pseudo-random remainder. In that setting, p will be processed by a call to PROCESSSPHERE which increments l.

4.4 Analysis

≪need to make another pass over this... Things to note:

- R changed to $\widetilde{O}(\log^2 n)$.

–Erik≫ EW

Lemma 4.2. *The following invariants hold.*

- *At any moment one has $\frac{r_2}{r_1} \geq c \cdot (1 - o(1))$ and $r_2 \leq c \cdot (1 + o(1))$.*
- *At any moment the number of calls to* PROCESSBALL *in the recursion stack is at most $\widetilde{O}(\log \log n)$.*

Proof. Our proof will proceed by keeping track of two quantities, γ and ξ as the algorithm proceeds down the tree. We will be able to analyze how these values change as the algorithm executes the subroutines PROCESSSPHERE and PROCESSBALL. We will then combine these two measures to get a potential function which always increases by a multiplicative factor of $(1 + \Omega(\varepsilon^2))$. By giving overall bounds on γ and ξ, we will deduce an upper bound on the depth of the tree. For any particular node of the tree v (where v may correspond to a call to PROCESSSPHERE or PROCESSBALL), we let

$$\gamma_v = \frac{r_2^2}{r_1^2} \qquad \text{and} \qquad \xi_v = \frac{r_2^2}{R^2}$$

where the values of r_1, r_2, and R are given by the procedure call at v. We will often refer to γ_v and ξ_v as γ and ξ, respectively, when there is no confusion. Additionally, we will often refer to how γ and ξ changes; in particular, if $\tilde{v}$ is a child of v, then we let $\tilde{\gamma}$ and $\tilde{\xi}$ be the values of $\gamma_{\tilde{v}}$ and $\xi_{\tilde{v}}$.

Claim 4.3. *Initially, $\gamma = c^2$, and it only changes when* PROCESSBALL *calls* PROCESSSPHERE. *Letting $\Delta R = \delta r_1 |i - j|$, there are two cases:*

- *If $0 \leq \dfrac{\Delta R^2}{r_1^2} \leq \frac{12\delta}{\lambda}$, then $\dfrac{\tilde{\gamma}}{\gamma} \geq 1 - 24\delta$.*
- *If $\dfrac{\Delta R^2}{r_2^2} \geq \frac{12\delta}{\lambda}$, then $\dfrac{\tilde{\gamma}}{\gamma} \geq 1 + \dfrac{\Delta R^2}{r_1^2} \cdot \frac{\lambda}{2} - 6\delta$.*

for $\lambda = 1 - (\frac{2}{c+1})^2 > 0$.

```
function ProcessSphere($P$, $r_1$, $r_2$, $o$, $R$, $l$)
    if $l = K$ then                                                        ▷ base case 1
        store $P$ explicitly
        return
    if $r_2 \geq 2R$ then                                                  ▷ base case 2
        store any point from $P$
        return
    $r^* \leftarrow r_2$
    if $\left(1 - \alpha\left(\frac{r_1}{R}\right)\alpha\left(\frac{r_2}{R}\right)\right)\sqrt{\rho_q} + \left(\alpha\left(\frac{r_1}{R}\right) - \alpha\left(\frac{r_2}{R}\right)\right)\sqrt{\rho_u} < \beta\left(\frac{r_1}{R}\right)\beta\left(\frac{r_2}{R}\right)$ then    ▷ base case 3
        $m \leftarrow |P|$
        $\widehat{R} \leftarrow (\sqrt{2} - \varepsilon)R$
        while $\exists\, x \in \partial B(o, R) : |B(x, \widehat{R}) \cap P| \geq \tau m$ do    ▷ remove dense clusters
            $B(\widetilde{o}, \widetilde{R}) \leftarrow$ the SEB for $P \cap B(x, \widehat{R})$
            ProcessBall($P \cap B(x, \widehat{R})$, $r_1$, $r_2$, $\widetilde{o}$, $\widetilde{R}$, $l$)
            $P \leftarrow P \setminus B(x, \widehat{R})$
        $r^* \leftarrow \sqrt{2}R$
    choose $\eta_u$ and $\eta_q$ such that:                               ▷ data independent portion
      • $F(\eta_u)/G(r_1/R, \eta_u, \eta_q) \leq n^{(\rho_u + o(1))/K}$;
      • $F(\eta_q)/G(r_1/R, \eta_u, \eta_q) \leq n^{(\rho_q + o(1))/K}$;
      • $G(r^*/R, \eta_u, \eta_q)/G(r_1/R, \eta_u, \eta_q) \leq n^{(\rho_q - 1 + o(1))/K}$.
    $T \leftarrow 100/G(r_1/R, \eta_u, \eta_q)$
    for $i \leftarrow 1 \ldots T$ do
        sample $z \sim N(0,1)^d$
        $P' \leftarrow \{p \in P \mid \langle z, p\rangle \geq \eta_u R\}$
        if $P' \neq \emptyset$ then
            ProcessSphere($P'$, $r_1$, $r_2$, $o$, $R$, $l + 1$)
function ProcessBall($P$, $r_1$, $r_2$, $o$, $R$, $l$)
    if $r_1 + 2R \leq r_2$ then                                            ▷ trivial instance of ProcessBall
        store any point from $P$
        return
    $P \leftarrow \{o + \delta r_1 \lceil \frac{\|p-o\|}{\delta r_1} \rceil \cdot \frac{p-o}{\|p-o\|} \mid p \in P\}$
    for $i \leftarrow 1 \ldots \lceil \frac{R}{\delta r_1} \rceil$ do
        $\widetilde{P} \leftarrow \{p \in P \colon \|p - o\| = \delta i r_1\}$
        if $\widetilde{P} \neq \emptyset$ then
            for $j \leftarrow 1 \ldots \lceil \frac{R + r_1 + 2\delta r_1}{\delta r_1} \rceil$ do
                if $\delta r_1 |i - j| \leq r_1 + 2\delta r_1$ then
                    $\widetilde{r_1} \leftarrow$ Project($\delta i r_1$, $\delta j r_1$, $r_1 + 2\delta r_1$)    ▷ computing $\widetilde{r_1}$ and $\widetilde{r_2}$ for projected instance
                    $\widetilde{r_2} \leftarrow$ Project($\delta i r_1$, $\delta j r_1$, $r_2 - 2\delta r_1$)
                    ProcessSphere($\widetilde{P}$, $\widetilde{r_1}$, $\widetilde{r_2}$, $o$, $\delta i r_1$, $l$)
function Project($R_1$, $R_2$, $r$)
    return $\sqrt{R_1(r^2 - (R_1 - R_2)^2)/R_2}$
```

Figure 5: Pseudocode of the data structure (SEB stands for *smallest enclosing ball*)

Note that initially, $\gamma = c^2$ since $r_1 = 1$ and $r_2 = c$. Now, $\tilde{\gamma}$ changes when ProcessBall(P, r_1, r_2, o, R, l) calls ProcessSphere$(\widetilde{P}, \tilde{r_1}, \tilde{r_2}, o, \delta i r_1, l)$ in line 36 of Figure 5. When this occurs:

$$\begin{aligned}
\frac{\tilde{\gamma}}{\gamma} &= \frac{\textsc{Project}(\delta i r_1, \delta j r_1, r_2 - 2\delta r_1)^2 / \textsc{Project}(\delta i r_1, \delta j r_1, r_1 + 2\delta r_1)^2}{r_2^2 / r_1^2} \\
&= \frac{(r_2 - 2\delta r_1)^2 - \Delta R^2}{r_2^2} \cdot \frac{r_1^2}{(r_1 + 2\delta r_1)^2 - \Delta R^2} \\
&= \frac{(1 - \frac{2\delta r_1}{r_2})^2 - \frac{\Delta R^2}{r_2^2}}{(1 + 2\delta)^2 - \frac{\Delta R^2}{r_1^2}} \geq 1 + \frac{\Delta R^2 (\frac{1}{r_1^2} - \frac{1}{r_2^2}) - 12\delta}{(1 + 2\delta)^2 - \frac{\Delta R^2}{r_1^2}} \\
&= 1 + \frac{\frac{\Delta R^2}{r_1^2} \cdot \lambda - 12\delta}{(1 + 2\delta)^2 - \frac{\Delta R^2}{r_1^2}}
\end{aligned}$$

assuming that $r_1(\frac{c+1}{2}) \leq r_2$ (we will actually show the much tighter bound of $r_1 \cdot c \cdot (1 - o(1)) \leq r_2$ toward the end of the proof) and setting $\lambda = 1 - (\frac{2}{c+1})^2$, where $\lambda \in (0, 1)$. Note that the denominator in the expression above is non-negative since $\frac{\Delta R^2}{r_1^2} \leq (1 + 2\delta)^2$. Now, consider two cases:

