The black-box complexity of nearest neighbor search*

Robert Krauthgamer[†] James R. Lee[‡]

January 25, 2005

Abstract

We define a natural notion of efficiency for approximate nearest-neighbor (ANN) search in general n-point metric spaces, namely the existence of a randomized algorithm which answers $(1+\varepsilon)$-approximate nearest neighbor queries in polylog(n) time using only polynomial space. We then study which families of metric spaces admit efficient ANN schemes in the black-box model, where only oracle access to the distance function is given, and any query consistent with the triangle inequality may be asked.

For $\varepsilon < \frac{2}{5}$, we offer a complete answer to this problem. Using the notion of metric dimension defined in [GKL03] (à la [Ass83]), we show that a metric space X admits an efficient $(1+\varepsilon)$-ANN scheme for any $\varepsilon < \frac{2}{5}$ *if and only if* $\dim(X) = O(\log\log n)$. For coarser approximations, clearly the upper bound continues to hold, but there is a threshold at which our lower bound breaks down—this is precisely when points in the "ambient space" may begin to affect the complexity of "hard" subspaces $S \subseteq X$. Indeed, we give examples which show that $\dim(X)$ does not characterize the black-box complexity of ANN above the threshold.

Our scheme for ANN in low-dimensional metric spaces is the first to yield efficient algorithms without relying on any additional assumptions on the input. In previous approaches (e.g., [Cla99, KR02, KL04, HKMR04]), even spaces with $\dim(X) = O(1)$ sometimes required $\Omega(n)$ query times.

1 Introduction

Nearest-neighbor search. Nearest-neighbor search (NNS) is the problem of preprocessing a set X of n points lying in a huge (possibly infinite) metric space (M, d) so that given a query $q \in M$, one can efficiently locate the nearest point to q among the points in X. Computing such nearest neighbors efficiently is a classical and fundamental problem with numerous practical applications. These include data compression, database queries, machine learning, computational biology, data mining, pattern recognition, and ad-hoc networks. A common feature of many of these examples is that comparing two elements is costly, hence the number of distance computations should be made as small as possible.

Most previous research has focused on the important special case when $M = \mathbb{R}^d$ and distances are computed according to some ℓ_p norm. While many types of data can be naturally represented in such a form, this is certainly not true for a significant number of applications, and it is therefore

*A preliminary version of this paper appeared in the 31st International Colloquium on Automata, Languages and Programming (ICALP 2004).

[†]IBM Almaden Research Center, 650 Harry Road, San Jose CA 95120, USA. Part of this work was done while this author was with the International Computer Science Institute and with the Computer Science Division of U.C. Berkeley. Email: `robi@almaden.ibm.com`

[‡]Computer Science Division, U.C. Berkeley, Berkeley, CA 94720, USA. Supported by NSF grant CCR-0121555 and an NSF Graduate Research Fellowship. Email: `jrl@cs.berkeley.edu`

desirable to address NNS in general metric spaces. On the other hand, data structures for general metrics might perform a nearest neighbor query in time as poorly as $\Omega(n)$ which is unacceptable in practice. Such a dependence is inherent even when only approximate solutions are required. A well-known example is where X forms a uniform metric, so that the interpoint distances in X are all equal, providing the preprocessing step with essentially no information.

