43 NEAREST NEIGHBORS IN HIGH-DIMENSIONAL SPACES

Alexandr Andoni and Piotr Indyk

INTRODUCTION

In this chapter we consider the following problem: given a set P of points in a high-dimensional space, construct a data structure that given any *query* point q finds the point in P closest to q. This problem, called ***nearest neighbor search***[1], is of significant importance to several areas of computer science, including pattern recognition, searching in multimedia data, vector compression [GG91], computational statistics [DW82], and data mining. Many of these applications involve data sets that are very large (e.g., a database containing Web documents could contain over one billion documents). Moreover, the dimensionality of the points is usually large as well (e.g., in the order of a few hundred). Therefore, it is crucial to design algorithms that scale well with the database size as well as with the dimension.

The nearest neighbor problem is an example of a large class of ***proximity problems***, which, roughly speaking, are problems whose definitions involve the notion of distance between the input points. Apart from nearest neighbor search, the class contains problems like closest pair, diameter, minimum spanning tree and variants of clustering problems.

Many of these problems were among the first investigated in the field of computational geometry. As a result of this research effort, many efficient solutions have been discovered for the case when the points lie in a space of *constant* dimension. For example, if the points lie in the plane, the nearest neighbor problem can be solved with $O(\log n)$ time per query, using only $O(n)$ storage [SH75, LT80]. Similar results can be obtained for other problems as well. Unfortunately, as the dimension grows, the algorithms become less and less efficient. More specifically, their space or time requirements grow *exponentially* in the dimension. In particular, the nearest neighbor problem has a solution with $O(d^{O(1)} \log n)$ query time, but using roughly $n^{O(d)}$ space [Cla88, Mei93]. Alternatively, if one insists on linear or near-linear storage, the best known running time bound for *random* input is of the form $\min(2^{O(d)}, dn)$, which is essentially linear in n even for moderate d. Worse still, the exponential dependence of space and/or time on the dimension (called the "curse of dimensionality") has been observed in applied settings as well. Specifically, it is known that many popular data structures (using linear or near-linear storage), exhibit query time linear in n when the dimension exceeds a certain threshold (usually 10–20, depending on the number of points), e.g., see [WSB98] for more information.

The lack of success in removing the exponential dependence on the dimension led many researchers to conjecture that no efficient solutions exists for these problems when the dimension is sufficiently large (e.g., see [MP69]). At the same time, it raised the question: Is it possible to remove the exponential dependence on d,

[1] Many other names occur in literature, including *best match*, *post office problem* and *nearest neighbor*.

Preliminary version (July 20, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

if we allow the answers to be ***approximate***. The notion of approximation is best explained for the nearest neighbor search: instead of reporting a point p closest to q, the algorithm is allowed to report *any* point within distance $(1+\epsilon)$ times the distance from q to p. Similar definitions can be naturally applied to other problems. Note that this approach is similar to designing efficient approximation algorithms for NP-hard problems.

Over the years, a number of researchers have shown that indeed in many cases approximation enables a reduction of the dependence on dimension from exponential to polynomial. In this chapter we will survey these results. In addition, we will discuss the issue of *proving* that the curse of dimensionality is inevitable if one insists on exact answers, and survey the known results in this direction.

Although this chapter is devoted almost entirely to approximation algorithms with running times polynomial in the dimension, the notion of approximate nearest neighbor was first formulated in the context of algorithms with exponential query times, leading to a large number of results including [Cla94, AMN+98, DGK01, HP01, Cha02a, SSS06, AMM09, ADFM11]. Chapter 32 of this Handbook covers such results in more detail.

43.1 APPROXIMATE NEAR NEIGHBOR

Almost all algorithms for proximity problems in high-dimensional spaces proceed by reducing the problem to the problem of finding an *approximate near neighbor*, which is the decision version of the approximate nearest-neighbor problem. Thus, we start from describing the results for the former problem.

All the NNS algorithms are based on space partitions (even if not always framed this way). We distinguish two broad classes of partitions: 1) data-independent approaches, where the partition is independent of the given dataset P, and 2) data-dependent approaches, where the partition depends on the dataset P.

For the definitions of metric spaces and normed spaces, see Chapter 8.

GLOSSARY

Approximate Near Neighbor, or (r,c)***-NN:*** Given a set P on n points in a metric space $M=(X,D)$, design a data structure that supports the following operation: For any query $q \in X$, if there exists $p \in P$ such that $D(p,q) \le r$, find a point $p' \in P$ such that $D(q,p') \le cr$

Dynamic problems: Problems that involve designing a data structure for a set of points (e.g., approximate near neighbor) and support insertions and deletions of points. We distinguish dynamic problems from their ***static*** versions by adding the word "Dynamic" (or letter "D") in front of their names (or acronyms). E.g., the dynamic version of the approximate near-neighbor problem is denoted by (r,c)-DNN.

Hamming metric: A metric (Σ^d, D) where Σ is a set of *symbols*, and for any $p,q \in \Sigma^d$, $D(p,q)$ is equal to the number of $i \in \{1 \dots d\}$ such that $p_i \neq q_i$.

Preliminary version (July 20, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

DATA-INDEPENDENT APPROACH

The first algorithms for (r, c)-NN in high dimensions were obtained by using the technique of random projections, as introduced in the paper by Kleinberg [Kle97]. Although his algorithms still suffered from the curse of dimensionality (i.e., used exponential storage or had $\Omega(n)$ query time), his ideas provided inspiration for designing improved algorithms.

Later algorithms are based on the technique of *Locality-Sensitive Hashing*. We describe both approaches next.

TABLE 43.1.1 Approximate Near Neighbors under the Hamming distance.

#	APPROX.	QUERY TIME	SPACE	UPDATE TIME
Very low query time	Source: [KOR00] (see also [HIM12])			
	$c = 1 + \epsilon$	$d \log n / \min(\epsilon^2, 1)$	$n^{O(1/\epsilon^2 + \log c/c)}$	$n^{O(1/\epsilon^2 + \log c/c)}$
Low query time	Source: [Pan06, Kap15, Laa15, ALRW17]			
	c	$dn^{o(1)}$	$n^{(\frac{c}{c-1})^2}$	$n^{(\frac{c}{c-1})^2}$
Balanced	Source: [HIM12, AINR14, AR15]			
	c	$dn^{\frac{1}{2c-1}+o(1)}$	$n^{1+\frac{1}{2c-1}+o(1)} + dn$	$dn^{\frac{1}{2c-1}+o(1)}$
Low space	Source: [Ind01a, Pan06, AI06, Kap15, Laa15, ALRW17]			
	c	$n^{\frac{2c-1}{c^2}}$	$dn^{1+o(1)}$	$dn^{o(1)}$

Algorithms via dimensionality reduction. We first focus on the case where all input and query points are binary vectors from $\{0,1\}^d$, and D is the Hamming distance. ***Dimensionality reduction*** is a randomized procedure that reduces the dimension of Hamming space from d to $k = O(\log n/\epsilon^2)$, while preserving a certain range of distances between the input points and the query up to a factor of $1 + \epsilon$. This notion has been introduced earlier in Chapter 8 in the context of Euclidean space. In the case of the Hamming space, the following holds.

THEOREM 43.1.1 *[KOR00] (see also [HIM12])*
For any given $r \in \{1 \ldots d\}$, $\epsilon \in (0,1]$ and $\delta \in (0,1)$, one can construct a distribution over mappings $A : \{0,1\}^d \to \{0,1\}^k$, $k = O(\log(1/\delta)/\epsilon^2)$, and a "scaling factor" S, so that for any $p, q \in \{0,1\}^d$, if $D(p,q) \in [r, 10r]$, then $D(A(p), A(q)) = S \cdot D(p,q)(1 \pm \epsilon)$ with probability at least $1 - \delta$.

The factor 10 can be replaced by any constant. As in the case of Euclidean norm, the mapping A is linear (over the field $GF(2)$). The $k \times n$ matrix A is obtained by choosing each entry of A independently at random from the set $\{0,1\}$. The probability that an entry is equal to 1 is roughly r/d.

The first algorithm in Table 43.1.1 is an immediate consequence of Theorem 43.1.1. Specifically, it allows us to reduce the $(r, c+\epsilon)$-NN problem in d-dimensional space to (r, c)-NN problem in k-dimensional space. Since the *exact*

Preliminary version (July 20, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

nearest neighbor problem in k-dimensional space can be solved by storing the answers to all 2^k queries q, the bound follows.