- Case 1: $0 \leq \frac{\Delta R^2}{r_1^2} \leq \frac{12\delta}{\lambda}$. In this case, we have:

  $$\begin{aligned}
  \frac{\tilde{\gamma}}{\gamma} &\geq 1 + \frac{\frac{\Delta R^2}{r_1^2} \cdot \lambda - 12\delta}{(1 + 2\delta)^2 - \frac{\Delta R^2}{r_1^2}} \\
  &\geq 1 - \frac{12\delta}{(1 + 2\delta)^2 - \frac{\Delta R^2}{r_1^2}} \geq 1 - \frac{12\delta}{(1/2)}.
  \end{aligned}$$

  Thus, the multiplicative decrease is at most $(1 - 24\delta)$ since $\delta = o(1)$.

- Case 2: $\frac{\Delta R^2}{r_1^2} \geq \frac{12\delta}{\lambda}$. In this case:

  $$\begin{aligned}
  \frac{\tilde{\gamma}}{\gamma} &\geq 1 + \frac{\frac{\Delta R^2}{r_1^2} \cdot \lambda - 12\delta}{(1 + 2\delta)^2 - \frac{\Delta R^2}{r_1^2}} \\
  &\geq 1 + \frac{\frac{\Delta R^2}{r_1^2} \cdot \lambda - 12\delta}{2} \\
  &= 1 + \frac{\Delta R^2}{r_1^2} \cdot \frac{\lambda}{2} - 6\delta.
  \end{aligned}$$

Claim 4.4. *Initially, $\xi \geq \widetilde{\Omega}\left(\frac{c^2}{\log^4 n}\right)$. ξ changes only when* ProcessBall *calls* ProcressSphere*, or vice-versa. When* ProcessBall *calls* ProcessSphere*, and some later* ProcessSphere *calls*

ProcessBall, *letting* $\Delta R = \delta r_1|i-j|$:

$$\frac{\tilde{\xi}}{\xi} \geq \Big(1+\Omega(\varepsilon^2)\Big)\left((1-2\delta)^2 - \frac{\Delta R^2}{r_1^2}(1-\lambda)\right)\left(\frac{1}{1+\frac{\Delta R}{R}}\right),$$

for $\lambda = 1-(\frac{2}{c+1})^2 > 0$.

The relevant procedure calls in Claim 4.4 are:

1. ProcessBall(P, r_1, r_2, o, R, l) calls ProcessSphere$(\widetilde{P}, \widetilde{r_1}, \widetilde{r_2}, o, \delta i r_1, l)$ in line 36 of Figure 5.

2. After possibly some string of calls to ProcessSphere, ProcessSphere$(P', \widetilde{r_1}, \widetilde{r_2}, o, \delta i r_1, l')$ calls ProcessBall$(P' \cap B(\widetilde{o}, \widetilde{R}), \widetilde{r_1}, \widetilde{r_2}, \widetilde{o}, l')$ in line 14 of Figure 5.

Since both calls to ProcessBall identified above have no ProcessBall calls in their path in the tree, we have the following relationships between the parameters:

- $\widetilde{r_1} = \textsc{Process}(\delta i r_1, \delta j r_1, r_1 + 2\delta)$,
- $\widetilde{r_2} = \textsc{Process}(\delta i r_1, \delta j r_1, r_2 - 2\delta)$,
- $\widetilde{R} \leq (1-\Omega(\varepsilon^2)) \cdot \delta i r_1$,

Using these setting of parameters,

$$\begin{aligned}\frac{\tilde{\xi}}{\xi} &= \Big(1+\Omega(\varepsilon^2)\Big)^2 \cdot \frac{\textsc{Process}(\delta i r_1, \delta j r_1, r_2 - 2\delta r_1)^2/\delta^2 i^2 r_1^2}{r_2^2/R^2} \\ &= \Big(1+\Omega(\varepsilon^2)\Big)^2 \left(\left(1-\frac{2\delta r_1}{r_2}\right)^2 - \frac{\Delta R^2}{r_2^2}\right) \cdot \frac{R^2}{\delta^2 i j r_1^2} \\ &\geq \Big(1+\Omega(\varepsilon^2)\Big)\left((1-2\delta)^2 - \frac{\Delta R^2}{r_1^2}\cdot(1-\lambda)\right)\left(\frac{1}{1+\frac{\Delta R}{R}}\right),\end{aligned}$$

where in the last step, we used the fact that $r_1(\frac{c+1}{2}) \leq r_2$, and that $\delta i r_1 \leq R$ and $\delta j r_1 \leq R + \Delta R$. Note that the lower bound is always positive since $\frac{\Delta R^2}{r_1^2} \leq 1 + 2\delta$, $\delta = o(1)$, and $\lambda \in (0,1)$ is some constant.

We consider the following potential function:

$$\Phi = \gamma^M \cdot \xi,$$

and we set $M = \frac{800}{\sqrt{24 \cdot \lambda \cdot \delta}}$.

Claim 4.5. *In every iteration of* ProcessBall *calling* ProcessSphere *which at some point calls* ProcessBall *again,* Φ *increases by a multiplicative factor of* $1+\Omega(\varepsilon^2)$.

We simply compute the multiplicative change in Φ by using Claim 4.3 and Claim 4.4. We will first apply the first case of Claim 4.3, where $0 < \frac{\Delta R^2}{r_1^2} \leq \frac{24\delta}{\lambda}$.

$$\begin{aligned}\frac{\widetilde{\Phi}}{\Phi} &\geq (1-24\delta)^M \cdot \left(1+\Omega(\varepsilon^2)\right) \cdot \left((1-2\delta)^2 - \frac{\Delta R^2}{r_1^2}(1-\lambda)\right) \cdot \left(\frac{1}{1+\frac{\Delta R}{R}}\right) \\ &\geq \left(1+\Omega(\varepsilon^2)\right) \cdot \left(1 - 24\delta M - 4\delta - \frac{\Delta R^2}{r_1^2} - \frac{\Delta R}{R}\right) \\ &\geq \left(1+\Omega(\varepsilon^2)\right) \cdot \left(1 - 24\delta M - 4\delta - \frac{24\delta}{\lambda} - \frac{\sqrt{96 \cdot \delta}}{\sqrt{\lambda}}\right),\end{aligned}$$

where the third inequality, we used $\frac{\Delta R^2}{r_1^2} \leq \frac{24\delta}{\lambda}$ and $r_1 \leq r_2 \leq 2R$ by line 5 of Figure 5. Finally, we note that $\varepsilon^2 \gg 24\delta M - 4\delta - \frac{24\delta}{\lambda} - \frac{\sqrt{96\delta}}{\sqrt{\lambda}}$, so in this case,

$$\frac{\widetilde{\Phi}}{\Phi} \geq \left(1+\Omega(\varepsilon^2)\right).$$

We now proceed to the second case, when $\frac{\Delta R^2}{r_2^2} > \frac{24\delta}{\lambda}$. Using case 2 of Claim 4.3, we have

$$\frac{\widetilde{\Phi}}{\Phi} \geq \left(1 + \frac{\Delta R^2}{r_1^2} \cdot \frac{\lambda}{4}\right)^M \cdot \left(1+\Omega(\varepsilon^2)\right) \cdot \left((1-2\delta)^2 - \frac{\Delta R^2}{r_1^2}(1-\lambda)\right) \cdot \left(\frac{1}{1+\frac{\Delta R}{R}}\right). \tag{15}$$

