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

November 7, 2016

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:

$$c^2\sqrt{\rho_q}+(c^2-1)\sqrt{\rho_u}=\sqrt{2c^2-1}. \tag{0.1}$$

In particular, for the approximation $c=2$ we get:

- Space $n^{1.77\cdots}$ and query time $n^{o(1)}$, significantly improving upon known data structures that support very fast queries [IM98, KOR00];
- Space $n^{1.14\cdots}$ and query time $n^{0.14\cdots}$, matching the optimal data-dependent Locality-Sensitive Hashing (LSH) from [AR15];
- Space $n^{1+o(1)}$ and query time $n^{0.43\cdots}$, 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 (0.1) 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 (0.1) 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 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. The full version containing all the proofs is available at https://arxiv.org/abs/1608.03580

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 fac-

tor $c > 1$. 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 Hamming distance (ℓ_1), the optimal value is $\rho = \frac{1}{c} \pm o(1)$ [IM98, MNP07, OWZ14]. For the Euclidean metric (ℓ_2), it is $\rho = \frac{1}{c^2} \pm o(1)$ [IM98, DIIM04, AI06, MNP07, OWZ14].

More recently, it has been shown that better bounds on ρ are possible if the 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 time vs. space trade-offs for a fixed approximation c. Indeed, data structures with *polynomial* space and *poly-logarithmic* query time were simultaneously introduced [IM98, KOR00].

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 techniques to prove $\omega(\log n)$ query time lower bounds. Due to this state of affairs, one may rely on *restricted* models of computation, which nevertheless capture existing upper bounds.

Early lower bounds for NNS were 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

[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.

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 Hamming 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*

$$c^2\sqrt{\rho_q} + \big(c^2-1\big)\sqrt{\rho_u} \geq \sqrt{2c^2-1}, \tag{1.2}$$

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 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 $1 \leq p < 2$ with c^2 replaced with c^p. In particular, for the Hamming space we get:

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

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 $d \cdot n^{\rho_u+o(1)}$. To apply this technique, one needs to ensure that preprocessing time is near-linear in the space used, which is the case for our data structure.

1.5.1 Techniques We now describe at a high level the proof of Theorem 1.1. 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. The algorithm for random instances uses 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 significantly worse bound than (1.2).

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 vectors $z_1, z_2, \ldots, z_T$. The sets $P_i \subseteq P$ (not necessarily disjoint) are defined for each z_i:

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

We then recurse by repeating the above procedure on each non-empty P_i. We stop the recursion once we reach K-th level in our tree, and store the remaining

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

set at each leaf. The above procedure generates a tree of depth K and degree at most T, where each leaf holds a subset of the dataset. To process a query point $q \in S^{d-1}$, we start at the root and descend into (potentially multiple children) P_i for which $\langle z_i, q\rangle \geq \eta_q$. 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 [AIL+15], which are routinely needed for understanding "Gaussian-induced" partitions.

This algorithm has two important consequences. First, we obtain the desired trade-off (1.2) 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:

$$(c^2+1)\sqrt{\rho_q} + (c^2-1)\sqrt{\rho_u} \geq 2c. \tag{1.3}$$

Even though it is inferior to the desired bound from (1.2)[3], it is still quite non-trivial. In particular, (1.3) 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 (1.3) 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 Eqn. (1.3) 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 smaller than 1, and we recurse on each cluster. For the remainder of the 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 (indeed, 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 isotropic", etc. 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 this algorithm will be simplified in the future.

Previous work suggested that time–space trade-off 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 both 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 $n^{o(1)}$,*

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

- *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 $\{-1,1\}^d$ of [O'D14].

1.6.2 Two cell probes To state our results for two cell probes, we first define the *decision* version of ANN (first introduced in [PTW10]). Suppose that with every data point $p \in P$ we associate a bit $x_p \in \{0,1\}$. A new goal is: given a query $q \in \{-1,1\}^d$ which is 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 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 adapt 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 1.1. *A* list-of-points data structure *for the ANN problem is defined as follows:*

- *We fix (possibly randomly) sets* $A_i \subseteq \{-1,1\}^d$*, for* $i = 1 \dots m$*; 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 $\|p - q\|_1 \leq cr$. If it exists, return p.*

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

For this model, we prove the following theorem.