This resulting algorithm is randomized and has a Monte Carlo guarantee of correctness. One can also obtain stronger guarantees of correctness: Las Vegas guarantee with similar performance and a deterministic algorithm with $3+\epsilon$ approximation are described in [Ind00].

We note that one can apply the same approach to solve the near-neighbor problem in the Euclidean space. In particular, it is fairly easy to solve the $(r, 1+\epsilon)$-NN problem in l_2^d using $n(1/\epsilon)^{O(d)}$ space [HIM12]. Applying the Johnson-Lindenstrauss lemma leads to an algorithm with storage bound similar to (although slightly worse than) the bound of algorithm (1a) [HIM12].

Algorithms via Locality-Sensitive Hashing. The storage bound for the first algorithm in Table 43.1.1 is high and, oftentimes, one needs space to be much closer to the linear in the dataset size. The next two algorithms in the table obtain a better trade-off between space and the query time.

These algorithms are based on the concept of ***Locality-Sensitive Hashing***, or *LSH* [HIM12] (see also [KWZ95, Bro98]). A family of hash functions $h : \{0,1\}^d \to U$ is called (r, cr, P_1, P_2)-sensitive (for $c > 1$ and $P_1 > P_2$) if for any $q, p \in \{0,1\}^d$

- if $D(p,q) \le r$, then $\Pr[h(q) = h(p)] \ge P_1$, and
- if $D(p,q) > cr$, then $\Pr[h(q) = h(p)] \le P_2$,

where $\Pr[\cdot]$ is defined over the random choice of h. We note that the notion of locality-sensitive hashing can be defined for any metric space M in a natural way (see [Cha02b] for sufficient and necessary conditions for existence of LSH for M).

For Hamming space, there are particularly simple LSH families: it is sufficient to take all functions h_i, $i = 1 \dots d$, such that $h_i(p) = p_i$ for $p \in \{0,1\}^d$. The resulting family is sensitive due to the fact that $\Pr[h(p) = h(q)] = 1 - D(p,q)/d$.

TABLE 43.1.2 Approximate Near Neighbors via LSH and data-dependent hashing approaches.

METRIC	TYPE	QUERY TIME	EXPONENT ρ	FOR $c = 2$	REFERENCE
ℓ_1	LSH	$n^\rho d$	$\rho = 1/c$	$\rho = 1/2$	[HIM12]
			$\rho \ge 1/c - o(1)$	$\rho \ge 1/2$	[MNP07, OWZ14]
	Data dependent hashing	$n^\rho d$	$\rho = \frac{1}{2c-1} + o(1)$	$\rho = 1/3$	[AINR14, AR15]
			$\rho \ge \frac{1}{2c-1} - o(1)$	$\rho \ge 1/3$	[AR16]
ℓ_2	LSH	$n^\rho d$	$\rho \le 1/c$	$\rho \le 1/2$	[HIM12, DIIM04]
			$\rho = 1/c^2 + o(1)$	$\rho = 1/4$	[AI06]
			$\rho \ge 1/c^2 - o(1)$	$\rho \ge 1/4$	[MNP07, OWZ14]
	Data dependent hashing	$n^\rho d$	$\rho = \frac{1}{2c^2-1} + o(1)$	$\rho = 1/7$	[AINR14, AR15]
			$\rho \ge \frac{1}{2c^2-1} - o(1)$	$\rho \ge 1/7$	[AR16]

An LSH family with a "large" gap between P_1 and P_2 immediately yields a solution to the (c,r)-NN problem. During preprocessing, all input points p are hashed to the bucket $h(p)$. In order to answer the query q, the algorithm retrieves the points in the bucket $h(q)$ and checks if any one of them is close to q. If the

Preliminary version (July 20, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

gap between P_1 and P_2 is sufficiently large, this approach can be shown to result in sublinear query time. Unfortunately, the P_1/P_2 gap guaranteed by the above LSH family is not large enough. However, the gap can be amplified by concatenating several independently chosen hash functions $h_1 \dots h_l$ (i.e., hashing the points using functions h' such that $h'(p) = (h_1(p), \dots, h_l(p))$. This decreases the probability of finding a near neighbor in one hash tables, and therefore we build a few hash tables, and, at query time, look-up one bucket of each of the hash tables. Details can be found in [HIM12].

The overall LSH algorithm uses $O(n^\rho)$ hash tables, where $\rho = \frac{\log P_1}{\log P_2} \in (0,1)$ turns out to be the key measure of quality of the hash function (equivalently, the space partition). In particular, the query time becomes $O(n^\rho d)$, and the space becomes $O(n^{1+\rho} + nd)$. For the above LSH family for the Hamming space, one can prove that $\rho = 1/c$, resulting in the [HIM12] algorithm with $O(n^{1/c}d)$ time and $O(n^{1+1/c} + nd)$ space.

A somewhat similar hashing-based algorithm (for the closest-pair problem) was earlier proposed in [KWZ95], and also in [Bro98]. Due to different problem formulation and analysis, comparing their performance with the guarantees of the LSH approach seems difficult.

Time-space trade-offs. While the LSH approach achieves sub-quadratic space, one may hope to obtain even better guarantee: a near-linear space, $\tilde{O}(nd)$. Indeed, such near-linear space algorithms have been proposed in [Ind01a, Pan06, AI06, Kap15, Laa15]. For the smallest possible space of $O(nd)$, [Kap15] obtains query time $O(n^{4/(c+1)}d)$. For other algorithms, see the "low space" regime in Table 43.1.1, as well as the "(very) low query time" regime for the opposite extreme, of the lowest possible query time.

Algorithms for l_2 and other ℓ_p norms. To solve the problem under the ℓ_1 norm, we can reduce it to the Hamming case. If we assume that all points of interest p have coordinates in the range $\{1, \dots, M\}$, we can define $U(p) = (U(p_1), \dots, U(p_d))$ where $U(x)$ is a string of x ones followed by $M - x$ zeros. Then we get $\|p - q\|_1 = D(U(p), U(q))$. In general, M could be quite large, but can be reduced to $d^{O(1)}$ in the context of approximate near neighbor [HIM12]. Thus we can reduce (r, c)-NN under l_1 to (r, c)-NN in the Hamming space.

For the Euclidean space ℓ_2, one can design more efficient algorithms. Specifically, [AI06] obtain $\rho = 1/c^2 + o(1)$, improving over the (optimal) $\rho = 1/c$ exponent for the Hamming space. For ℓ_p's, for $p \in [1, 2)$, one can then reduce the problem to the ℓ_2 case by embedding the $(p/2)$-th root of ℓ_p into ℓ_2 [Kal08, Theorem 4.1] (see also a quantitative version in [Ngu14]).

For ℓ_p with $p < 1$, [DIIM04] obtain an LSH exponent $\rho = 1/c^p + o(1)$.

Las Vegas algorithms. The standard LSH scheme guarantees 90% success probability of recovering an (approximate) near neighbor (Monte Carlo randomness). There are algorithms that guarantee to return an (approximate) near neighbor, albeit the runtime is in expectation only (Las Vegas randomness) [GPY94, Ind00, AGK06, Pag16]. Such algorithms proceed by constructing a number of space partitions, such that any pair of close points will collide in at least on of the space partitions. The query time of such algorithms is usually (polynomially) higher than the query time of the Monte Carlo ones.

Preliminary version (July 20, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

Lower bounds. For algorithms based on the LSH concept, we can prove tight lower bounds, ruling out better exponents ρ [MNP07, OWZ14]. In particular, for $p \in (0, 2]$, the tight exponent is $\rho = 1/c^p \pm o(1)$. Such lower bounds assume that P_1 is not too small, namely inversely exponential in $d^{\Omega(1)}$. LSH schemes with P_1 exponentially small in d is not useful for the high dimensional spaces with $d \gg \log n$ — in this situation, $1/P_1 \ll 1/n$. We note however that in the "moderate dimension regime," when $d = \Theta(\log n)$, it may be tolerable to have $P_1 = 2^{-\Theta(d)}$ and indeed, [BDGL16, Laa15] obtain somewhat better exponents ρ in this case.)

For $p > 2$, it is natural to conjecture that any LSH scheme must incur either a super-constant factor approximation, or must have P_1 exponentially small in $d^{\Omega(1)}$; see [AN11] for partial progress towards the conjecture.