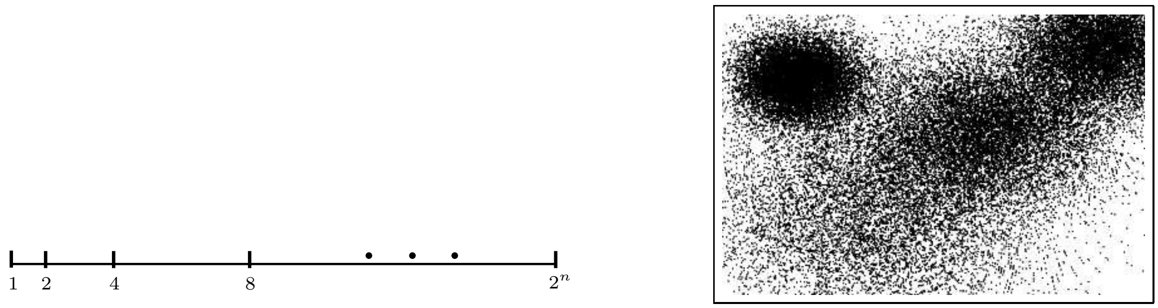
Metric dimension. Given this state of affairs, an increasing amount of recent attention has focused on understanding the complexity of NNS in terms of a metric's implicit structure. In Euclidean spaces, an obvious and common measure for a metric's complexity is the dimension of the Euclidean host space. Thus it is natural that to characterize the complexity of general metric spaces, one ought to define an analogous notion of *metric dimension,* and indeed this approach has been pursued to great success in recent papers [Cla99, KR02, KL04, HKMR04], where significant progress on solving exact and approximate versions of the NNS problem in general metrics has been made.

Unfortunately, each of these works falls short of offering the sort of generality that one should desire from such an approach. In [Cla99], to achieve efficient algorithms (for exact NNS), it is necessary to make some strong assumptions about the distribution of queries. In [KR02, HKMR04], the notion of dimension is too restrictive, eliminating large classes of metric spaces which should be considered low-dimensional, and for which efficient algorithms should exist (see [KL04] for a more detailed explanation).

Finally, in [KL04], a more satisfying notion of dimension (taken from [GKL03], and independently used in a different form by [Cla99]) is proposed, but the algorithms in both [KL04] and [Cla99] are efficient only under the additional assumption that the *aspect ratio* Φ (i.e. the ratio of the largest to smallest distance in X) is at most polynomial in $n = |X|$. In particular, the algorithm presented in [KL04] answers approximate nearest neighbor queries in time $2^{O(\dim(X))} \log \Phi$. Thus even when the set of points is $X = \{1, 2, 4, \ldots, 2^n\} \subseteq \mathbb{R}$ with the line metric $d(x, y) = |x - y|$, as in Figure 1(a), the algorithms of [KL04], as well as those of [Cla99, KR02, HKMR04], require $\Omega(n)$ time to answer some queries (i.e. they are no better than the trivial algorithm which tests every point). Despite the fact that (X, d) is clearly "low-dimensional" (being a subset of the real line), previous approaches perform dismally. Besides being theoretically disappointing, these algorithms are incapable of searching for (even approximate) nearest neighbors in highly clustered data (e.g. Figure 1(b)).

Efficient algorithms in the black-box model. In the present work, we are concerned with *approximate nearest neighbor search* (ANN). The $(1+\varepsilon)$-ANN problem is defined as follows: Given a query $q \in M$, we are required to return an element $a \in X$ for which $d(q, a) \leq (1+\varepsilon)\, d(q, X)$, where $d(q, X)$ is the distance from q to the closest point in X. (This is after an initial preprocessing of the point set X.) We resolve the aforementioned shortcomings by presenting an ANN data structure for general metric spaces which is efficient whenever $\dim(X)$ (defined formally in Section 1.2) is small, and under no additional assumptions. For this purpose we assume throughout that $\varepsilon > 0$ is fixed, i.e., independent of $|X|$.

We will discuss our definition of "efficient" momentarily. Beforehand, let us describe another question that arises in the above framework: Is "dimension" the right notion to characterize the complexity of ANN in general metric spaces? Certainly one may motivate the study of algorithms for "low-dimensional" metrics by their abundance in practical settings (see [KL04]), but one should also consider how tightly $\dim(X)$ captures the difficulty of nearest neighbor search in general metrics. To this end, we consider a *black-box model* of nearest neighbor search in arbitrary metric spaces, where the query is accessed as an oracle via the distance function. We say that a metric



(a) The doubling line

(b) Mixture of Gaussians in the plane

Figure 1: Clustered point sets

space X admits an *efficient* $(1+\varepsilon)$-ANN scheme if there exists a (possibly randomized) algorithm which answers any possible $(1+\varepsilon)$-approximate nearest neighbor query in polylogarithmic (in n) time using only polynomial (in n) space. Although quite a natural definition, we offer additional motivation in Section 1.2, where the model is specified more precisely. Under this complexity regime, we prove the following theorem.