THEOREM 1.5. (SEE SECTION 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:*

$$c\sqrt{\rho_q} + (c-1)\sqrt{\rho_u} \geq \sqrt{2c-1}. \tag{1.4}$$

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 1.5 uses the random instance as a hard distribution, for which being data-dependent seems to offer no advantage. Indeed, a data-dependent lower bound in the standard LSH regime (where $\rho_q = \rho_u$) has been recently shown in [AR16], and matches (1.4) for $\rho_q = \rho_u$.

1.7 Related work: past and concurrent 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 [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 manuscript [Chr16]. The recent manuscript of [Chr16] has significant intersection with this paper, and its relation to the arXiv preprints [Laa15c, ALRW16] that are now subsumed by this paper. 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 [Chr16] and [ALRW16] have been announced on arXiv. In [Chr16], the author also obtains an upper bound similar to Theorem 1.2 (trade-offs for the entire $\mathbb{R}^d$, but which are sub-optimal), using a different (data-*independent*) reduction from the worst-case to the spherical case. Besides the upper bound, the author of [Chr16] also proved a conditional lower bound, similar to our lower bound from Theorem 1.5. This lower bound of [Chr16] 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 upper bound for the Euclidean case comes 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 good 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 frequently write $\alpha^2(s)$ and $\beta^2(s)$ to denote $(\alpha(s))^2$ and $(\beta(s))^2$, respectively.

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

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

where $u \in S^{d-1}$ is an arbitrary point on the unit sphere. 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

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

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 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 $N(0, 1)$.

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

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

The second is follows from a standard computation (see the appendix of [AIL+15] for a proof).

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 $1 + o(1)$.

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

THEOREM 3.1. *For every $c > 1$, $r > 0$, $\rho_q \geq 0$ and $\rho_u \geq 0$ such that $cr < 2$ and*

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

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 3.1 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 (3.5) 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 the reduction from [Val15].

COROLLARY 3.1. *For every $c > 1$, $r > 0$, $\rho_q \geq 0$ and $\rho_u \geq 0$ such that*

$$\big(c^2 + 1\big)\sqrt{\rho_q} + \big(c^2 - 1\big)\sqrt{\rho_u} \geq 2c, \tag{3.5}$$

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 $n^{\rho_q+o(1)}$.*

Second, Theorem 3.1 gives an improved tradeoff for *random* instances as defined in Section 2, corresponding to the case $r = \frac{\sqrt{2}}{c}$. Note that (3.5) is the worst when $r \to 0$. This means that for random instances we can obtain a trade-off between ρ_q and ρ_u which is significantly better than (3.5). Thus, we give the trade-off for random instances, which matches the trade-off promised in Theorem 1.1.

COROLLARY 3.2. *For every $c > 1$, $\rho_q \geq 0$ and $\rho_u \geq 0$ such that*

$$c^2\sqrt{\rho_q} + \big(c^2 - 1\big)\sqrt{\rho_u} \geq \sqrt{2c^2 - 1}, \tag{3.6}$$

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.

Figure 3 plots the time-space trade-off in (3.5) and (3.6) for $c = 2$. Note that (3.6) is much better than (3.5), especially when $\rho_q = 0$, where (3.5) gives space $n^{2.77\cdots}$, while (3.6) gives much better space $n^{1.77\cdots}$. In Section 4, we show how to get best of both worlds: we obtain the trade-off (3.6) for *worst-case* instances. The remainder of the section is devoted to proving Theorem 3.1.

3.3 Data structure

3.3.1 Description Fix K and T to be positive integers, where 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 r, 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 r excluding the root (but including v). Each node v, except for the root, stores a random Gaussian 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 r, $P_r = P$, because $\mathcal{L}_r = \emptyset$ (remember that the path $\mathcal{L}_r$ excludes the root). 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, ..., $K-1$, 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 r 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.