DATA-DEPENDENT APPROACH

The data-independent techniques from the previous section have natural limitations. It turns out that sometimes vastly better algorithms are possible using *data-dependent* approaches. In particular, these are the methods where the hash function h itself depends on the entire dataset P. Note that an important requirement for a data-dependent hashing function is to have *efficient evaluation procedure* on a new (query) point q. Without such condition, the obvious best data-dependent partition would be the Voronoi diagram—i.e., $h(q)$ returns the identity of the closest point from the dataset—which is obviously useless (computing the hash function is as hard as the original problem!). Indeed, the space partitions mentioned below are (provably) better than the data-independent variants while being efficient to compute as well.

As before, the resulting algorithms give an improvement for worst-case datasets.

Algorithm for the ℓ_∞ norm. The first algorithm from this category is for near-neighbor problem under the ℓ_∞ norm, where data-independent methods are otherwise powerless. In particular, [Ind01b] solves (r, c)-NN under the l_∞^d norm with the following guarantees, for any $\rho > 0$:

- Approximation factor: $c = O(\lfloor \log_{1+\rho} \log 4d \rfloor)$; if $\rho = \log d$ then $c = 3$.
- Space: $dn^{1+\rho}$.
- Query time: $O(d \log n)$ for the static, or $(d + \log n)^{O(1)}$ for the dynamic case.
- Update time: $d^{O(1)} n^\rho$ (described in [Ind01a]).

The basic idea of the algorithm is to use a divide and conquer approach. In particular, consider hyperplanes H consisting of all points with one (say the ith) coordinate equal to the same value. The algorithm tries to find a hyperplane H having the property that the set of points $P_L \subset P$ that are on the left side of H and at distance $\geq r$ from H, is not "much smaller" than the set P_M of points at distance r from H. Moreover, a similar condition has to be satisfied for an analogously defined set P_R of points on the right side of H. If such H exists, we divide P into $P_{LM} = P_L \cup P_M$ and set $P_{RM} = P \setminus P_L$ and build the data structure recursively on P_{LM} and P_{RM}. It is easy to see that while processing a query q, it suffices to recurse on either P_{LM} or P_{RM}, depending on the side of H the query q lies on. Also, one can prove that the increase in storage caused by duplicating P_M is moderate. On the other hand, if H does not exist, one can prove that a large

Preliminary version (July 20, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

subset $P' \subset P$ has $O(cr)$ diameter. In such a case we can pick any point from P' as its representative, and apply the algorithm recursively on $P \setminus P'$.

Algorithms for the ℓ_p norms. Later work defined data-dependent hashing more formally and showed that one can obtain better runtime exponents ρ than LSH, for the exponent ρ defined similarly to the one from LSH. In particular, data-dependent hashing gives the following performance: for the Hamming space, the exponent is $\rho = \frac{1}{2c-1}$, and for the Euclidean space, the exponent is $\rho = \frac{1}{2c^2-1}$. See Table 43.2.1 for a comparison with LSH and lower bounds.

The data-dependent hashing algorithms in [AINR14, AR15] have two major components. First, if the given dataset has a certain "canonical geometric configuration," then one can design a data-*independent* hashing (LSH) scheme with better parameters than for datasets in a general position. This canonical setting essentially corresponds to a dataset which is distributed (pseudo-)randomly on a sphere (i.e., points are at $\approx \pi/2$ angle with respect to the origin), and the query is planted to be at $\theta < \pi/2$ angle from some point in the dataset. Second, there is a procedure to decompose any *worst-case* dataset and reduce it to this canonical case.

For ℓ_p, where $0 < p < 2$, all algorithms for ℓ_2 apply as well (with c^2 replaced by c^p in the exponent ρ); see [Ngu14].

For ℓ_p, where $p > 2$, efficient algorithms are possible via data-dependent hashing, however optimal bounds are not presently known. [And09] shows how to obtain $O(\log \log d)$ approximation, and [BG15, NR06] obtain approximation $2^{O(p)}$.[2]

Lower Bounds. One can prove that, for $p \leq 2$ and $p = \infty$, the above exponents ρ are optimal within the class of data-dependent hashing schemes. To prove such lower bounds, one has to formalize the class of data-dependent hashing schemes (which in particular would rule out the aforementioned Voronoi diagram solution). For ℓ_∞, matching lower bounds were shown in [ACP08, KP12], and for ℓ_p in [AR16]. Both type of results formalize the class by assuming that the hash function has description complexity of $n^{1-\Omega(1)}$, as well as that $d = \log^{1+\Omega(1)} n$ and P_1 is not too small. These lower bounds also have implication for (unconditional) cell probe lower bounds; see Section 43.3.

Time-space trade-offs. As for LSH, one can obtain other time-space trade-offs with the data-dependent approach as well. In particular, [Laa15, ALRW17] obtain trade-offs for the ℓ_2 space. One can obtain an algorithm with query time $n^{\rho_q + o(1)}d$ and space $n^{1+\rho_s+o(1)} + O(nd)$ for any $\rho_s, \rho_q > 0$ that satisfy the following equality:

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

This trade-off is also optimal (in the right formalization) [ALRW17, Chr16]. For $\rho_q = 0$, there are also cell-probe lower bounds; see Section 43.3.

GLOSSARY

Product metrics: An f-product of metrics $X_1, \ldots, X_k$ with distance functions $D_1, \ldots, D_k$ is a metric over $X_1 \times \ldots \times X_k$ with distance function D such that $D((p_1, \ldots, p_k), (q_1, \ldots, q_k)) = f(D_1(p_1, q_1), \ldots, D_k(p_k, q_k))$.

[2] At the moment of writing of this document, [NR06] does not seem to be available but is referenced in [Nao14, Remark 4.12].

Preliminary version (July 20, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

Although the l_∞ data structure seems to rely on the geometry of the l_∞ norm, it turns out that it can be used in a much more general setting. In particular, assume that we are given k metrics $M_1 \dots M_k$ such that for each metric M_i we have a data structure for (a variant of) (r, c)-NN in metric M_i, with $Q(n)$ query time and $S(n)$ space. In this setting, it is possible to construct a data structure solving $(r, O(c \log \log n))$-NN in the max-product metric M of $M_1, \dots, M_k$ (i.e., an f-product with f computing the maximum of its arguments) [Ind02]. The data structure for M achieves query time roughly $O(Q(n) \log n + k \log n)$ and space $O(kS(n)n^{1+\delta})$, for any constant $\delta > 0$. The data structure could be viewed as an abstract version of the data structure for the l_∞ norm (note that the l_∞^d norm is a max-product of l_p^1 norms). For the particular case of the l_∞^d norm, it is easy to verify that the result of [Ind02] provides a $O(\log \log n)$-approximate algorithm using space polynomial in n. At the same time, the algorithm of [Ind01b] has $O(\log \log d)$-approximation guarantee when using the same amount of space. Interestingly, the former data structure gives an approximation bound comparable to the latter one, while being applicable in a much more general setting.

The above result can also be used for developing NNS under product spaces, defined as follows. For a vector $x \in \mathbb{R}^{d_1 \cdot d_2}$, its $\ell_p^{d_1}(\ell_q^{d_2})$ norm is computed by taking the ℓ_q norm of each of the d_1 rows and then taking the ℓ_p norm of these d_1 values. For such product norms (in fact for any fixed iterated product norm), [AIK09, And09] showed how to obtain efficient NNS with $O(\log \log n)^{O(1)}$ approximation.

EXTENSIONS VIA EMBEDDINGS

Most of the algorithms described so far work only for l_p norms. However, they can be used for other metric spaces M, by using low-distortion embeddings of M into l_p norms. See Chapter 8 for more information.

Similarly, one can use low-distortion embedding of M into a product space to obtain efficient NNS under M. This has been used for the Ulam metric, which is the edit (Levenshtein) distance on nonrepetitive strings. In particular, [AIK09] showed a $O(1)$-distortion embedding of Ulam distance into an iterated product space, which gives the currently best known NNS algorithms for Ulam.

43.2 REDUCTIONS TO APPROXIMATE NEAR NEIGHBOR

GLOSSARY

We define the following problems, for a given set of points P in a metric space $M = (X, D)$:

Approximate Closest Pair, or c-CP: Find a pair of points $p', q' \in P$ such that $D(p', q') \leq c \min_{p,q \in P, p \neq q} D(p, q)$

Approximate Close Pair, or (r, c)-CP: If there exists $p, q \in P, p \neq q$, such that $D(p, q) \leq r$, find a pair $p', q' \in P, p' \neq q'$, such that $D(q', p') \leq cr$.