Theorem 1.1. *Let $0 < \varepsilon < \frac{2}{5}$ be fixed. Then every metric space X admits an efficient $(1+\varepsilon)$-ANN scheme in the black-box model* if and only if $\dim(X) = O(\log\log n)$.

Thus, below a certain approximation threshold, $\dim(X)$ captures precisely the complexity of the problem. The behavior above the threshold is quite different, and we demonstrate in Section 2 that for sufficiently coarse approximations, the "ambient space" begins to play a crucial role, and even metrics of very high dimension may become tractable. We note that the constants of these constructions are not optimized; our primary goal is simply to show the existence of an approximation threshold, on either side of which drastically different behaviors are exhibited.

1.1 Techniques

The proof of Theorem 1.1 follows immediately from Theorem 2.1, which shows a lower bound on the black-box complexity of NNS in terms of $\dim(X)$, and from Theorem 3.1, which (together with Lemma 1.3) proves an upper bound on the NNS complexity.

Upper bounds. We give the first efficient (i.e. polylog(n) query time and poly(n) space) $(1+\varepsilon)$-ANN scheme for metrics whose doubling dimension, denoted by $\dim(X)$ (and defined in Section 1.2), is small. In particular, these bounds hold whenever $\dim(X) = O(\log\log n)$. For instance, in the case where $\dim(X) = O(1)$, our algorithm answers queries in $O(\log n)$ time and $O(n^3)$ space, thus making only $O(\log n)$ calls to the distance function. We note that the space bound we achieve for arbitrary metrics—obtained in Section 3.2—is a polynomial whose degree is *independent* of $\dim(X)$ and the desired approximation. Indeed, our data structure can be built without knowledge of ε (which can be passed as a parameter at query-time). When $\dim(X)$ is small, a general reduction from $O(1)$-ANN to $(1+\varepsilon)$-ANN follows easily from the techniques of [KL04], which we review in Section 1.2.

Our data structure is based on two new techniques. The first is a structural theorem about the existence of "dense," "well-separated" clusters of points in low-dimensional metrics. These sets manifest themselves in the form of *ring separators*— "thick" annuli whose inside and outside each contain a large fraction of the points. (A similar object is used in the construction of the ring-cover trees of [IM98] which are used to solve ANN in $\mathbb{R}^d$. Our setting is quite different, since we are not reducing to the problem of point location in equal balls. Hence we must show that for low-dimensional metrics, ring separators exist unconditionally.) Using these separators, we build a binary decision tree of height $2^{O(\dim(X))} \log n$ which can be used to answer $O(1)$-ANN queries in time $2^{O(\dim(X))} \log n$. Unfortunately, the natural implementation of this tree requires space $n^{2^{O(\dim(X))}}$, which is $n^{\Omega(\mathrm{polylog}(n))}$ even when $\dim(X) = \Theta(\log \log n)$.

This exponential blow-up in space is a typical problem encountered in NNS algorithms based on metric decompositions, and is the most difficult technical challenge faced by the present work. In Section 3.2, we overcome this problem for low-dimensional metrics, obtaining a data structure that uses $O(n^3)$ space whenever $\dim(X) = O(\log \log n)$. In addition, even for arbitrary n-point metrics (with no bound on the dimension), the space consumption is only polynomial in n. This improvement requires a second new technique which amounts to "folding" the decision tree back onto itself, often merging many apparently distinct branches into one. The difficulties and solutions are discussed more thoroughly in Section 3.2. This folding allows us to obtain a very compact "approximate" representation of the previously huge decision tree, while incurring only a small additional overhead at every node.