We claim the above expression is at least $1+\Omega(\varepsilon^2)$. This follows from three observations:

- $\frac{\Delta R^2}{r_1^2} \geq \frac{24\delta}{\lambda}$ implies that $\frac{\sqrt{\lambda}}{\sqrt{24 \cdot \delta}} \geq \frac{r_1}{\Delta R}$.
- Since $r_1 \leq r_2 \leq 2R$ (by line 5 of Figure 5), $\frac{2}{r_1} \geq \frac{1}{R}$, so $\frac{\Delta R^2}{r_1^2} \cdot \frac{2\sqrt{\lambda}}{\sqrt{24 \cdot \delta}} \geq \frac{2\Delta R}{r_1} \geq \frac{\Delta R}{R}$.
- Thus, if $M = \frac{800}{\sqrt{24 \cdot \lambda \cdot \delta}}$,

$$\frac{\Delta R^2}{r_1^2} \cdot \frac{\lambda}{4} \cdot M \geq 100 \cdot \frac{\Delta R^2}{r_1^2} \cdot \frac{2 \cdot \sqrt{\lambda}}{\sqrt{24 \cdot \delta}} \geq 100 \cdot \frac{\Delta R}{R}.$$

Furthermore, $\frac{\Delta R^2}{r_1^2} \cdot \frac{\lambda}{4} \cdot M \gg 4\delta + \frac{\Delta R^2}{r_1^2}$, which means that in this case,

$$\frac{\widetilde{\Phi}}{\Phi} \geq \left(1+\Omega(\varepsilon^2)\right).$$

Having lower bounded the multiplicative increase in Φ, we note that initially,

$$\Phi_0 = \widetilde{\Omega}\left(\frac{c^{2M+2}}{\log^4 n}\right).$$

Claim 4.6. *At all moments in the algorithm $\gamma \leq \log n$.*

Note that before reaching $\frac{r_2^2}{r_1^2} \geq \log n$, line 9 of Figure 5 will always evaluate to false, and the algorithm will continue to proceed in a data-independent fashion without further changes to r_1, r_2 or R. Another way to see this is that when $\frac{r_2}{r_1} \geq \sqrt{\log n}$, then the curve in Figure 2 corresponding to [AI06] will give a data structure with runtime $n^{o(1)}$ and space $n^{1+o(1)}$.

Additionally, line 5 of Figure 5 enforces that all moments in the algorithm, $\xi \leq 4$. Thus, at all moments,

$$\Phi \leq O(\log^M n).$$

Thus, the number of times that ProcessBall appears in the stack is $\widetilde{O}(\log \log n)$. We will now show the final part of the proof which we stated earlier:

Claim 4.7. *For the first $\widetilde{O}\left(\log \log n\right)$ many iterations,*

$$r_1 \cdot c \cdot (1 - o(1)) \leq r_2.$$

Note that showing this will imply $\eta \geq \left(\frac{c+1}{2}\right)^2$. From Claim 4.3, $\eta \geq c^2 \left(1 - 24\delta\right)^N$, where $N = \widetilde{O}\left(\log \log n\right)$, which is in fact, always at most $c^2(1-o(1))$. In order to show that $r_2 \leq c\cdot(1+o(1))$, note that r_2 only increases by a multiplicative factor of $(1 + 2\delta)$ in each call of ProcessBall. This finishes the proof of all invariants. □

Lemma 4.8. *During the algorithm we can always be able to choose η_u and η_q such that:*

- $F(\eta_u)/G(r_1/R, \eta_u, \eta_q) \leq n^{(\rho_u + o(1))/K}$;
- $F(\eta_q)/G(r_1/R, \eta_u, \eta_q) \leq n^{(\rho_q + o(1))/K}$;
- $G(r^*/R, \eta_u, \eta_q)/G(r_1/R, \eta_u, \eta_q) \leq n^{(\rho_q - 1 + o(1))/K}$.

Proof. We will focus on the the part of ProcessSphere where we find settings for η_u and η_q. There are two important cases:

- $r^* = r_2$. This happens when the third "if" statement evaluates to false. In other words, we have that

  $$\left(1 - \alpha\left(\frac{r_1}{R}\right)\alpha\left(\frac{r_2}{R}\right)\right)\sqrt{\rho_q} + \left(\alpha\left(\frac{r_1}{R}\right) - \alpha\left(\frac{r_2}{R}\right)\right)\sqrt{\rho_u} \geq \beta\left(\frac{r_1}{R}\right)\beta\left(\frac{r_2}{R}\right). \qquad (16)$$

  Since in a call to ProcessSphere, all points are on the surface of a sphere of radius R, the expression corresponds to the expression from Theorem 3.3. Thus, as described in Section 3.3.3, we can set η_u and η_q to satisfy the three conditions.

- $r^* = \sqrt{2}R$. This happens when the third "if" statement evaluates to true. We have by Lemma 4.2 $\frac{r_2}{r_1} \geq c - o(1)$. Since (16) does not hold, thus $r_2 < \sqrt{2}R$. Hence, $r_1 \leq \frac{\sqrt{2}R}{c} - o(1)$. If this is the case since $r_1 \leq r^*/c - o(1)$, we are instantiating parameters as in Subsection 3.3.3 where $r = \frac{r_1}{R}$ and $cr = \frac{r^*}{R}$.

□

Lemma 4.9. *The probability of success of the data structure is at least* 0.9*.*

Proof. In all the cases, except for the handling of the pseudo-random remainder, the data structure is deterministic. Therefore, the proof follows in exactly the same way as Lemma 3.6. In this case, we also have at each step that $T = \frac{100}{G(r_1/R, \eta_u, \eta_q)}$, and the induction is over the number of times we handle the pseudo-random remainder. □

Lemma 4.10. *The total space the data structure occupies is at most* $n^{1+\rho_u+o(1)}$ *in expectation.*

Proof. We will prove that the total number of explicitly stored points (when $l = K$) is at most $n^{1+\rho_u+o(1)}$. We will count the contribution from each point separately, and use linearity of expectation to sum up the contributions. In particular, for a point $p \in P_0$, we want to count the number of lists where p appears in the data structure. Each root to leaf path of the tree has at most K calls to ProcessSphere which increment l, and at most $\widetilde{O}(\log \log n)$ calls to ProcessBall, and thus $\widetilde{O}(\log \log n)$ calls to ProcessSphere which do not increment l. Thus, once we count the number of lists, we may multiply by $K + \widetilde{O}(\log \log n) = n^{o(1)}$ to count the size of the whole tree.

For each point, we will consider the subtree of the data structure where the point was processed. In particular, we may consider the tree corresponding to calls to ProcessSphere and ProcessBall which process p. As discussed briefly in Section 4.3, we distinguish between calls to ProcessSphere which contain p in a dense cluster, and calls to ProcessSphere which contain p in the pseudo-random remainder. We increment l only when p lies in the pseudo-random remainder.

Claim 4.11. *It suffices to consider the data structure where each node is a function call to* ProcessSphere *which increments l, i.e., when p lies in the pseudo-random remainder, since the total amount of duplication of points corresponding to other nodes is* $n^{o(1)}$*.*

We will account for the duplication of points in calls to ProcessBall and calls toProcessSphere which do not increment l. Consider the first node v in a path from the root which does not increment l, this corresponds to a call to ProcessSphere which had p in some dense cluster. Consider the subtree consisting of descendants of v where the leaves correspond to the first occurrence of ProcessSphere which increments l. We claim that every internal node of the tree corresponds to alternating calls to ProcessBall and ProcessSphere which do not increment l. Note that calls to ProcessSphere which do not increment l never replicate p. Each call to ProcessBall replicates p in $b := \frac{2r_1(1+2\delta)}{\delta}$ many times. Since $r_1 \leq r_2 \leq c + o(1)$ by Lemma 4.2, $b = O(\delta^{-1})$. We may consider contracting the tree and at edge, multiplying by the number of times we encounter ProcessBall.

Note that p lies in a dense cluster if and only if it does not lie in the pseudo-random remainder. Thus, our contracted tree looks like a tree of K levels, each corresponding to a call to ProcessSphere which contained p in the pseudo-random remainder.