Approximate Chromatic Closest Pair, or c-CCP: Assume that each point $p \in P$ is labeled with a color $c(p)$. Find a pair of points p, q such that $c(p) \neq c(q)$ and $D(p, q)$ is approximately minimal (as in the definition of c-CP).

Preliminary version (July 20, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

Approximate Bichromatic Closest Pair, or c-BCP: As above, but $c(p)$ assumes only two values.

Approximate Chromatic/Bichromatic Close Pair, or (r,c)-CCP/(r,c)-BCP: Decision versions of c-CCP or c-BCP (as in the definition of (r,c)-CP).

Approximate Farthest Pair, or Diameter, or c-FP: Find $p, q \in P$ such that $D(p,q) \geq \max_{p',q' \in P} D(p',q')/c$. The decision problem, called ***Approximate Far Pair***, or (r,c)-FP, is defined in the natural way.

Approximate Farthest Neighbor, or c-FN: A maximization version of the Approximate Near Neighbor. The decision problem, called Approximate Far Neighbor or (r,c)-FN, is defined in a natural way.

Approximate Minimum Spanning Tree, or c-MST: Find a tree T spanning all points in P whose weight $w(T) = \sum_{(p,q)\in T} D(p,q)$ is at most c times larger than the weight of any tree spanning P.

Approximate Bottleneck Matching, or c-BM: Assuming $|P|$ is even, find a set of $|P|/2$ nonadjacent edges E joining points in P (i.e., a matching), such that the following function is minimized (up to factor of c)

$$\max_{\{p,q\}\in E} D(p,q)$$

Approximate Facility Location, or c-FL: Find a set $F \subset P$ such that the following function is minimized (up to factor of c), given the cost function $c : P \to \mathbb{R}^+$

$$\sum_{p\in F} c(p) + \sum_{p\in P} \min_{f\in F} D(p,f)$$

In general, we could have two sets: P_c of *cities* and P_f of *facilities*; in this case we require that $F \subset P_f$ and we are only interested in the cost of P_c.

Spread (of a point set): The ratio between the diameter of the set to the distance between its closest pair of points.

In this section we show that the problems defined above can be efficiently reduced to the approximate near-neighbor problem discussed in the previous section.

First, we observe that any problem from the above list, say $c(1+\delta)$-P for some $\delta > 0$, can be easily reduced to its decision version (say (r,c)-P), if we assume that the spread of $P \cup \{q\}$ is always bounded by some value, say Δ. For simplicity, assume that the minimum distance between the points in P is 1. The reduction proceeds by building (or maintaining) $O(\log_{1+\delta} \Delta)$ data structures for (r,c)-P, where r takes values $(1+\delta)^i/2$ for $i = 0, 1 \ldots$. It is not difficult to see that a query to $c(1+\delta)$-P can be answered by $O(\log\log_{1+\delta} \Delta)$ calls to these structures for (r,c)-P, via binary search.

In general, the spread of P could be unbounded. However, in many cases it is easy to reduce it to $n^{O(1)}$. This can be accomplished, for example, by "discretizing" the input to c-MST or c-FL. In those cases, the above reduction is very efficient.

Reductions from other problems are specified in the following table. The bounds for the time and space used by the algorithm in the "To" column are denoted by $T(n)$ and $S(n)$, respectively.

We mention that a few other reductions have been given in [KOR00, BOR04]. For the problems discussed in this section, they are less efficient than the reductions in the above table. Additionally, [BOR04] reduces the problems of computing

TABLE 43.2.1 Reductions to Approximate Near Neighbors.

#	FROM	TO	TIME	SPACE
1	Source: [HIM12].			
	$c(1+\delta)$-NN	(r,c)-NN	$T(n)\log^{O(1)} n$	$S(n)\log^{O(1)} n$
2	Source: [Epp95]; amortized time.			
	c-DBCP (r,c)-DBCP	c-DNN (r,c)-DNN	$T(n)\log^{O(1)} n$ $T(n)\log^{O(1)} n$	$S(n)\log^{O(1)} n$ $S(n)\log^{O(1)} n$
3	Source: [HIM12]; via Kruskal alg.			
	$c(1+\delta)$-MST	(r,c)-DBCP	$nT(n)\log^{O(1)} n$	
4	Source: [GIV01, Ind01a]; via Primal-Dual			
	$3c^3(1+\delta)$-FL	(r,c)-DBCP	$nT(n)\log^{O(1)} n$	
5	Source: [GIV01, Ind01a].			
	$2c$-BM	c-DBCP	$nT(n)\log^{O(1)} n$	

approximate agglomerative clustering and *sparse partitions* to $O(n\log^{O(1)} n)$ calls to a dynamic approximate nearest neighbor data structure. See [BOR04] for the definitions and algorithms.

Also, we mention that a reduction from $(1+\epsilon)$-approximate farthest neighbor to $(1+\epsilon)$-approximate nearest neighbor (for the static case and under the l_2 norm) has been given in [GIV01]. However, a direct (and dynamic) algorithm for the approximate farthest neighbor in l_2^d, achieving a better query and update times of $dn^{1/(1+\epsilon)^2}$, has been given in [Ind03]. The former paper also presents an algorithm for computing a $(\sqrt{2}+\epsilon)$-approximate diameter (for any $\epsilon > 0$) of a given pointset in $dn\log^{O(1)} n$ time.

We now describe briefly the main techniques used to achieve the above results.

Nearest neighbor. We start from the reduction of c-NN to (r,c)-NN. As we have seen already, the reduction is easy if the spread of P is small. Otherwise, it is shown that the data set can be clustered into $n/2$ clusters, in such a way that:

- If the query point q is "close" to one of the clusters, it must be far away from a constant fraction of points in P; thus, we can ignore these points in the search for an approximate nearest neighbor.
- If the query point q is "far" from a cluster, then all points in the clusters are equally good candidates for the *approximate* nearest neighbor; thus we can replace the cluster by its representative point.

See the details of the construction in [HIM12].

Bichromatic closest pair. A very powerful reduction from various variants of c-DBCP to c-DNN was given by Eppstein [Epp95]. His algorithm was originally designed for the case $c = 1$, but it can be verified to work also for general $c \geq 1$ [Epp99]. Moreover, as mentioned in the original paper, the reduction does not require the distance function D be a metric.

The basic idea of the algorithm is to try to maintain a graph that contains an edge connecting the two closest bichromatic points. A natural candidate for such a graph is the graph formed by connecting each point to its nearest neighbor. This,

Preliminary version (July 20, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

however, does not work, because a vertex in such a graph can have very high degree, leading to high update cost. Another option would be to maintain a single path, such that the ith vertex points to its nearest neighbor of the opposite color, chosen from points not yet included in the path. This graph has low degree, but its rigid structure makes it difficult to update it at each step. So the actual data structure is based on the path idea but allows its structure to degrade in a controlled way, and only rebuilds it when it gets too far degraded, so that the rebuilding work is spread over many updates. Then, however, one needs to keep track of the information from the degraded parts of the path, which can be done using a second shorter path, and so on. The constant factor reduction in the lengths of each successive path means the total number of paths is only logarithmic.

We note that recent research [Val15, KKK16] showed an alternative approach to the bichromatic closest pair problem, which, in certain settings, vastly outperforms the approach via the NNS algorithms. This line of work is best described using the following parameterization. Suppose the two color classes of points $A, B \subset \mathbb{R}^d$ satisfy the following: 1) all points are of unit norm, and 2) for each $a \in A, b \in B$, we have that $\langle a, b\rangle < \alpha$ except for a single (close) pair that satisfies $\langle a, b\rangle \geq \beta$, for some $0 < \alpha < \beta < 1$. Then the algorithm of [Val15] obtains a runtime of $n^{\frac{5-\omega}{4-\omega}+\omega\frac{\log\beta}{\log\alpha}} \cdot d^{O(1)}$, where $2 \leq \omega < 2.373$ is the exponent of the fast matrix multiplication algorithm. An algorithm in [KKK16] runs in time $n^{2\omega/3+O(\log\beta/\log\alpha)} \cdot d^{O(1)}$. The most interesting case is where the point sets A and B are random, except for a pair that has inner product ϵ (termed the *light bulb problem* [Val88]). In this case the runtime of these algorithms is of the form $n^{2-\Omega(1)}(d/\epsilon)^{O(1)}$, improving over $n^{2-O(\epsilon)}$ time obtainable via LSH methods. The algorithm of [Val15] also obtains $n^{2-\Omega(\sqrt{\epsilon})} \cdot d^{O(1)}$ time for the standard $(1+\epsilon)$-BCP.