We note that since the doubling dimension was introduced in [GKL03], and the premise that "low-dimensional" general metrics should be algorithmically tractable was set forth, an increasing number of works have found applications of this idea to optimization problems; we mention, in particular, the predecessor to this work [KL04] and the new results of [Tal04] for approximating problems like TSP in low-dimensional metrics. In this context, we also mention the related work of [KKL03] in machine learning. We believe that the development and discovery of the rich properties of low-dimensional metrics continued herein will find additional application elsewhere.

Lower bounds. Our lower bounds are entirely information theoretic. Given a metric space X, there exists a "nearly-uniform" submetric $S \subseteq X$ whose size is roughly $k \geq 2^{\Omega(\dim(X))}$. Nearly uniform means that the aspect ratio (the ratio of the largest pairwise distance in S to the smallest) is bounded by a small constant. In Section 2, we then prove that, for every $\delta > 0$, this "large" subset S must contain within itself a subset $T \subseteq S$ with very small aspect ratio, i.e. $2 + \delta$, and yet which satisfies $|T| \geq k^{\delta'}$ (where δ' depends on δ, of course). This is a very simple (yet interesting) Ramsey-like property of metric spaces. Similar arguments have been made in independently in [BLMN03].

Now, if an algorithm were not allowed to compute distances from the query to $X \setminus T$ (i.e. the "ambient space"), then a lower bound of $\Omega(k^{\delta'})$ queries for $(1+\delta)$-ANN would follow fairly easily for T. And indeed, by a slightly technical extension argument, we can prove that any algorithm solving the $(1+\varepsilon)$-ANN problem must make at least $2^{\Omega(\dim(X))}$ queries to the distance oracle for $\varepsilon < \frac{2}{5}$. This shows that in the black-box model, querying against the ambient space cannot help too much when one requires a sufficiently fine approximation.

But our lower bound breaks down for coarser approximations, and we show that this is for good reason: When only a 3-approximation is desired, there are n-point metrics X with $\dim(X) = \Omega(\log n)$ for which every query against X can be decided in $O(\log n)$ time in the black-box model. Thus above a certain approximation threshold, $\dim(X)$ no longer characterizes the complexity of ANN.

1.2 Preliminaries

Metric spaces. Let (X, d) be an n-point metric space, and let $S \subseteq X$ be a subset. We denote by

$$B_S(x, r) = \{y \in S : d(x, y) < r\}$$

the open ball of radius r about x in S. When $S = X$, we omit the subscript S. We write $d(x, S) = \inf_{y \in S} d(x, y)$. Define $\mathrm{diam}(S) = \sup_{x,y \in S} d(x, y)$, and let the *aspect ratio* of S be the quantity

$$\Phi(S) = \frac{\mathrm{diam}(S)}{\inf_{x,y \in S} d(x, y)}.$$

Finally, we say that a subset Y of X is a *β-net* if it satisfies (1) For every $x, y \in Y$, $d(x, y) \geq \beta$ and (2) $X \subseteq \bigcup_{y \in Y} B(y, \beta)$. Such nets always exist for any $\beta > 0$. For finite metrics, they can be constructed greedily. For arbitrary metrics, proof of their existence is an easy application of Zorn's lemma.

The doubling dimension. We recall that the *doubling constant* $\lambda(X)$ is the least value λ such that every ball in X can be covered by λ balls of half the radius (see, e.g., [Hei01]). Though we will work with this quantity, it is technically more accurate to define a slightly different quantity $\lambda'(X)$ to be the least value λ' such that every *set* in X can be covered by λ' *sets* of half the *diameter*.

The *doubling dimension* [GKL03] is then defined by $\dim(X) = \log_2 \lambda'(X)$. Here are some simple properties which demonstrate that $\dim(X)$ is a robust and meaningful notion.

1. For $X = \mathbb{R}^k$ equipped with any norm, $\dim(X) = \Theta(k)$.
2. If $S \subseteq X$, then $\dim(S) \leq \dim(X)$.
3. $\dim(X_1 \cup \cdots \cup X_m) \leq \max_i \dim(X_i) + \log m$.
   (In particular, $\dim(X) \leq \log |X|$.)