The number of children of some nodes may be different; however, the number of times ProcessBall is called in each branch of computation is $U := \tilde{O}(\log \log n)$, the total amount of duplication of points due to ProcessBall is at most $b^U = n^{o(1)}$. Now, the subtree of nodes processing p contains K levels with each T children, exactly like the data structure for Section 3.

Claim 4.12. *A node v corresponding to* ProcessSphere(P, r_1, r_2, o, R, l) *has in expectation, p appearing in $n^{((K-l)\rho_u + o(1))/K}$ many lists in the subtree of v.*

The proof is an induction over the value of l in a particular node. For our base case, consider some node v corresponding to a function call of ProcessSphere which is a leaf, so $l = K$, in this case, each point is only stored at most once, so the claim holds.

Suppose for the inductive assumption the claim holds for some l, then for a particular node at level $l - 1$, consider the point when p was part of the pseudo-random remainder. In this case, p is duplicated in

$$T \cdot F(\eta_u) = \frac{100 \cdot F(\eta_u)}{G(r_1/R, \eta_u, \eta_q)} \leq n^{(\rho_u + o(1))/K}$$

many children, and in each child, the point appears $n^{((K-l)\rho_u + o(1))/K}$ many times. Therefore, in a node v, p appears in $n^{((K-l+1)\rho_u + o(1))/K}$ many list in its subtree. Letting $l = 0$ for the root gives the desired outcome. □

Lemma 4.13. *The expected query time is at most $n^{\rho_q + o(1)}$.*

Proof. We need to bound the expected number of nodes we traverse as well as the number of points we enumerate for nodes with $l = K$.

We first bound the number of nodes we traverse. Let $A(u, l)$ be an upper bound on the expected number of visited nodes when we start in a ProcessSphere node such that there are u ProcessBall nodes in the stack and l non-cluster nodes. By Lemma 4.2,

$$u \leq U := \widetilde{O}(\log \log n),$$

and from the description of the algorithm, we have $l \leq K$. We will prove $A(0, 0) \leq n^{\rho_q + o(1)}$, which corresponds to the expected number of nodes we touch starting from the root.

We claim

$$A(u, l) \leq \exp(\log^{2/3 + o(1)} n) \cdot A(u + 1, l) + n^{(\rho_q + o(1))/K} \cdot A(u, l + 1). \tag{17}$$

There are at most $1/\tau = \exp(\log^{2/3} n)$ cluster nodes, and in each node, we recurse on $\frac{2r_1(1+2\delta)}{\delta} = \exp(\log^{o(1)} n)$ possible annuli with calls to ProcessSphere nodes where u increased by 1 and l remains the same. On the other hand, there are

$$T \cdot F(\eta_q) = \frac{100 \cdot F(\eta_q)}{G(r_1/R, \eta_u, \eta_q)} \leq n^{(\rho_q + o(1))/K}$$

caps, where the query falls, in expectation. Each calls ProcessSphere where u remains the same and l increased by 1.

Solving (17):

$$A(0,0) \leq \binom{U+K}{K} \exp(U \cdot \log^{2/3+o(1)} n) \cdot n^{\rho_q+o(1)} \leq n^{\rho_q+o(1)}.$$

We now give an upper bound on the number of points the query algorithm will test at level K. Let $B(u,l)$ be an upper bound on the expected fraction of the dataset in the current node that the query algorithm will eventually test at level K (where we count multiplicities). u and l have the same meaning as discussed above.

We claim

$$B(u,l) \leq \frac{1}{\tau} \cdot B(u+1,l) + n^{(\rho_q-1+o(1))/K} \cdot B(u,l+1)$$

The first term comes from recursing down dense clusters. The second term is a bit more subtle. In particular, suppose $r_2 = r^*$, then the expected fraction of points is

$$\begin{aligned} T \cdot G(r_2/R, \eta_u, \eta_q) \cdot B(u,l+1) &= \frac{100 \cdot G(r_2/R, \eta_u, \eta_q) \cdot B(u,l+1)}{G(r_1/R, \eta_u, \eta_q)} \\ &\leq n^{(\rho_q-1+o(1))/K} \cdot B(u,l+1) \end{aligned}$$

by the setting of η_u and η_q. On the other hand, there is the other case when $r^* = \sqrt{2}R$, which occurs after having removed some clusters. In that case, consider a particular cap containing the points $\widetilde{P}_i$. For points with distance to the query at most $(\sqrt{2}-\varepsilon)R$, there are at most a τn of them. For the far points, $\widetilde{P}_i$ a $G(\sqrt{2}-\varepsilon, \eta_u, \eta_q)$ fraction of the points in expectation.

$$\begin{aligned} T \cdot F(\eta_q) \cdot \Big(\tau + G(\sqrt{2}-\varepsilon, \eta_u, \eta_q)\Big) \cdot B(u,l+1) &= \frac{100 \cdot F(\eta_q) \cdot \Big(\tau + G(\sqrt{2}-\varepsilon, \eta_u, \eta_q)\Big) \cdot B(u,l+1)}{G(r_1/R, \eta_u, \eta_q)} \\ &\leq \frac{100 \cdot F(\eta_q) \cdot G(\sqrt{2}, \eta_u, \eta_q) \cdot B(u,l+1)}{G(r_1/R, \eta_u, \eta_q)} \\ &\leq n^{(\rho_q-1+o(1))/K} \cdot B(u,l+1) \end{aligned}$$

Where we used that $\tau \ll G(\sqrt{2}, \eta_u, \eta_q)$ and $G(\sqrt{2}-\varepsilon, \eta_u, \eta_q) \leq G(\sqrt{2}, \eta_u, \eta_q) \cdot n^{o(1)/K}$ (since $\varepsilon = o(1)$), and that $r^* = \sqrt{2}R$. Unraveling the recursion, we note that $u \leq U$ and $l \leq K \sim \sqrt{\ln n}$. Additionally, we have that $B(u,K) \leq 1$, since we do not store duplicates in the last level. Therefore,

$$B(0,0) \leq \binom{U+K}{U} \left(\frac{1}{\tau}\right)^U \cdot \left(n^{(\rho_q-1+o(1))/K}\right)^K = n^{\rho_q-1+o(1)}.$$

□

5 Lower bounds: preliminaries

We introduce a few techniques and concepts to be used primarily for our lower bounds. We start by defining the approximate nearest neighbor search problem.

Definition 5.1. *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 $P \subset \{-1, 1\}^d$ 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 $1 - \delta$.*

5.1 Graphical Neighbor Search and robust expansion

We introduce a few definitions from [PTW10] to setup the lower bounds for the ANN.

Definition 5.2 ([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 $U = V = \{-1, 1\}^d$, and connecting two points $u \in U, v \in V$ iff their Hamming distance most r (see details in [PTW10]). We will also ensure q is not closer than cr to other points apart from the near neighbor.

[PTW10] showed lower bounds for ANN are intimately tied to the following quantity of a metric space.

Definition 5.3 (Robust Expansion [PTW10])**.** *Consider a GNS graph $G = (U, V, E)$, and fix a distribution e on $E \subset U \times V$, where μ is the marginal distribution on U and η is the marginal distribution on V. For $\delta, \gamma \in (0, 1]$, the* robust expansion $\Phi_r(\delta, \gamma)$ *is:*

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

5.2 Locally-decodable codes (LDC)

Our 2-probe lower bounds uses results on Locally-Decodable Codes (LDCs). We present the standard definitions and results on LDCs below, although in Section 8, we will use a weaker definition of LDCs for our 2-query lower bound.

Definition 5.4. *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 5.5 (Theorem 4 from [KdW04]). *If $C : \{0,1\}^n \to \{0,1\}^m$ is a $(2, \delta, \varepsilon)$-LDC, then*

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

6 Lower bounds: one-probe data structures

6.1 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 corresponding to the random instance of Section 2.

Definition 6.1. *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 $i \in [d]$ independently; with probability σ, we set $y_i = x_i$, and 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

$$T_\sigma f(x) = \mathop{\mathbb{E}}_{y \sim N_\sigma(x)}[f(y)] \tag{18}$$

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.