Finally, there is recent work on *exact* BCP for medium dimensions, which also relies on faster matrix multiplication. In particular, the algorithm of [AW15] achieves a runtime of $n^{2-1/O(c\log^2 c)}$ for Hamming space of dimension $d = c\log n$ for $c > 1$.

Minimum spanning tree. Many existing algorithms for computing MST (e.g., Kruskal's algorithm) can be expressed as a sequence of operations on a CCP data structure. For example, Kruskal's algorithm repetitively seeks the lightest edge whose endpoints belong to different components, and then merges the components. These operations can be easily expressed as operations on a CCP data structure, where each component has a different color. The contribution of [HIM12] was to show that in case of Kruskal's algorithm, using an *approximate* c-CCP data structure enables one to compute an *approximate* c-MST. Also, note that c-CCP can be implemented by $\log n$ c-BCP data structures [HIM12]. Other reductions from c-MST to c-BCP are given in [BOR04, IST99].

Minimum bottleneck matching. The main observation here is that a matching is also a spanning forest with the property that any connected component has even cardinality (call it an *even* forest). At the same time, it is possible to convert *any* even forest to a matching, in a way that increases the length of the longest edge by at most a factor of 2. Thus, it suffices to find an even forest with minimum edge length. This can be done by including longer and longer edges to the graph, and stopping at the moment when all components have even cardinality. It is not difficult to implement this procedure as a sequence of c-CCP (or c-BCP) calls.

Preliminary version (July 20, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

Other algorithms. The algorithm for the remaining problem (c-FL) is obtained by implementing the primal-dual approximation algorithm [JV01]. Intuitively, the algorithm proceeds by maintaining a set of balls of increasing radii. The latter process can be implemented by resorting to c-CCP. The approximation factor follows from the analysis of the original algorithm.

43.3 LOWER BOUNDS

In the previous sections we presented many algorithms solving approximate versions of proximity problems. The main motivation for designing approximation algorithms was the "curse of dimensionality" conjecture, i.e., the conjecture that finding exact solutions to those problems requires either superpolynomial (in d) query time, or superpolynomial (in n) space. In this section we state the conjecture more rigorously, and describe the progress toward proving it. We also describe the lower bounds for the approximation algorithms as well, which sometimes match the algorithmic results presented earlier.

Curse of dimensionality. We start from the exact near-neighbor problem, where the curse of dimensionality can be formalized as follows.

CONJECTURE 43.3.1

Assume that $d = n^{o(1)}$ but $d = \omega(\log n)$. Any data structure for $(r, 1)$-NN in Hamming space over $\{0,1\}^d$, with $d^{O(1)}$ query time, must use $n^{\omega(1)}$ space.

The conjecture as stated above is probably the weakest version of the "curse of dimensionality" phenomenon for the near-neighbor problem. It is plausible that other (stronger) versions of the conjecture could hold. In particular, at present, we do not know any data structure that simultaneously achieves $o(dn)$ query time and $2^{o(d)}$ space for the above range of d. At the same time, achieving $O(dn)$ query time with space dn, or $O(d)$ query time with space 2^d is quite simple (via linear scan or using exhaustive storage).

Also note that if $d = O(\log n)$, achieving $2^{o(d)} = o(n)$ space is impossible via a simple incompressibility argument.

Below we describe the work toward proving the conjecture. The first results addressed the complexity of a simpler problem, namely the *partial match* problem. This problem is of importance in databases and other areas and has been long investigated (e.g., see [Riv74]). Thus, the lower bounds for this problem are interesting in their own right.

GLOSSARY

Partial match: Given a set P of n vectors from $\{0,1\}^d$, design a data structure that supports the following operation: For any query $q \in \{0,1,*\}^d$, check if there exists $p \in P$ such that for all $i = 1 \ldots d$, if $q_i \neq *$ then $p_i = q_i$.

It is not difficult to see that any data structure solving $(r, 1)$-NN in the Hamming metric $\{0,1\}^d$, can be used to solve the partial match problem using essentially the same space and query time. Thus, any lower bound for the partial match prob-

Preliminary version (July 20, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

lem implies a corresponding lower bound for the near-neighbor problem. The best currently known lower bound for the partial match has been established in [Păt10], following earlier work of [JKKR04, BOR03, MNSW98]. Their lower bound holds in the *cell-probe* model, a very general model of computation, capturing e.g., the standard Random Access Machine model. The lower bound implies that any (possibly randomized) cell-probe algorithm for the partial match problem, in which the algorithm is allowed to retrieve at most $O(n^{1-\epsilon})$ bits from any memory cell in one step for $\epsilon > 0$, must use space $2^{\Omega(d/t)}$ for t cell probes (memory accesses, and hence query time).

Similar space lower bounds have been proven for *exact* (possibly randomized) near-neighbor [BR02], and for *deterministic* $O(1)$-approximate near neighbor [Liu04]. Note that allowing both approximation and randomization allows for much better upper bounds (see the "very low query time" regime in Table 43.1.1).

All the aforementioned lower bounds are proved in a very general model, using the tools of *asymmetric communication complexity*. As a result, they cannot yield lower bounds of $\omega(d/\log n)$ query time, for $n^{O(1)}$ space, as we now explain.

The communication complexity approach interprets the data structure as a communication channel between Alice (holding the query point q) and Bob (holding the database P). The goal of the communication is to determine the nearest neighbor of q. Since the data structure has polynomial size, each access to one of its memory cell is equivalent to Alice sending $O(\log n)$ bits of information to Bob. If we show that Alice needs to send at least a bits to Bob to solve the problem, we obtain $\Omega(a/\log n)$ lower bound for the query time. However, $b \leq d$, since Alice can always choose to transmit the whole input vector q. Thus, $\Omega(d/\log n)$ lower bound is the best result one can achieve using the communication complexity approach.

A partial step toward removing this obstacle was made in [BV02], employing the *branching programs* model of computation. In particular, they focused on randomized algorithms that have very small (inversely polynomial in n) probability of error. They showed that any algorithm for the $(r, 1)$-NN problem in the Hamming metric over $\{1 \ldots d^6\}^d$ has either $\Omega(d \log(d \log d/S))$ query time or uses $\Omega(S)$ space. This holds for $n = \Omega(d^6)$. Thus, if the query time is $o(d \log d)$, then the data structure must use $2^{d^{\Omega(1)}}$ space.

Another such step was made in [PT09], who show higher lower bound for *near-linear* space. In particular, they show that number of cell probes must be $t = \Omega(d \cdot \log \frac{Sd}{n})$ for space S.

Lower bounds for approximate randomized algorithms. More recent efforts focused on lower bounds for approximate randomized problem, where most of the improved algorithms were obtained. Even if it is harder to prove lower bounds in this regime, researchers often managed to prove *tight* lower bounds, matching the known algorithms.

First of all, if we consider the *nearest neighbor* problem (in contrast to the *near* neighbor, which is considered in this chapter), [CR10, CCGL03] show that any data structure for the $O(1)$-approximate nearest neighbor on $\{0, 1\}^d$ requires either $\Omega(\log \log d/ \log \log \log d)$ query time or $n^{\omega(1)}$ space. [CR10] also show a matching upper bound.

For the *near* neighbor problem, the authors of [AIP06] show that a $(1 + \epsilon)$-approximation requires $\left(n^{\Omega(1/\epsilon^2)}\right)^{1/t}$ space lower bound for t cell-probe data structures for the Hamming space. This lower bound matches the upper bound of

[KOR00, HIM12] (the "very low query time" regime in Table 43.1.1), which uses just 1 cell probe. We note that while the lower bound of [AIP06] is tight up to a constant in the exponent, a tight constant was shown in [ALRW17], although the matching upper bound uses $n^{o(1)}$ cell probes.

Restricting the attention to *near-linear space* algorithms, higher lower bounds were shown. In particular, the authors of [PTW08, PTW10] show how to obtain the following lower bounds: any t cell probe data structure for (r, c)-NNS under ℓ_1 requires space $n^{1+\Omega(1/c)/t}$. Note that this lower bound goes beyond the simple communication complexity framework described above. The lower bounds from [PTW10] were tightened in [ALRW17], which, when setting $t \in \{1, 2\}$, match the space bound of the data structure of [Laa15, ALRW17] for $n^{o(1)}$ query time.