It is not difficult to see that $\log_2 \lambda(X)$ and $\log_2 \lambda'(X)$ differ by at most a factor of two, and thus in everything that follows, we assume that $\dim(X) = \log_2 \lambda(X)$, as it makes the exposition simpler.

The following simple lemma expresses the standard packing/covering duality.

Lemma 1.2 (Nearly-uniform metrics). *Let (X, d) be a metric space, and let $S \subseteq X$. If the aspect ratio of the metric induced on S is at most $\Phi \geq 2$, then $|S| \leq \Phi^{O(\dim(X))}$.*

Proof. Let $d_{\min} = \inf\{d(x, y) : x, y \in S\}$ and $d_{\max} = \sup\{d(x, y) : x, y \in S\}$ be the minimum and maximum interpoint distance in S, respectively, and assume that $\Phi = \frac{d_{\max}}{d_{\min}} < \infty$. Notice that S is contained in a ball of radius $2\, d_{\max} \leq 2\Phi\, d_{\min}$ in X (centered at any point of S). Applying the definition of doubling dimension iteratively several times we get that this ball, and in particular S, can be covered by $2^{\dim(X) \cdot O(\log \Phi)}$ balls of radius $d_{\min}/3$. Each of these balls can cover at most one point of S (by definition of $d_{\min}$) and thus $|S| \leq 2^{\dim(X) \cdot O(\log \Phi)} \leq \Phi^{O(\dim(X))}$. □

In particular, the above lemma provides a bound on the cardinality of a δR-net intersected with a ball of radius R. Namely, such an intersection contains at most $(\frac{1}{\delta})^{O(\dim(X))}$ points.

The black-box model and efficiency. Our model is quite simple. Suppose that (X, d) is a metric space. We assume that the only thing known about the query (and thus the only constraint on the query) is that the space $(X \cup \{q\}, d)$ is again a metric space, i.e. that the query does not violate the triangle inequality. The only access that an algorithm has to the query is through oracle calls to the distance function, i.e. the values $d(q, x)$ for $x \in X$. We assume that $d(q, \cdot)$ can be evaluated in unit time (although this is without loss of generality, since our upper bounds scale linearly with the time needed to evaluate the distance function, and our lower bounds are in terms of the number of calls to $d(q, \cdot)$).

We define an algorithm as "efficient" if, after the preprocessing phase, it can answer any query in polylog(n) time using only poly(n) space for any fixed $\varepsilon > 0$. We make no restriction on preprocessing time or space, but we note that in all of our upper bounds, both are linear in the space used by the algorithm for answering a query.

As for the running time, we note that all of the algorithms in [Cla99, KR02, KL04] strive for polylog(n) query times, thus it is *the* natural candidate for "efficiency." We also note that the best algorithms for ANN in high-dimensional Euclidean spaces answer queries in polylog(n) time [IM98, KOR98, HP01]. As for space, poly(n) is again the natural choice, but this assumption should not be abused. Straightforward implementations of the algorithms of [IM98] and [KOR98], although outstanding theoretical achievements, are hampered due to their extremely high space complexity (the degree of the polynomial grows with $\frac{1}{\varepsilon}$ for $(1+\varepsilon)$-ANN).

Even in the worst case (i.e. when $\dim(X) = \Omega(\log n)$), the algorithms of Section 3.2 use only poly(n) space (independent of the approximation factor desired). When $\dim(X) = O(\log\log n)$, the space consumption is $O(n^3)$. This factor has not been optimized, and we hope that eventually a near-linear space algorithm can be obtained, at least for the case when $\dim(X) = O(1)$.

The [KL04] reduction to $O(1)$-ANN. The next lemma follows immediately from the ANN algorithm of [KL04].