We provide pseudocode for the description of the data structure above in Figure 2. The procedure BUILD(P, 0, $\perp$) builds the data structure for dataset P and returns the root of the tree, r. The procedure QUERY(q, r) queries the data structure with root r at point q.

3.3.2 Analysis

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

```
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 ← 1...T do
            sample a Gaussian vector z' ~ N(0,1)^d
            P'' ← {p ∈ P' | ⟨z', p⟩ ≥ η_u}
            if P'' ≠ ∅ then
                add Build(P'', l + 1, z') as a child of v
    return v
```

Figure 1: Pseudocode for building procedure of data-independent partitions

```
function Query(q, v)
    if v.l = K then
        for p ∈ v.P do
            if ‖p − q‖ ≤ cr then
                return p
    else
        for v' : v' is a child of v do
            if ⟨v'.z, q⟩ ≥ η_q then
                p ← Query(q, v')
                if p ≠ ⊥ then
                    return p
    return ⊥
```

Figure 2: Pseudocode for querying procedure of data-independent partitions

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 $\|p-q\|_2 \le r$. Thus, the data structure succeeds when Query(q, r) returns some point $p' \in P$ with $\|q-p'\|_2 \le cr$.

Lemma 3.3. *If*

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

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

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

Lemma 3.4. *The expected space consumption of the data structure is at most*

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

Query time Finally, we analyze the query time.

Lemma 3.5. *The expected query time is at most*

$$n^{o(1)} \cdot T \cdot (1 + T \cdot F(\eta_q))^K + n^{1+o(1)} \cdot (T \cdot G(cr, \eta_u, \eta_q))^K. \quad (3.7)$$

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 3.1.

First, we set $K = \sqrt{\ln n}$. In order to satisfy the requirement of Lemma 3.3, we set

$$T = \frac{100}{G(r, \eta_u, \eta_q)}. \quad (3.8)$$

Second, we (approximately) balance the terms in the query time (3.7). Toward this goal, we aim to have

$$F(\eta_q)^K = n \cdot G(cr, \eta_u, \eta_q)^K. \quad (3.9)$$

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 (3.9), 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

$$\left|\sqrt{\sigma} - \alpha(cr)\sqrt{\tau}\right| = \beta(cr), \quad (3.10)$$

since $\alpha^2(cr) + \beta^2(cr) = 1$. 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 (3.10) is satisfied. Combining these identities, we obtain (3.5).

Namely, we set $\sqrt{\sigma} = \alpha(cr)\sqrt{\tau} + \beta(cr)$ to satisfy (3.10). 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 is the trade-off in (3.5).

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

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 (1.2) (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 ($\mathbb{R}^d = \mathbb{R}^{d/K} \times \ldots \times \mathbb{R}^{d/K}$), 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 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

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

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:

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

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.

4 Upper bounds: data-dependent partitions

In this section we give 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*

$$c^2\sqrt{\rho_q} + \left(c^2 - 1\right)\sqrt{\rho_u} \geq \sqrt{2c^2 - 1}, \tag{4.11}$$

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 3.1 and Corollary 3.2. Like Corollary 3.1, our data structure works for worst-case datasets; however, we improve upon the trade-off between time and space complexity from Corollary 3.1 to that of random