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 and d, $d(x,y) \leq \frac{d}{2c}(1+o(1))$ 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 6.2 (Robust expansion). *Consider 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 $m \geq 1$,

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

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 6.3 (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*

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

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

Theorem 6.4 (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*

$$\|T_\sigma f\|_q \leq \|f\|_p.$$

Theorem 6.5 (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*

$$\langle f, g \rangle \leq \|f\|_s \|g\|_t.$$

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 applied to f will measure the neighborhood of set A. We 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 6.6. *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, apply the Bonami-Beckner Inequality to each part.

$$\langle T_\sigma f, f \rangle = \langle T_{\sqrt{\sigma}} f, T_{\sqrt{\sigma}} g \rangle \leq \|T_{\sqrt{\sigma}} f\|_s \|T_{\sqrt{\sigma}} g\|_t.$$

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

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

In addition,

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

We finally apply the Bonami-Beckner Inequality to both norms to obtain

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

□

We are now ready to prove Lemma 6.2.

Proof of Lemma 6.2. We use Lemma 6.6 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 $\chi_A : \{-1,1\}^d \to \{0,1\}$ and $\chi_B : \{-1,1\}^d \to \{0,1\}$ as the indicator functions for A and B. Then,

$$\gamma = \Pr_{x \sim \mu, y \sim N_\sigma(x)}[x \in B \mid y \in A] = \frac{1}{a}\langle T_\sigma \chi_A, \chi_B \rangle \leq a^{\frac{1}{p}-1} b^{\frac{1}{q}}. \tag{19}$$

Therefore, $\gamma^q a^{q-\frac{q}{p}} \leq b$. Let A and B be the minimizers of $\frac{b}{a}$ satisfying (19) and $a \leq \frac{1}{m}$.

$$\Phi_r\left(\frac{1}{m}, \gamma\right) = \frac{b}{a} \geq \gamma^q a^{q-\frac{q}{p}-1} \geq \gamma^q m^{1+\frac{q}{p}-q}.$$

□

6.2 Lower bounds for one-probe data structures

In this section, we prove Theorem 1.3. Our proof relies on the main result of [PTW10] for the GNS problem:

Theorem 6.7 (Theorem 1.5 [PTW10])**.** *There exists an absolute constant γ such that the following holds. Any randomized cell-probe data structure making t probes and using m cells of w bits for a weakly independent instance of GNS which is correct with probability greater than $\frac{1}{2}$ must satisfy*

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

Proof of Theorem 1.3. The lower bound follows from a direct application of Lemma 6.2 to Theorem 6.7. Setting $t = 1$ in Theorem 6.7, we obtain

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

for some $p, q \in [1, \infty)$ and $(p-1)(q-1) = \sigma^2$. Rearranging the inequality and letting $p = 1 + \frac{\log\log n}{\log n}$, and $q = 1 + \sigma^2 \frac{\log n}{\log\log n}$, we obtain

$$m \geq \frac{\gamma^{\frac{p}{p-1}} n^{\frac{p}{pq-q}}}{w^{\frac{p}{pq-q}}} \geq n^{\frac{1}{\sigma^2}-o(1)}.$$

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

Corollary 6.8. *Any 1 cell probe data structure with cell size $n^{o(1)}$ 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. □

7 Lower bounds: list-of-points data structures

In this section, we prove Theorem 1.6, i.e., a tight lower bound against data structure that fall inside the "list-of-points" model, as defined in Def. 1.5.

Theorem 7.1 (Restatement of Theorem 1.6)**.** *Let D be a list-of-points data structure which solves (c, r)-ANN for n points in the d-dimensional Hamming space with $d = \omega(\log n)$. Suppose D is specified by a sequence of m sets $\{A_i\}_{i=1}^m$ and a procedure for outputting a subset $I(q) \subset [m]$ using expected space $s = n^{1+\rho_u}$, and expected query time $n^{\rho_q - o(1)}$ with success probability $\frac{2}{3}$. Then*

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

We will prove the lower bound by giving a lower bound on list-of-points data structures which solve the random instance for the Hamming space defined in Section 2. The dataset consists of n points $\{u_i\}_{i=1}^n$ where each $u_i \sim V$ drawn uniformly at random, and a query v is drawn from the neighborhood of a random dataset point. Thus, we may assume D is a deterministic data structure.

Fix a data structure D, where $A_i \subset V$ specifies which dataset points are placed in L_i. Additionally, we may define $B_i \subset V$ which specifies which query points scan L_i, i.e., $B_i = \{v \in V \mid i \in I(v)\}$. Suppose we sample a random dataset point $u \sim V$ and then a random query point v from the neighborhood of u. Let

$$\gamma_i = \Pr[v \in B_i \mid u \in A_i]$$

represent the probability that query v scans the list L_i, conditioned on u being in L_i. Additionally, we write $s_i = \mu(A_i)$ as the normalized size of A. The query time for D is given by the following expression:

$$\begin{aligned} T &= \sum_{i=1}^{m} \chi_{B_i}(v) \left(1 + \sum_{j=1}^{n} \chi_{A_i}(u_j)\right) \\ \mathbb{E}[T] &= \sum_{i=1}^{m} \mu(B_i) + \sum_{i=1}^{m} \gamma_i \mu(A_i) + (n-1) \sum_{i=1}^{m} \mu(B_i)\mu(A_i) \\ &\geq \sum_{i=1}^{m} \Phi_r(s_i, \gamma_i) s_i + \sum_{i=1}^{m} s_i \gamma_i + (n-1) \sum_{i=1}^{m} \Phi_r(s_i, \gamma_i) s_i^2. \end{aligned} \tag{20}$$

Since the data structure succeeds with probability γ,

$$\sum_{i=1}^{m} s_i \gamma_i \geq \gamma = \Pr_{j \sim [n], v \sim N(u_j)} [\exists i \in [m] : v \in B_i, u_j \in A_i]. \tag{21}$$

Additionally, since D uses at most s space,

$$n \sum_{i=1}^{m} s_i \leq O(s). \tag{22}$$

Using the two constraints in (21) and (22), we will use the estimates of robust expansion in order to find a lower bound for (20). From Lemma 6.2, for any $p, q \in [1, \infty)$ where $(p-1)(q-1) = \sigma^2$ where $\sigma = 1 - \frac{1}{c}$,

$$\begin{aligned} \mathbb{E}[T] &\geq \sum_{i=1}^{m} s_i^{q - \frac{q}{p}} \gamma_i^q + (n-1) \sum_{i=1}^{m} s_i^{q - \frac{q}{p} + 1} \gamma_i^q + \gamma \\ \gamma &\leq \sum_{i=1}^{m} s_i \gamma_i \\ O\left(\frac{s}{n}\right) &\geq \sum_{i=1}^{m} s_i. \end{aligned}$$

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

$$\mathbb{E}[T] \geq \sum_{i \in S} v_i^q \left(s_i^{-\frac{q}{p}} + (n-1) s_i^{-\frac{q}{p}+1}\right) \geq \sum_{i \in S} \left(\frac{\gamma}{|S|}\right)^q \left(s_i^{-\frac{q}{p}} + (n-1) s_i^{-\frac{q}{p}+1}\right) \tag{23}$$

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

$$F = \sum_{i \in S} \left(s_i^{-\frac{q}{p}} + (n-1) s_i^{-\frac{q}{p}+1} \right). \tag{24}$$

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

$$q = 1 - \sigma^2 + \sigma\beta \qquad p = \frac{\beta}{\beta - \sigma} \qquad \beta = \sqrt{\frac{1-\sigma^2}{\rho_u}} \tag{25}$$

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 7.2. *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 4.

Proof. In this setting, p and q are constants, and $q \geq p$. Therefore, $\frac{q}{p} \geq 1$. F can be viewed as consisting of the contributions of each s_i's in Equation 24, constrained by (22). One can easily verify that F minimized when $s_i = O(\frac{s}{n|S|})$, so substituting in (23),

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

since $q - q/p > 0$ and $|S| \leq s$. In addition, p, q and γ are constants, and note the fact $s = n^{1+\rho_u}$, and (25), we let n^{ρ_q} be the best query time we can achieve. Combining these facts, along with the lower bound for ρ_q in (7), we obtain the following relationship between ρ_q and ρ_u:

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

□

Lemma 7.3. *When $\rho_u > \frac{1}{2c-1}$,*

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

where ρ_q and ρ_u satisfy Equation 4.