Lemma 1.3 (The [KL04] reduction). *For every n-point metric space (X,d) there is an algorithm consuming $O(n^2)$ space, such that given a query q, two parameters $\alpha > 1$ and $0 < \varepsilon < 1/2$, and a point in X which is an α-ANN to q, the algorithm computes a $(1+\varepsilon)$-ANN in time $(\alpha/\varepsilon)^{O(\dim(X))} + O(\log n)$.*

The idea is that given $x \in X$ which is an α-ANN to q, it suffices to enumerate, for $r = \Theta(\varepsilon/\alpha) \cdot d(q,x)$, over all the r-net points in $B(x, 3d(x,q))$, and output the one which is closest to q. In essence, the data structure of [KL04] maintains a 2^i-net for every integer i, together with pointers from every 2^i-net point to the nearby 2^{i-1}-net points; this consumes only $2^{O(\dim(X))} \cdot n$ space. A simple iterative procedure then yields the desired enumeration. The main difference from [KL04] is that here we wish to use the α-approximate nearest neighbor point given to us as a "warm start". And indeed, we can maintain for every point $x \in X$ and every integer 2^j the closest point to x in the 2^j-net, so that the total space is only $O(n^2)$ and the access time is $O(\log n)$.

The running time of this reduction is only polylog(n) whenever $\dim(X) = O(\log\log n)$ and $\alpha/\varepsilon = O(1)$, thus we content ourselves with finding $O(1)$-ANNs in everything that follows.

2 Lower bounds

In this section, we show that for any metric space X and any fixed $\varepsilon < \frac{2}{5}$, solving the $(1+\varepsilon)$-ANN problem on X is as hard as unordered search in a k-element database with $k = 2^{\Omega(\dim(X))}$. It will follow that any algorithm (deterministic or randomized) which solves the $(1+\varepsilon)$-ANN problem on X must make at least $2^{\Omega(\dim(X))}$ calls to the distance oracle for some query q. We note that the constructions of this section are not optimized; our goal is simply to show the existence of an approximation threshold, on either side of which drastically different behaviors are exhibited.

Theorem 2.1. *For every metric space X and every fixed $\varepsilon < \frac{2}{5}$, every algorithm solving the $(1+\varepsilon)$-ANN problem on X must make at least $2^{\Omega(\dim(X))}$ calls to the distance oracle for some query q. For randomized algorithms, this bound holds in expectation.*

First, we require a partial converse to Lemma 1.2.

Lemma 2.2. *For any n-point metric space X and any $2 < \Phi_0 \leq 4$, there exists a subset $S \subseteq X$ with $\Phi(S) \leq \Phi_0$ and $|S| \geq 2^{\dim(X)\lceil \log \Phi_0 - 1\rceil}$.*

Proof. Let k be the least number for which every subset $S \subseteq X$ with $\Phi(S) \leq \Phi_0$ satisfies the bound $|S| \leq k$. Then every ball $B(x, \frac{1}{2}\Phi_0 r)$ can be covered by k balls of radius r. To see this, let N be an r-net in $B(x, \Phi_0 r)$. Then it is clear that $\Phi(N) \leq \Phi_0$, hence $|N| \leq k$. But from the definition of a net, the r-balls around points in N cover all of $B(x, \frac{1}{2}\Phi_0 r)$.

Now consider $B(x, R)$ for some $x \in X$. From the above discussion, we see that $B(x, R)$ can be covered by k balls of radius $2R/\Phi_0$. Since each of these smaller balls can be covered by k balls of radius $R(2/\Phi_0)^2$, we see that $B(x, 2R)$ can be covered by k^2 balls of radius $R(2/\Phi_0)^2$.