Finally, for NNS under the ℓ_∞ metric, there are similar lower bounds, which are also based on the tools of asymmetric communication complexity. The authors of [ACP08] show that any deterministic decision tree for ℓ_∞ must incur an approximation of $O(\log_{1+\rho} \log n)$ if using space $n^{1+\rho}$ (unless the query time is polynomially large). This matches the upper bound of [Ind01b] described above. This lower bound was also extended to randomized decision trees in [PTW10, KP12].

REDUCTIONS

Despite the progress toward resolving the "curse of dimensionality" conjecture and the widespread belief in its validity, proving it seems currently beyond reach. Nevertheless, it is natural to assume the validity of the conjecture (or its variants), and see what conclusions can be derived from this assumption. Below we survey a few results of this type.

In order to describe the results, we need to state another conjecture.

CONJECTURE 43.3.2

Let $d = n^{o(1)}$ but $d = \log^{\omega(1)} n$. Any data structure for the partial match problem with parameters d and n that provides $d^{O(1)}$ query time must use $2^{d^{\Omega(1)}}$ space.

Note that, for the same ranges[3] of d, Conjecture 43.3.2 is analogous to Conjecture 43.3.1, but much stronger: it considers an easier problem, and states stronger bounds. However, since the partial match problem was extensively investigated on its own, and no algorithm with bounds remotely resembling the above have been discovered (cf. [CIP02] for a survey), Conjecture 43.3.2 is believed to be true.

Assuming Conjecture 43.3.2, it is possible to show lower bounds for some of the approximate nearest neighbor problems discussed in Section 43.1. In particular, it was shown [Ind01b] that any data structure for (r, c)-NN under l_∞^d for $c < 3$ can be used to solve the partial match problem with parameter d, using essentially the same query time and storage (the number of points in the database is the same in both cases). Thus, unless Conjecture 43.3.2 is false, the 3-approximation algorithm from Section 43.1 is optimal, in the sense that it provides the smallest approximation factor possible while preserving polynomial (in d) query time and subexponential (in d) storage. Note that this result resembles the nonapproximability results based on the P $\neq$ NP conjecture.

On the other hand, it was shown [CIP02] that the exact near-neighbor prob-

[3]For $d = \log^{\omega(1)} n$, Conjecture 43.3.2 is true by a simple incompressibility argument. At the same time, the status of Conjecture 43.3.1 for $d \in [\omega(\log n), \log^{O(1)} n]$ is still unresolved.

Preliminary version (July 20, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

lem under the l_∞^k norm can be reduced to solving the partial match problem with the parameter $d = (k + \log n)^{O(1)}$; the number of points n is the same for both problems. In fact, the same holds for a more general problem of *orthogonal range queries.* Thus, Conjecture 43.3.2, and its variant for the $(r, 1)$-NN under l_∞^d (or for orthogonal range queries), are equivalent. This strengthens the belief in the validity of Conjecture 43.3.2, since the exact nearest neighbor under l_∞ norm and the orthogonal range query problem have received additional attention in the Computational Geometry community.

43.4 OTHER TOPICS

There are many other research directions on high-dimensional NNS; we briefly mention some of them below.

Average-case algorithms. The approximate algorithms described so far are designed to work for any (i.e., worst-case) input. However, researchers have also investigated *exact* algorithms for the NN problem that achieve fast query times for *average* input.

In fact, the first such average-case NNS solutions have been proposed for its offline version (the bichromatic closest pair from above). In particular, [Val88] introduced the *light-bulb problem*, defined as follows: the point set P is chosen at random i.i.d. from $\{0, 1\}^d$, and a point q is inserted at random within distance r from some point $p \in P$; the goal is to find q given $P \cup \{q\}$. Some results were obtained in [PRR95, GPY94] (with performance comparable to LSH [HIM12]), and [Dub10] (with performance comparable to data-dependent LSH [AR15]), as well as in [Val15, KKK16] which use fast matrix multiplication (see Section 43.2).

For ℓ_∞^d, there also are average-case algorithms for the NNS problem. Consider a point set where each point, including the query, is chosen randomly i.i.d. from $[0, 1]^d$. In this setting, it was shown [AHL01, HL01] that there is a nearest neighbor data structure using $O(dn)$ space, with query time $O(n \log d)$. Note that a naive algorithm would suffer from $O(nd)$ query time. The algorithm uses a clever pruning approach to quickly eliminate points that *cannot* be nearest neighbors of the query point.

Low-intrinsic dimension. Another way to depart from the worst-case analysis is to assume further structure in the dataset P. A particularly lucrative approach is to assume that the dataset has "low intrinsic dimension." This is justified by the fact that the high-dimensional data is often actually explained by a few free parameters, and the coordinate values are functions of these parameters. One such notion is the *doubling dimension* of a pointset P, defined as follows. Let λ be the smallest number such that any ball of radius r in P can be covered by λ balls of radius $r/2$ (with centers in P), for any $r > 0$. Then we say that the pointset P has doubling dimension $\log \lambda$. This definition was introduced in [Cla99, GKL03], and is related to the Assouad constant [Ass83]. Note that it can be defined for any metric space on P.

Algorithms for datasets P with intrinsic dimension k typically achieve performance comparable to NNS algorithms for k-dimensional spaces. Some algorithms designed for low doubling dimensional spaces include [KL04, BKL06, IN07].

Preliminary version (July 20, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

There are now a few notions of "intrinsic dimension," including KR-dimension [KR02], smooth manifolds [BW09, Cla08], and others [CNBYM01, FK97, IN07, Cla06, DF08, AAKK14].

NNS for higher dimensional flats. Finally, a further line of work investigated NNS in a setting where the dataset points or the queries are in fact larger objects, such as lines, or, more generally, k-dimensional flats. The new goal becomes to find, say, the dataset point closest to a given *query line* (up to some approximation). For the case when the dataset is composed of points in $\mathbb{R}^d$ and the queries are lines, the results of [AIKN09, MNSS15] provide efficient algorithms, with polynomial space and sublinear query time. Such algorithms often use vanilla approximate NNS algorithms as subroutines. For the dual case, where the dataset is composed of lines and the queries are points, the work of [Mah15] provides an efficient algorithm (improving over the work of [Mag07, BHZ07]). Some of the cited results generalize lines to k-dimensional flats, with performance degrading (rapidly) with the parameter k.

We note that the problem is intrinsically related to the partial match problem, since a query with k "don't care" symbols can be represented as a k-dimensional flat query.

43.5 SOURCES AND RELATED MATERIAL

RELATED CHAPTERS

Chapter 8: Low-distortion embeddings of discrete metric spaces
Chapter 28: Arrangements
Chapter 32: Proximity algorithms
Chapter 40: Range searching

REFERENCES

[AAKK14] A. Abdullah, A. Andoni, R. Kannan, and R. Krauthgamer. Spectral approaches to nearest neighbor search. In *Proc. 55th IEEE Sympos. Found. Comp. Sci.*, pages 581–590, 2014. Full version at `arXiv:1408.0751`.

[ACP08] A. Andoni, D. Croitoru, and M. Pătraşcu. Hardness of nearest neighbor under L-infinity. In *Proc. 49th IEEE Sympos. Found. Comp. Sci.*, pages 424–433, 2008.

[ADFM11] S. Arya, G.D. Da Fonseca, and D.M. Mount. Approximate polytope membership queries. In *Proc. 43rd ACM Sympos. Theory Comput.*, pages 579–586, 2011. Full version at `arXiv:1604.01183`.

[AGK06] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins. In *Proc. 32nd Conf. Very Large Data Bases*, pages 918–929, 2006.

[AHL01] H. Alt and L. Heinrich-Litan. Exact l_∞-nearest neighbor search in high dimensions. *Proc. 17th Sympos. Comput. Geom.*, pages 157–163, ACM Press, 2001.

[AI06] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest

Preliminary version (July 20, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

neighbor in high dimensions. In *Proc. 47th IEEE Sympos. Found. Comp. Sci.*, pages 459–468, 2006.

[AIK09] A. Andoni, P. Indyk, and R. Krauthgamer. Overcoming the ℓ_1 non-embeddability barrier: Algorithms for product metrics. In *Proc. 20th ACM-SIAM Sympos. Discrete Algorithms*, pages 865–874, 2009.