Proof. We follow a similar pattern to Lemma 7.2.

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

Consider the case when each $\frac{\partial F}{\partial s_i}(s_i) = 0$, by setting $s_i = \dfrac{q}{(p-q)(n-1)}$. 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. Thus, F is minimized at this point, and $\mathbb{E}[T] \geq \left(\frac{\gamma}{|S|}\right)^q |S| \left(\frac{q}{(p-q)(n-1)}\right)^{-\frac{q}{p}}$. Since $q \geq 1$ and $|S| \leq s$,

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

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

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

which is the same expression for ρ_q as in Lemma 7.2. □

Lemma 7.4. *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, 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

$$\mathbb{E}[T] = \Omega(\gamma^q s^{1-q} n^{\frac{q}{p}}) = n^{1-\sigma^2-o(1)},$$

which is the desired expression. □

8 Lower bounds: two-probe data structures

In this section, we prove the cell probe lower bound for $t = 2$ cell probes stated in Theorem 1.4.

We follow the framework in [PTW10] and prove lower bounds for GNS when $U = V$ with measure μ (see Def. 5.2). We assume there is an underlying graph G with vertex set V. For any point $p \in V$, we write p's *neighborhood*, $N(p)$, as the set of points with an edge incident on p in G.

In the 2-probe GNS problem, we are given a dataset $P = \{p_i\}_{i=1}^n \subset V$ of n points as well as a bit-string $x \in \{0,1\}^n$. The goal is to build a data structure supporting the following types of queries: given a point $q \in V$, if there exists a unique neighbor $p_i \in N(q) \cap P$, return x_i with probability at least $\frac{2}{3}$ after making two cell-probes.

We let D denote a data structure with m cells of w bits each. D will depend on the dataset P as well as the bit-string x. We will prove the following theorem.

Theorem 8.1. *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 1.4 will follow from Theorem 8.1 together with the robust expansion bound from Lemma 6.2 for the special case of *non-adaptive* probes. We will later show how to reduce adaptive algorithms losing a sub-polynomial factor in the space for $w = o(\log n)$ in Section 8.6.3. We now proceed to proving Theorem 8.1.

At a high-level, we show that a "too-good-to-be-true", 2-probe data structure implies a weaker notion of 2-query locally-decodable code (LDC) with small noise rate using the same amount of space[7]. Even though our notion of LDC is weaker than Def. 5.4, we adapt the tools for showing 2-query LDC lower bounds from [KdW04]. These arguments, using quantum information theory, are very robust and work well with the weaker 2-query LDC we construct.

We note that [PTW08] was the first to suggest the connection between ANN and LDCs. This work represents the first concrete connection which gives rise to better lower bounds.

Proof structure. The proof of Theorem 8.1 proceeds in six steps.

1. First we use Yao's principle to focus on deterministic non-adaptive data structures for GNS with two cell-probes. We provide distributions over n-point datasets P, as well as bit-strings x and a query q, and assume the existence of a deterministic data structure succeeding with probability at least $\frac{2}{3}$.

2. We simplify 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]).

3. We use ideas from [PTW10] to understand how queries neighboring particular dataset points probe various cells of the data structure. We fix an n-point dataset P with a constant fraction of the points satisfying the following condition: many possible queries in the neighborhood of these points probe disjoint pairs of cells.

[7] 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.

4. For the fixed dataset P, we show that we can recover a constant fraction of bits of x with significant probability even if we corrupt the contents of some cells.

5. We reduce to data structures with 1-bit words in order to apply the LDC arguments from [KdW04].

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

8.1 Deterministic data structures

Definition 8.2. *A non-adaptive randomized algorithm R for the GNS problem making two cell-probes is an algorithm specified by the following two components:*

1. *A procedure which preprocess a dataset $P = \{p_i\}_{i=1}^n$ of n points, as well as a bit-string $x \in \{0,1\}^n$ in order to output a data structure $D \in (\{0,1\}^w)^m$.*

2. *An algorithm R that given a query q, chooses two indices $(i,j) \in [m]^2$ and specifies a function $f_q \colon \{0,1\}^w \times \{0,1\}^w \to \{0,1\}$.*

We require the data structure D and the algorithm R satisfy

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

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

Note that the procedure which outputs the data structure does not depend on the query q, and that the algorithm R does not depend on the dataset P or bit-string x.

Definition 8.3. *We define the following distributions:*

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

Lemma 8.4. *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 succeeding with probability at least $\frac{2}{3}$ when the dataset $P \sim \mathcal{P}$, the bit-string $x \sim \mathcal{X}$, and $q \sim \mathcal{Q}(P)$.*

Proof. We apply 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, we let $A^D(q)$ denote output of the algorithm A. We assume that $A(q)$ outputs a pair of indices (j,k) as well as the function $f_q\colon \{0,1\}^w \times \{0,1\}^w \to \{0,1\}$, and thus, we use $A^D(q)$ as the output of $f_q(D_j, D_k)$. For any fixed dataset $P = \{p_i\}_{i=1}^n$ and bit-string $x \in \{0,1\}^n$,

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

This notation allows us to succinctly state the probability of correctness when the query is a neighbor of p_i.

For the remainder of the section, we let A denote a non-adaptive deterministic algorithm succeeding with 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 $q \sim \mathcal{Q}(P)$.

8.2 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. Since A is deterministic, the indices $(i,j) \in [m]^2$ which A outputs are completely determined by two collections $\mathcal{A}_1 = \{A_{1,j}\}_{j\in[m]}$ and $\mathcal{A}_2 = \{A_{2,j}\}_{j\in[m]}$ which independently partition the query space V. On query q, if $q \in A_{1,i}$ and $q \in A_{2,j}$, algorithm A outputs the indices (i,j).

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

Definition 8.5. *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 8.6. *Let A be a deterministic non-adaptive algorithm for GNS making two cell-probes using m cells. There exists a deterministic non-adaptive algorithm A' for GNS making two cell-probes using $3m$ cells which has low contention and succeeds with the same probability.*

Proof. Suppose $\mu(A_{t,j}) \geq \frac{1}{m}$ for some $j \in [m]$. We partition $A_{t,j}$ into enough sets $\{A_{t,k}^{(j)}\}_k$ of measure $\frac{1}{m}$ and at most one set with measure between 0 and $\frac{1}{m}$. For each of set $A_{t,k}^{(j)}$, we make a new cell j_k with the same contents as cell j. When a query lies inside $A_{t,k}^{(j)}$ the t-th probe is made to the new cell j_k instead of cell j.

We apply the above transformation on all sets with $\mu(A_{t,j}) \geq \frac{1}{m}$. In the resulting data structure, in each partition $\mathcal{A}_1$ and $\mathcal{A}_2$, there can be at most m cells of measure $\frac{1}{m}$ and at most m sets with measure less than $\frac{1}{m}$. Therefore, the transformed data structure has at most $3m$ cells. Since the contents remain the same, the data structure succeeds with the same probability. □

Given Lemma 8.6, we 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 is absorbed in the asymptotic notation.

8.3 Datasets which shatter

We fix some $\gamma > 0$ to be a sufficiently small constant.

Definition 8.7 (Weak-shattering [PTW10])**.** *We say a partition $A_1, \ldots, A_m$ 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 $(\cdot)^+ : \mathbb{R} \to \mathbb{R}^+$ is the identity on positive real numbers and zero otherwise.

Lemma 8.8 (Shattering [PTW10])**.** *Let $A_1, \ldots, A_k$ collection of disjoint subsets of measure at most $\frac{1}{m}$. Then*

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

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 dataset points which are shattered with respect to the collections $\mathcal{A}_1$ and $\mathcal{A}_2$. Intuitively, queries which are near-neighbors of these dataset points probe various disjoint cells in the data structure, so their corresponding bit is stored in many cells.