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

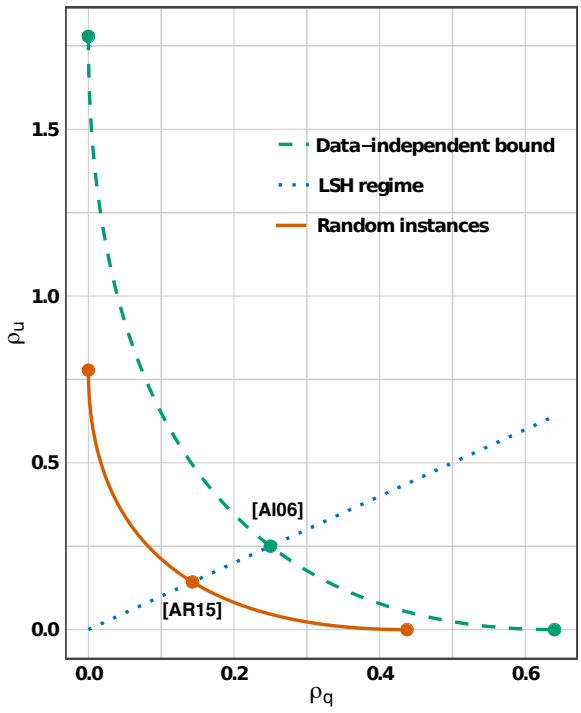

Figure 3: Trade-offs between query time $n^{\rho_q+o(1)}$ and space $n^{1+\rho_u+o(1)}$ for the Euclidean distance and approximation $c = 2$. The green dashed line corresponds to the simple data-independent bound for *worst-case* instances from Corollary 3.1. The red solid line corresponds to the bound for *random* instances from Corollary 3.2, 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].

instances in Corollary 3.2. See Figure 3 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 $\sqrt{\log n}$ 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 $\widetilde{O}(\sqrt{\log n})$ [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 $d = \Theta(\log n \cdot \log\log n)$; 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. Using a reduction from [Val15], we may also assume the entire dataset P_0 and a query lie on a sphere $\partial B(0, R)$ of radius $R = \widetilde{O}_c(\sqrt{\log n})$.

We partition P_0 into various components: s *dense* components, denoted by $C_1, C_2, \ldots, C_s$, 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 4). 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 slightly smaller ball E_i of radius $(1 - \Theta(\varepsilon^2))R$ (see Figure 4). 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 $(1-\Theta(\varepsilon^2))R$. We treat the pseudo-random component $\widetilde{P}$ similarly to the random instance from Section 2 described in

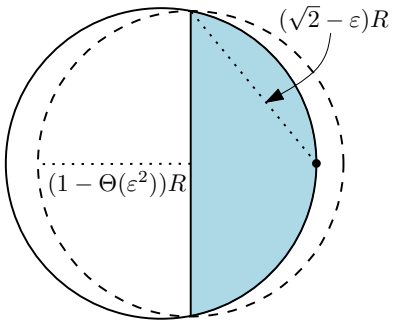

Figure 4: Covering a spherical cap of radius $(\sqrt{2}-\varepsilon)R$

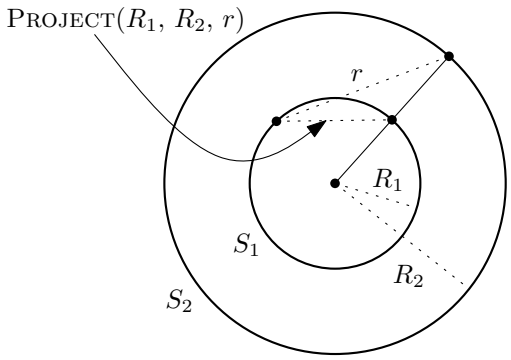

Figure 5: The definition of Project

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 $(1 - \Theta(\varepsilon^2))$. Initially $R = \widetilde{O}_c(\sqrt{\log n})$, so in $O_c(\log\log n/\varepsilon^2)$ iterations of removing dense clusters, we arrive at the case of $R \leq c/\sqrt{2}$, where Corollary 3.2 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 δ and treat each annulus as a sphere. This discretization of a ball adds to the complexity of the analysis, but is not fundamental from the conceptual point of view.

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 6 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 perform the reduction from [Val15]: we translate the points and think of them as lying on a sphere of radius $R = \widetilde{O}_c(\sqrt{\log n})$. 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 5). 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.

ProcessSphere. We consider three base cases. If $l = K$, we stop and store P explicitly. 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. If the algorithm from Section 3 can give the desired point on the time–space trade-off, then we choose $\eta_u, \eta_q > 0$ and T appropriately and build a single level of the tree from Section 3. We check for this last condition using (3.5).