[AIKN09] A. Andoni, P. Indyk, R. Krauthgamer, and H.L. Nguyen. Approximate line nearest neigbor in high dimensions. In *Proc. 20th ACM-SIAM Sympos. Discrete Algorithms*, pages 293–301, 2009.

[AINR14] A. Andoni, P. Indyk, H.L. Nguyen, and I. Razenshteyn. Beyond locality-sensitive hashing. In *Proc. 25th ACM-SIAM Sympos. Discrete Algorithms*, pages 1018–1028, 2014.

[AIP06] A. Andoni, P. Indyk, and M. Pătraşcu. On the optimality of the dimensionality reduction method. In *Proc. 47th IEEE Sympos. Found. Comp. Sci.*, pages 449–458, 2006.

[ALRW17] A. Andoni, T. Laarhoven, I.P. Razenshteyn, and E. Waingarten. Optimal hashing-based time-space trade-offs for approximate near neighbors. In *Proc. 28th ACM-SIAM Sympos. Discrete Algorithms*, pages, 47–66, 2017.

[AMM09] S. Arya, T. Malamatos, and D.M. Mount. Space-time tradeoffs for approximate nearest neighbor searching. *J. ACM*, 57:1, 2009.

[AMN+98] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *J. ACM*, 6:891–923, 1998.

[AN11] A. Andoni and H. Nguyen. Lower bounds for locality sensitive hashing. Manuscript, 2011.

[And09] A. Andoni. *Nearest Neighbor Search: the Old, the New, and the Impossible.* PhD thesis, MIT, Cambridge, 2009.

[AR15] A. Andoni and I. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proc. 47th ACM Sympos. Theory Comput.*, pages 793–801, 2015. Full version available at `arXiv:1501.01062`.

[AR16] A. Andoni and I. Razenshteyn. Tight lower bounds for data-dependent locality-sensitive hashing. In *Proc. Sympos. Comput. Geom.*, vol 51 of *LIPIcs*, pages 9:1-9:11, Schloss Dagstuhl, 2016.

[Ass83] P. Assouad. Plongements lipschitziens dans $\mathbb{R}^n$. *Bull. Soc. Math. France*, 111:429–448, 1983.

[AW15] J. Alman and R. Williams. Probabilistic polynomials and Hamming nearest neighbors. In *Proc. 56th IEEE Sympos. Found. Comp. Sci.*, pages 136–150, 2015.

[BDGL16] A. Becker, L. Ducas, N. Gama, and T. Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *Proc. 27th ACM-SIAM Sympos. Discrete Algorithms*, pages 10–24, 2016.

[BG15] Y. Bartal and L.-A. Gottlieb. Approximate nearest neighbor search for ℓ_p-spaces $(2 < p < \infty)$ via embeddings. Preprint, `arXiv:1512.01775`, 2015.

[BHZ07] R. Basri, T. Hassner, and L. Zelnik-Manor. Approximate nearest subspace search with applications to pattern recognition. In *Proc. IEEE Conf. Computer Vision Pattern Recogn.*, pages 1–8, 2007.

[BKL06] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proc. 23rd Internat. Conf. Machine Learning*, pages 97–104, ACM Press, 2006.

[BOR03] A. Borodin, R. Ostrovsky, and Y. Rabani. Lower bounds for high dimensional nearest neighbor search and related problems. In *Discrete and Computational Geometry: The Goodman-Pollack Festschrift*, pages 253–274, Springer, Berlin, 2003.

[BOR04] A. Borodin, R. Ostrovsky, and Y. Rabani. Subquadratic approximation algorithms for clustering problems in high dimensional spaces. *Mach. Learn.*, 56:153–167, 2004.

[BR02] O. Barkol and Y. Rabani. Tighter bounds for nearest neighbor search and related problems in the cell probe model. *J. Comput. Syst. Sci.*, 64:873–896, 2002.

[Bro98] A. Broder. Filtering near-duplicate documents. *Proc. Fun with Algorithms*, Carleton University, 1998.

[BV02] P. Beame and E. Vee. Time-space tradeoffs, multiparty communication complexity, and nearest-neighbor problems. *Proc. 34th ACM Sympos. Theory Comput.*, pages 688–697, 2002.

[BW09] R.G. Baraniuk and M.B. Wakin. Random projections of smooth manifolds. *Found. Comput. Math.*, 9:51–77, 2009.

[CCGL03] A. Chakrabarti, B. Chazelle, B. Gum, and A. Lvov. A lower bound on the complexity of approximate nearest-neighbor searching on the Hamming cube. In *Discrete and Computational Geometry: The Goodman-Pollack Festschrift*, pages 313–328, Springer, Berlin, 2003.

[Cha02a] T.M. Chan. Closest-point problems simplified on the RAM. In *Proc. 13th ACM-SIAM Sympos. Discrete Algorithms*, pages 472–473, 2002.

[Cha02b] M.S. Charikar. Similarity estimation techniques from rounding. In *Proc. 34th ACM Sympos. Theory Comput.*, pages 380–388, 2002.

[Chr17] T. Christiani. A framework for similarity search with space-time tradeoffs using locality-sensitive filtering. In *Proc. 20th ACM-SIAM Sympos. Discrete Algorithms*, pages 31–46, 2017.

[CIP02] M. Charikar, P. Indyk, and R. Panigrahy. New algorithms for subset query, partial match, orthogonal range searching and related problems. *Proc. Internat. Coll. Automata, Languages, Progr.*, vol. 2380 of *LNCS*, pages 451–462, Springer, Berlin, 2002.

[Cla88] K.L. Clarkson. A randomized algorithm for closest-point queries. *SIAM J. Comput.*, 17:830–847, 1988.

[Cla94] K.L. Clarkson. An algorithm for approximate closest-point queries. *Proc. 10th Sympos. Comput. Geom.*, pages 160–164, ACM Press, 1994.

[Cla99] K.L. Clarkson. Nearest neighbor queries in metric spaces. *Discrete Comput. Geom.*, 22:63–93, 1999.

[Cla06] K.L. Clarkson. Nearest-neighbor searching and metric space dimensions. In G. Shakhnarovich, T. Darrell, and P. Indyk, editors, *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, pages 15–59, MIT Press, Cambridge, 2006.

[Cla08] K.L. Clarkson. Tighter bounds for random projections of manifolds. In *Proc. 24th Sympos. Comput. Geom.*, pages 39–48, ACM Press, 2008.

[CNBYM01] E. Chávez, G. Navarro, R. Baeza-Yates, and J. Marroquin. Searching in metric spaces. *ACM Comput. Surv*, 33:273–321, 2001.

[CR10] A. Chakrabarti and O. Regev. An optimal randomised cell probe lower bound for approximate nearest neighbor searching. *SIAM J. Comput.*, 39:1919–1940, 2010.

Preliminary version (July 20, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

[DF08] S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. In *Proc. 40th ACM Sympos. Theory Comput.*, pages 537–546, 2008.

[DGK01] C.A. Duncan, M.T. Goodrich, and S. Kobourov. Balanced aspect ratio trees: Combining the advantages of k-d trees and octrees. *J. Algorithms*, 38:303–333, 2001.

[DIIM04] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. 20th Sympos. Comput. Geom.*, pages 253–262, ACM Press, 2004.

[Dub10] M. Dubiner. Bucketing coding and information theory for the statistical high dimensional nearest neighbor problem. *IEEE Trans. Inform. Theory*, 56:4166–4179, 2010.

[DW82] L. Devroye and T. Wagner. Nearest neighbor methods in discrimination. P.R. Krishnaiah and L.N. Kanal, editors, *Handbook of Statistics*, vol. 2, North-Holland, Amsterdam, 1982.

[Epp95] D. Eppstein. Dynamic Euclidean minimum spanning trees and extrema of binary functions. *Discrete Comput. Geom.*, 13:111–122, 1995.

[Epp99] D. Eppstein. Personal communication. 1999.

[FK97] C. Faloutsos and I. Kamel. Relaxing the uniformity and independence assumptions using the concept of fractal dimension. *J. Comput. System Sci.*, 55:229–240, 1997.

[GG91] A. Gersho and R. Gray. *Vector Quantization and Data Compression.* Kluwer, Norwell, 1991.

[GIV01] A. Goel, P. Indyk, and K. Varadarajan. Reductions among high-dimensional geometric problems. *Proc. 12th ACM-SIAM Sympos. Discrete Algorithms*, pages 769–778, 2001.

[GKL03] A. Gupta, R. Krauthgamer, and J.R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Proc. 44th IEEE Sympos. Found. Comp. Sci.*, pages 534–543, 2003.