Definition 8.9. *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 fixed dataset point $p \in P$, we refer to $\beta(p)$ as the set holding the *slack* in the shattering of measure at most 2γ. 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 8.8 implies that $\Pr_{p \sim \mu}[S(\mathcal{A}, p)] \geq 1 - \gamma$.

Lemma 8.10. *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. 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 via a Chernoff bound, 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 8.11. *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. For any fixed dataset $P = \{p_i\}_{i=1}^n$, let

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

Then,

$$\begin{aligned} \frac{2}{3} &\leq \mathop{\mathbb{E}}_{P \sim \mathcal{P}}[\mathbf{P}] \\ &= (1 - o(1)) \cdot \mathop{\mathbb{E}}_{P \sim \mathcal{P}}[\mathbf{P} \mid \ P \text{ is good}] + o(1) \cdot \mathop{\mathbb{E}}_{P \sim \mathcal{P}}[\mathbf{P} \mid \ P \text{ is not good}] \\ \frac{2}{3} - o(1) &\leq (1 - o(1)) \cdot \mathop{\mathbb{E}}_{P \sim \mathcal{P}}[\mathbf{P} \mid P \text{ is good}]. \end{aligned}$$

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

8.4 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 8.11, i.e., P satisfies

$$\Pr_{x \sim \mathcal{X}, q \sim \mathcal{Q}(P)}[A^D(q) = x_i] \geq \frac{2}{3} - o(1). \tag{26}$$

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 $D \in (\{0, 1\}^w)^m$.

Definition 8.12. *We call $D' \in (\{0, 1\}^w)^m$ a* corrupted *version of D at k cells if D and D' differ on at most k cells, i.e., if $|\{i \in [m] : D_i \neq D'_i\}| \leq k$.*

Definition 8.13. *For a fixed $x \in \{0, 1\}^n$, let*

$$c_x(i) = \Pr_{q \sim N(p_i)}[A^D(q) = x_i] \tag{27}$$

denote the recovery probability of bit i. *Note that from the definitions of $\mathcal{Q}(P)$, $\mathbb{E}[c_x(i)] \geq \frac{2}{3} - o(1)$, where the expectation is taken over $x \sim \mathcal{X}$ and $i \in [n]$.*

In this section, we show there exist a subset $S \subset [n]$ of size $\Omega(n)$ where each $i \in S$ has constant recovery probability averaged over $x \sim \mathcal{X}$, even if the algorithm probes a corrupted version of data structure. We let $\varepsilon > 0$ be a sufficiently small constant.

Lemma 8.14. *Fix a vector $x \in \{0,1\}^n$, and let $D \in (\{0,1\}^w)^m$ be the data structure that algorithm A produces on dataset P and 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,*

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

Proof. Note that $c_x(i)$ represents the probability that algorithm A run on a uniformly chosen query from the neighborhood of p_i returns the correct answer, i.e. x_i. We denote the subset $C_1 \subset N(p)$ of queries that when run on A return x_i; so, $\mu(C_1) = c_x(i)$ by definition.

By assumption, p_i is (K, γ)-weakly shattered by $\mathcal{A}_1$ and $\mathcal{A}_2$, so by Def. 8.9, we specify some $\beta(p) \subset N(p)$ where $\mu(C_1 \cap \beta(p)) \leq \mu(\beta(p)) \leq 2\gamma$. Let $C_2 = C_1 \setminus \beta(p)$, where $\mu(C_2) \geq c_i(x) - 2\gamma$. Again, by assumption that p_i is (K, γ)-weakly shattered, each $j \in [m]$ and $t \in \{1, 2\}$ satisfy $\mu(C_2 \cap A_{t,j}) \leq \frac{1}{K}$. Let $\Delta \subset [m]$ be the set of εK cells where D and D' differ, and let $C_3 \subset C_2$ be given by

$$C_3 = C_2 \setminus \left(\bigcup_{j \in \Delta} (A_{1,j} \cup A_{2,j}) \right).$$

Thus,

$$\mu(C_3) \geq \mu(C_2) - \sum_{j \in \Delta} (\mu(C_2 \cap A_{1,j}) + \mu(C_2 \cap A_{2,j})) \geq c_i(x) - 2\gamma - 2\varepsilon.$$

If $q \in C_3$, then on query q, algorithm A probes cells outside of Δ, so $A^{D'}(q) = A^D(q) = x_i$. □

Lemma 8.15. *There exists a set $S \subset [n]$ of size $\Omega(n)$ with the following property. If $i \in S$, then events $S(\mathcal{A}_1, p_i)$ and $S(\mathcal{A}_2, p_i)$ occur, and*

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

where ν is a constant. [8]

Proof. For $i \in [n]$, let E_i be the event that $S(\mathcal{A}_1, p_i)$ and $S(\mathcal{A}_2, p_i)$ occur and $\mathbb{E}_{x \sim \mathcal{X}}[c_x(i)] \geq \frac{1}{2} + \nu$. Additionally, let

$$\mathbf{P} = \Pr_{i \in [n]}[E_i].$$

[8] One can think of ν as around $\frac{1}{10}$.

We set $S = \{i \in [n] \mid E_i\}$, so it remains to show that $\mathbf{P} = \Omega(1)$. To this end,

$$\begin{aligned}
\frac{2}{3} - o(1) &\le \mathop{\mathbb{E}}_{x\sim\mathcal{X}, i\in[n]}[c_x(i)] && \text{(by Equations 26 and 27)}\\
&\le 4\gamma + \mathbf{P} + \left(\frac{1}{2} + \nu\right) \cdot (1 - \mathbf{P}) && \text{(since } P \text{ is good)}\\
\frac{1}{6} - o(1) - 4\gamma - \nu \le \mathbf{P} \cdot \left(\frac{1}{2} - \nu\right).
\end{aligned}$$

□

Fix the set $S \subset [n]$ satisfying the conditions of Lemma 8.15. We combine Lemma 8.14 and Lemma 8.15 to obtain the following condition on the dataset.

Lemma 8.16. *Whenever $i \in S$,*

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

where $\eta = \nu - 2\gamma - 2\varepsilon$ and D' differs from D in εK cells.

Proof. Whenever $i \in S$, p_i is (K, γ)-weakly shattered. By Lemma 8.15, A outputs x_i with probability $\frac{1}{2} + \nu$ on average when probing the data structure D on input $q \sim N(p_i)$, i.e

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

Therefore, from Lemma 8.14, 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,

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

□

Summarizing the results of the section, we conclude with the following theorem.

Theorem 8.17. *There exists a two-probe algorithm and a subset $S \subseteq [n]$ of size $\Omega(n)$, satisfying the following property. When $i \in S$, we can recover x_i with probability at least $\frac{1}{2} + \eta$ over a random choice of $x \sim \mathcal{X}$, even if we probe a corrupted version of the data structure at εK cells.*

Proof. We describe how one can recover bit x_i from a data structure generated by algorithm A. In order to recover x_i, we generate a random query $q \sim N(p_i)$ and probe the data structure at the cells specified by A. From Lemma 8.16, there exists a set $S \subset [n]$ of size $\Omega(n)$ for which the described algorithm recovers x_i with probability at least $\frac{1}{2} + \eta$, where the probability is taken on average over all possible $x \in \{0, 1\}^n$. □

Since we fixed the dataset $P = \{p_i\}_{i=1}^n$ satisfying the conditions of Lemma 8.11, we will abuse a bit of notation, and refer to algorithm A as the algorithm which recovers bits of x described in Theorem 8.17. 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.

8.5 Decreasing the word size

In order to apply the lower bounds of 2-query locally-decodable codes, we reduce to the case when the word size w is one bit.

Lemma 8.18. *There exists a deterministic non-adaptive algorithm A' which on input $x \in \{0,1\}^n$ builds a data structure D' using $m \cdot 2^w$ cells of 1 bit. For any $i \in S$ as well as any corruption C which differs from D' in at most εK cells 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 $D \in (\{0,1\}^w)^m$ on input $x \in \{0,1\}^n$, construct the following data structure $D' \in (\{0,1\})^{m\cdot 2^w}$. For each cell $D_j \in \{0,1\}^w$, make 2^w cells containing all parities of the w bits in D_j. This procedure increases the size of the data structure by a factor of 2^w.