Otherwise, we proceed by first 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 $B(o, R + r_2)$.

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 radius δ. First, we round all distances from points to o to a multiple of δ. This rounding can change the distance between any pair of points by at most 2δ by the triangle inequality. Then, for every possible *admissible* distances δi from o to a data point and δj from o to a query, we build a separate data structure via PROCESSSPHERE. By admissible distances δi and δj, we mean those distances where $|\delta(i-j)| \leq r_1 + 2\delta$ for integers i and j. Distances which are not admissible yield trivial instances.

We compute the new distance thresholds $\widetilde{r}_1$ and $\widetilde{r}_2$ for each new instance of PROCESSSPHERE as follows. After rounding, the new thresholds for the ball instance should be $r_1 + 2\delta$ and $r_2 - 2\delta$, since distances can change by at most 2δ. The new thresholds for the projected points are given by PROJECT.

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 δ and enumerate all possible the 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:

- $\varepsilon = \frac{1}{\log\log\log n}$;
- $\delta = \exp\big(-(\log\log\log n)^C\big)$;
- $\tau = \exp\big(-\log^{2/3} n\big)$,

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

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

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 6.

- 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 any number of calls to ProcessBall, corresponding to cluster nodes, and T calls to ProcessSphere. The calls to ProcessBall do not replicate any points. The only points which may be replicated are in the pseudo-random remainder. These points can be replicated in the T calls to ProcessSphere.
- 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 δ around the center as well as possible distance for a query. For each annulus, there are $\frac{r_1}{\delta} + 2$ notable distances; after rounding by δ, a valid query can be at most $r_1 + 2\delta$ away from a particular annulus, thus, each point gets duplicated at most $\frac{r_1}{\delta} + 2$ 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

Lemma 4.1. *The following invariants hold.*

- *At any moment one has* $\frac{r_2}{r_1} \geq c - o_c(1)$ *and* $r_2 \leq c + o_c(1)$.
- *At any moment the number of calls to* ProcessBall *in the recursion stack is at most* $O_c\left((\varepsilon^{-1} \log\log n)^{O(1)}\right)$.

Lemma 4.2. *During the algorithm we will always be able to choose* η_u *and* η_q *such that:*

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

Lemma 4.3. *The probability of success of the data structure is at least* 0.9.

Lemma 4.4. *The total space the data structure occupies is at most* $n^{1+\rho_u+o(1)}$ *in expectation.*

Lemma 4.5. *The expected query time is at most* $n^{\rho_q+o(1)}$.

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.

function ProcessSphere(P, r_1, r_2, o, R, l)
  if $l = K$ **then**
    store P explicitly
    return
  if $r_2 \geq 2R$ **then**
    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**
    $m \leftarrow |P|$
    $\widehat{R} \leftarrow (\sqrt{2} - \varepsilon)R$
    while $\exists x \in \partial B(o, R) : |B(x, \widehat{R}) \cap P| \geq \tau m$ **do**
      $B(\widetilde{o}, \widetilde{R}) \leftarrow$ 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 η_u and η_q such that:
  - $F(\eta_u)/G(r_1/R, \eta_u, \eta_q) \leq n^{\rho_u/K + o(1)}$;
  - $F(\eta_q)/G(r_1/R, \eta_u, \eta_q) \leq n^{\rho_q/K + o(1)}$;
  - $G(r^*/R, \eta_u, \eta_q)/G(r_1/R, \eta_u, \eta_q) \leq n^{(\rho_q - 1)/K + o(1)}$.