[GPY94] D. Greene, M. Parnas, and F. Yao. Multi-index hashing for information retrieval. In *Proc. 35th IEEE Sympos. Found. Comp. Sci.*, pages 722–731, 1994.

[HIM12] S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory Comput.*, 8:321–350, 2012.

[HL01] L. Heinrich-Litan. Exact l_∞-nearest neighbor search in high dimensions. Personal communication, 2001.

[HP01] S. Har-Peled. A replacement for Voronoi diagrams of near linear size. In *Proc. 42nd IEEE Sympos. Found. Comp. Sci.*, pages 94–103, 2001.

[IN07] P. Indyk and A. Naor. Nearest neighbor preserving embeddings. *ACM Trans. Algorithms*, 3:31, 2007.

[Ind00] P. Indyk. Dimensionality reduction techniques for proximity problems. *Proc. 11th ACM-SIAM Sympos. Discrete Algorithms*, pages 371–378, 2000.

[Ind01a] P. Indyk. *High-Dimensional Computational Geometry.* Ph.D. Thesis. Department of Computer Science, Stanford University, 2001.

[Ind01b] P. Indyk. On approximate nearest neighbors in ℓ_∞ norm. *J. Comput. System Sci.*, 63:627–638, 2001.

[Ind02] P. Indyk. Approximate nearest neighbor algorithms for Frechet metric via product metrics. *Proc. 18th Sympos. Comput. Geom.*, pages 102–106, ACM Press, 2002.

Preliminary version (July 20, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

[Ind03] P. Indyk. Better algorithms for high-dimensional proximity problems via asymmetric embeddings. *Proc. 14th ACM-SIAM Sympos. Discrete Algorithms*, pages 539–545, 2003.

[IST99] P. Indyk, S.E. Schmidt, and M. Thorup. On reducing approximate MST to closest pair problems in high dimensions. Manuscript, 1999.

[JKKR04] T.S. Jayram, S. Khot, R. Kumar, and Y. Rabani. Cell-probe lower bounds for the partial match problem. *J. Comput. Syst. Sci.*, 69:435–447, 2004.

[JV01] K. Jain and V.V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. *J. ACM*, 48:274–296, 2001.

[Kal08] N.J. Kalton. The nonlinear geometry of Banach spaces. *Rev. Mat. Complut.*, 21:7–60, 2008.

[Kap15] M. Kapralov. Smooth tradeoffs between insert and query complexity in nearest neighbor search. In *Proc. 34th ACM Sympos. Principles Database Syst.*, pages 329–342, 2015.

[KKK16] M. Karppa, P. Kaski, and J. Kohonen. A faster subquadratic algorithm for finding outlier correlations. In *Proc. 27th ACM-SIAM Sympos. Discrete Algorithms*, pages 1288–1305, 2016.

[KL04] R. Krauthgamer and J.R. Lee. Navigating nets: Simple algorithms for proximity search. *Proc. 15th ACM-SIAM Sympos. Discrete Algorithms*, pages 798–807, 2004.

[Kle97] J. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. *Proc. 29th ACM Sympos. Theory Comput.*, pages 599–608, 1997.

[KOR00] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.*, 30:457–474, 2000.

[KP12] M. Kapralov and R. Panigrahy. NNS lower bounds via metric expansion for ℓ_∞ and EMD. In *Proc. Internat. Coll. Automata, Languages and Progr.*, vol. 7391 of *LNCS*, pages 545–556, Springer, Berlin, 2012.

[KR02] D. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. *Proc. 34th ACM Sympos. Theory Comput.*, pages 741–750, 2002.

[KWZ95] R. Karp, O. Waarts, and G. Zweig. The bit vector intersection problem. In *Proc. 36th IEEE Sympos. Found. Comp. Sci.*, pages 621–630, 1995.

[Laa15] T. Laarhoven. Tradeoffs for nearest neighbors on the sphere. Preprint, `arXiv:1511.07527`, 2015.

[Liu04] D. Liu. A strong lower bound for approximate nearest neighbor searching in the cell probe model. *Inform. Process. Lett*, 92:23–29, 2004.

[LT80] R.J. Lipton and R.E. Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9:615–627, 1980.

[Mag07] A. Magen. Dimensionality reductions in ℓ_2 that preserve volumes and distance to affine spaces. *Discrete Comput. Geom.*, 38:139–153, 2007.

[Mah15] S. Mahabadi. Approximate nearest line search in high dimensions. In *Proc. 26th ACM-SIAM Sympos. Discrete Algorithms*, pages 337–354, 2015.

[Mei93] S. Meiser. Point location in arrangements of hyperplanes. *Inform. and Comput.*, 106:286–303, 1993.

[MNP07] R. Motwani, A. Naor, and R. Panigrahy. Lower bounds on locality sensitive hashing. *SIAM J. Discrete Math.*, 21:930–935, 2007.

[MNSS15] W. Mulzer, H.L. Nguyên, P. Seiferth, and Y. Stein. Approximate k-flat nearest neighbor search. In *Proc. 47th ACM Sympos. Theory Comput.*, pages 783–792, 2015.

Preliminary version (July 20, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

[MNSW98] P. Miltersen, N. Nisan, S. Safra, and A. Wigderson. Data structures and asymmetric communication complexity. *J. Comput. Syst. Sci.*, 57:37–49, 1998.

[MP69] M. Minsky and S. Papert. *Perceptrons.* MIT Press, Cambridge, 1969.

[Nao14] A. Naor. Comparison of metric spectral gaps. *Anal. Geom. Metr. Spaces*, 2(1), 2014.

[Ngu14] H.L. Nguyen. *Algorithms for High Dimensional Data.* PhD thesis, Princeton University, 2014.

[NR06] A. Naor and Y. Rabani. On approximate nearest neighbor search in ℓ_p, $p > 2$. Manuscript, 2006.

[OWZ14] R. O'Donnell, Y. Wu, and Y. Zhou. Optimal lower bounds for locality sensitive hashing (except when q is tiny). *ACM Trans. Comput. Theory*, 6:5, 2014.

[Pag16] R. Pagh. Locality-sensitive hashing without false negatives. In *Proc. 27th ACM-SIAM Sympos. Discrete Algorithms*, pages 1–9, 2016.

[Pan06] R. Panigrahy. Entropy-based nearest neighbor algorithm in high dimensions. In *Proc. 17th ACM-SIAM Sympos. Discrete Algorithms*, pages 1186–1195, 2006.

[Păt10] M. Pătraşcu. Unifying the landscape of cell-probe lower bounds. In *SIAM J. Comput.*, 40:827–847, 2010.

[PRR95] R. Paturi, S. Rajasekaran, and J. Reif. The light bulb problem. *Inform. and Comput.*, 117:187–192, 1995.

[PT09] M. Pătraşcu and M. Thorup. Higher lower bounds for near-neighbor and further rich problems. *SIAM J. Comput.*, 39:730–741, 2009.

[PTW08] R. Panigrahy, K. Talwar, and U. Wieder. A geometric approach to lower bounds for approximate near-neighbor search and partial match. In *Proc. 49th IEEE Sympos. Found. Comp. Sci.*, pages 414–423, 2008.

[PTW10] R. Panigrahy, K. Talwar, and U. Wieder. Lower bounds on near neighbor search via metric expansion. In *Proc. 51st IEEE Sympos. Found. Comp. Sci.*, pages 805–814, 2010.

[Riv74] R.L. Rivest. *Analysis of Associative Retrieval Algorithms.* Ph.D. thesis, Stanford University, 1974.

[SH75] M.I. Shamos and D. Hoey. Closest point problems. *Proc. 16th IEEE Sympos. Found. Comp. Sci.*, pages 152–162, 1975.

[SSS06] Y. Sabharwal, N. Sharma, and S. Sen. Nearest neighbors search using point location in balls with applications to approximate Voronoi decompositions. *J. Comput. Syst. Sci.*, 72:955–977, 2006.

[Val88] L.G. Valiant. Functionality in neural nets. In *Proc. 1st Workshop Comput. Learning Theory*, pages 28–39, Morgan Kaufmann, San Francisco, 1988.

[Val15] G. Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *J. ACM*, 62:13, 2015.

[WSB98] R. Weber, H.J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. *Proc. 24th Conf. Very Large Data Bases*, pages 194 205, Morgan Kaufman, San Francisco, 1998.

Preliminary version (July 20, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.