Fix $i \in S$ and $q \in N(p_i)$ be a query. If the 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. If C' is a sequence of $m \cdot 2^w$ cells which differ in εK many cells from D', then

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

whenever $i \in S$. □

For the remainder of the section, we will prove a version of Theorem 8.1 for algorithms with 1-bit words. Given Lemma 8.18, 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.

8.6 Connection to locally-decodable codes

To complete the proof of Theorem 8.1, it remains to prove the following lemma.

Lemma 8.19. *Let A be a non-adaptive deterministic algorithm which makes 2 cell probes to a data structure D of m cells of 1 bit and recover x_i with probability $\frac{1}{2}+\eta$ on random input $x \in \{0,1\}^n$ even after εK cells are corrupted 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 applied to LDC lower bounds.

8.6.1 Crash course in quantum computing

We introduce a few concepts from quantum computing that are necessary in our subsequent arguments. The quantum state of a *qubit* is described by a unit-length vector in $\mathbb{C}^2$. We write the quantum state as a linear combination of the basis states $\binom{1}{0} = |0\rangle$ and $\binom{0}{1} = |1\rangle$. The quantum state $\alpha = \binom{\alpha_1}{\alpha_2}$ 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 $|\alpha_1|^2 + |\alpha_2|^2 = 1$. The quantum state of an *m-qubit system* is a unit 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 Hermitian 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 $\frac{M_i|\phi\rangle}{\|M_i|\phi\rangle\|}$, 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 $\sum_{c\in\{0,1\}, j\in[m]} (-1)^{c\cdot y_j} \alpha_{cj} |c\rangle |j\rangle$.

8.6.2 Weak quantum random access codes from GNS algorithms

Definition 8.20. *$C : \{0,1\}^n \to \{0,1\}^m$ 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 8.21 (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 8.21 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 8.22 (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 8.21. We assume the existence of a GNS algorithm A which builds a data structure $D \in \{0,1\}^m$.

Our algorithm A from Theorem 8.17 does not satisfy the strong properties of an LDC, preventing us from applying 8.21 directly. However, it does have some LDC-*ish* guarantees. In particular, we can recover bits in the presence of 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 8.23. *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 8.19). 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 8.23, we can complete the proof of Lemma 8.19.

Proof of Lemma 8.19. 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 8.23, 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

$$\begin{aligned}H(XM) &= n+\frac{1}{2^n}\sum_{x\in\{0,1\}^n}H(\rho_x)\geq n = H(X)\\ H(M) &\leq s,\end{aligned}$$

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,

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

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

$$\begin{aligned}\frac{2m}{a^2\varepsilon K}(\log m+1) &\geq \Omega\left(n\eta^2\right)\\ \frac{m\log m}{n} &\geq \Omega\left(\varepsilon K\eta^2\right).\end{aligned}$$

□

It remains to prove Lemma 8.23, 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 8.24. *Fix an $x\in\{0,1\}^n$ and $i\in[n]$. Let $D\in\{0,1\}^m$ 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. □

Without loss of generality, 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 8.24, we know there exist a state $|Q_i\rangle$ and a measurement $\{R, I - R\}$ 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 8.25. *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

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

which we can simplify to say

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

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:

$$\mathop{\mathbb{E}}_{x\in\{0,1\}^n}\left[p\left(A_i(x)+B(x)\right) \mid x_i = 1\right] + \mathop{\mathbb{E}}_{x\in\{0,1\}^n}\left[p\left(A_i(x)+B(x)\right) \mid x_i = 0\right] \geq \frac{8\eta}{7} \tag{28}$$

$$\mathop{\mathbb{E}}_{x\in\{0,1\}^n}\left[p\left(A_i(x)-B(x)\right) \mid x_i = 1\right] + \mathop{\mathbb{E}}_{x\in\{0,1\}^n}\left[p\left(A_i(x)-B(x)\right) \mid x_i = 0\right] \geq \frac{8\eta}{7} \tag{29}$$

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 (28) and (29) that we get the desired expression. □

Lemma 8.26. *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

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

From Lemma 8.25, 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

$$\begin{aligned}\mathop{\mathbb{E}}_{x\in\{0,1\}^n}&[\Pr[\text{returns correctly}]] \\ &= \frac{1}{2}\mathop{\mathbb{E}}_{x\in\{0,1\}^n}[\Pr[\text{returns } 1] \mid x_i = 1] + \frac{1}{2}\mathop{\mathbb{E}}_{x\in\{0,1\}^n}[\Pr[\text{returns } 0] \mid x_i = 0]\,.\end{aligned} \tag{30}$$

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 (30),

$$\mathop{\mathbb{E}}_{x\in\{0,1\}^n}[\Pr[\text{returns correctly}]] = \frac{1}{2}(1-q)q_1 + \frac{1}{2}(q + (1-q)(1-q_0)) \geq \frac{1}{2} + \frac{b}{2}.$$

□

Now we can finally complete the proof of Lemma 8.23.

Proof of Lemma 8.23. 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 8.26. The following simple calculation (done in [KdW04]) gives the desired probability of success on random input,

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

□

8.6.3 On adaptivity

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

Lemma 8.27. *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 Lemma 8.27, from an adaptive algorithm succeeding with probability $\frac{2}{3}$, we obtain a non-adaptive algorithm succeeding with probability $\frac{1}{2} + \Omega(2^{-w})$. This value is lower than the intended $\frac{2}{3}$, but we may still reduce to a weak LDC, where we require $\gamma = \Theta(2^{-w})$, $\varepsilon = \Theta(2^{-w})$, and $|S| = \Omega(2^{-w} n)$. With these minor changes to the parameters in Subsections 8.1 through 8.6, one can easily verify

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

This inequality 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 1.3. In particular, for any $p, q \in [1, \infty)$ where $(p-1)(q-1) = \sigma^2$, we have

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

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 $n \to \infty$. Then,

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

9 Acknowledgments

We would like to thank Jop Briët for helping us to navigate literature about LDCs. We also thank Omri Weinstein for useful discussions. Thanks to Adam Bouland for educating us on the topic of quantum computing. Thijs Laarhoven is supported by the SNSF ERC Transfer Grant CRETP2-166734 FELICITY. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-16-44869. It is also supported in part by NSF CCF-1617955 and Google Research Award.

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.

[ACW16] Josh Alman, Timothy M. Chan, and Ryan Williams. Polynomial representations of threshold functions with applications. In *FOCS*, 2016.

[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.

[AGK06] Arvind Arasu, Venkatesh Ganti, and Raghav Kaushik. Efficient exact set-similarity joins. In *Proceedings of the 32nd international conference on Very large data bases*, pages 918–929. VLDB Endowment, 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 Pǎtraşcu. On the optimality of the dimensionality reduction method. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 449–458, 2006.

[ALRW16] Alexandr Andoni, Thijs Laarhoven, Ilya Razenshteyn, and Erik Waingarten. Lower bounds on time-space trade-offs for approximate near neighbors. *CoRR*, abs/1605.02701, 2016.

[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 32nd International Symposium on Computational Geometry*, 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.

[Chr17] Tobias Christiani. A framework for similarity search with space-time tradeoffs using locality-sensitive filtering. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2017.

[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.

[DG99] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of the Johnson–Lindenstrauss lemma. *ICSI technical report TR-99-006, Berkeley, CA*, 1999.

[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 20th Annual Symposium on Computational Geometry*, 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.

[GPY94] Daniel H. Greene, Michal Parnas, and F. Frances Yao. Multi-index hashing for information retrieval. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 722–731, 1994.

[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.

[Ind00] Piotr Indyk. Dimensionality reduction techniques for proximity problems. *Proceedings of the Ninth ACM-SIAM Symposium on Discrete Algorithms*, 2000.

[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.

[LLR94] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 577–591, 1994.

[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.

[MO15] Alexander May and Ilya Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In *EUROCRYPT*, 2015.

[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.

[OvL81] Mark H. Overmars and Jan van Leeuwen. Some principles for dynamizing decomposable searching problems. *Information Processing Letters*, 12(1):49–53, 1981.

[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.