  $T \leftarrow 100/G(r_1/R, \eta_u, \eta_q)$
  for $i \leftarrow 1 \dots T$ **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**
    store any point from P
    return
  $P \leftarrow \{o + \delta\lceil\frac{\|p-o\|}{\delta}\rceil \cdot \frac{p-o}{\|p-o\|} \mid p \in P\}$
  for $i \leftarrow 0 \dots \lceil\frac{R}{\delta}\rceil$ **do**
    $\widetilde{P} \leftarrow \{p \in P : \|p - o\| = \delta i\}$
    if $\widetilde{P} \neq \emptyset$ **then**
      for $j \leftarrow 0 \dots \lceil\frac{R + r_1 + 2\delta}{\delta}\rceil$ **do**
        if $\delta|i - j| \leq r_1 + 2\delta$ **then**
          $\widetilde{r_1} \leftarrow$ Project(δi, δj, $r_1 + 2\delta$)
          $\widetilde{r_2} \leftarrow$ Project(δi, δj, $r_2 - 2\delta$)
          ProcessSphere($\widetilde{P}$, $\widetilde{r_1}$, $\widetilde{r_2}$, o, δi, l)
function Project(R_1, R_2, r)
  return $\sqrt{R_1(r^2 - (R_1 - R_2)^2)/R_2}$

Figure 6: Pseudocode of the data structure (SEB stands for *smallest enclosing ball*)

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

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

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 $i \in [d]$. With probability σ, $y_i = x_i$. With probability $1-\sigma$, y_i is set uniformly at random.

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

$$T_\sigma f(x) \quad = \mathbb{E}_{y \sim N_\sigma(x)}[f(y)] \tag{6.12}$$

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 6.1. (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 $m \geq 1$,*

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

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 Generalized Small-Set Expansion Theorem:

THEOREM 6.1. ([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 Bonami-Beckner Inequality and the Hölder Inequality.

THEOREM 6.2. ([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*

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

THEOREM 6.3. *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*

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

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.

6.2 Lower bounds for one-probe data structures In this section, we give Theorem 1.3. Our proof relies on the main result of [PTW10] for the GNS problem:

THEOREM 6.4. (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*

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

The lower bound follows from a direct application of Lemma 6.1 to Theorem 6.4, where we let $p = 1 + \frac{\log\log n}{\log n}$, and $q = 1 + \sigma^2 \frac{\log n}{\log\log n}$.

COROLLARY 6.1. *Any 1 cell probe data structures with cell size $O(\log n)$ for c-approximate nearest neighbors on the sphere in ℓ_2 needs $n^{1+\frac{2c^2-1}{(c^2-1)^2}-o(1)}$ many cells.*

7 Lower bounds: list-of-points data structures

In this section we give Theorem 1.5, i.e., a tight lower bound against data structure that fall inside the "list-of-points" model, as defined in Def. 1.1.

Recall that $A_i \subset V$ is the subset of dataset points which get placed in L_i. Let $B_i \subset V$ the subset of

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

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

and $s_i = \mu(A_i)$.

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

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

and taking expectations, we conclude

$$\mathbb{E}[T] \geq \sum_{i=1}^{m} \Phi_r(s_i, \gamma_i) s_i + \sum_{i=1}^{m} s_i \gamma_i + (n-1) \sum_{i=1}^{m} \Phi_r(s_i, \gamma_i) s_i^2.$$

Since the data structure succeeds with probability γ,

$$\sum_{i=1}^{m} s_i \gamma_i \geq \gamma$$

Since we use at most space $O(s)$, $n \sum_{i=1}^{m} s_i \leq O(s)$. Using our estimates of the robust expansion from Lemma 6.1, we conclude Theorem 1.5.

8 Lower bounds: two-probe data structures

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

As in [PTW10], we 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 particular point $p \in V$, its neighborhood $N(p)$ is the set of points with an edge to p in the graph G.

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

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 6.1 for the special case when probes to the data structure are non-adaptive. In the rest of this section, we prove Theorem 8.1. The case of adaptive algorithms loses a sub-polynomial factor in the space for $w = o(\log n)$.

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

We note that [PTW08] was the first to suggest the connection between nearest neighbor search and locally-decodable codes. This work represents the first concrete connection which gives rise to better lower bounds.

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

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 Vi-*

[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.

sion: 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 Pătraşcu. 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.

[Chr16] 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)*, 2016.

[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 Am-

buj 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.

[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.