Continuing in this manner, we see that every ball of radius R can be covered by $k^{\lceil 1/(\log \Phi_0 - 1)\rceil}$ balls of radius $R/2$. It follows that $k^{\lceil 1/(\log \Phi_0 - 1)\rceil} \geq 2^{\dim(X)}$ so that $k \geq 2^{\dim(X)(\log \Phi_0 - 1)}$. In other words, there exists some subset $S \subseteq X$ with $\Phi(S) \leq \Phi_0$ and $|S| \geq 2^{\dim(X)(\log \Phi_0 - 1)}$. □

Theorem 2.3. *Let (X, d) be any metric space which contains a submetric $S \subseteq X$ with $\Phi(S) = \Phi_0$ and $|S| \geq k$. Then there exists some $\varepsilon = \varepsilon(\Phi_0) > 0$ such that any algorithm for $(1+\varepsilon)$-ANN on X must make at least $\Omega(k)$ calls to the distance oracle for some query q. For randomized algorithms, this holds in expectation.*

Proof. Let $S = \{x_1, x_2, \ldots, x_k\}$. Let $d_{\max} = \max_{x,y \in S} d(x, y)$ and $d_{\min} = \min_{x,y \in S} d(x, y)$. To each index $i \in \{1, \ldots, k\}$, we associate a query q_i. The idea is that q_i is close to x_i and far from $S \setminus \{x_i\}$, and hence a $(1+\varepsilon)$-ANN scheme essentially discovers x_i. Note however that $d(q_i, x_i)$ should not be too small (say 0), as otherwise the distance between q_i and some fixed point, say x_1, might reveal the value of i (e.g. if the distances $d(x_1, x_j)$ are distinct). Another complication is that a $(1+\varepsilon)$-ANN algorithm might make use of (or return as an answer) the points in $X \setminus S$, i.e. the "ambient space." Hence, when defining $d(q_i, y)$ for $y \in X$ we must make a smooth transition between the cases where y is (i) close to x_i, (ii) close to $S \setminus \{x_i\}$, or (iii) far from both.

Formally, the query q_i is defined by:

$$d(q_i, y) = \begin{cases} \frac{1}{2}d_{\max} + \frac{1}{4}d_{\min} + d(y, x_i) & \text{if } d(y, x_i) < \frac{1}{2}d_{\min} \\ \frac{1}{2}d_{\max} + \frac{3}{4}d_{\min} & \text{if } d(y, x_i) \geq \frac{1}{2}d_{\min} \text{ and } d(y, S) \leq \frac{1}{2}d_{\min} \\ \frac{1}{2}d_{\max} + \frac{1}{4}d_{\min} + d(y, S) & \text{otherwise} \end{cases}$$

First, we must assure that the space $(X \cup \{q_i\}, d)$ satisfies the triangle inequality for every $1 \leq i \leq k$. The proof (which is slightly technical) appears in Lemma A.1 of the appendix.

Now, let us continue in proving our lower bound. First, note that for query q_i, the unique closest point is x_i and $d(q_i, x_i) = \frac{1}{2}d_{\max} + \frac{1}{4}d_{\min}$. Also, note that for $y \notin B(x_i, \frac{1}{2}d_{\min})$, we have $d(q_i, y) \geq \frac{1}{2}d_{\max} + \frac{3}{4}d_{\min}$. Thus any $(1+\varepsilon)$-ANN algorithm, for $\varepsilon < 1/(\Phi_0 + \frac{1}{2})$, must find some point $y \in B(x_i, \frac{1}{2}d_{\min})$ on query q_i. This is because

$$\frac{\frac{1}{2}d_{\max} + \frac{3}{4}d_{\min}}{\frac{1}{2}d_{\max} + \frac{1}{4}d_{\min}} = 1 + \frac{1}{\Phi_0 + \frac{1}{2}}.$$

In other words, given query q_i, where $i \in \{1, \ldots, k\}$, the algorithm must be able to determine which ball $B(x_i, \frac{1}{2}d_{\min})$ contains the "close" points. And since for distinct x_i, these balls are disjoint, the algorithm must be able to figure out the index i.

We claim that this is as hard as searching for the index in an unordered list. In other words, the best the algorithm can do given a query q is ask the question "Is $q = q_i$?" To see this, and note first that for y not in $S' = \{x \in X : d(x, S) \leq \frac{1}{2}d_{\min}\}$, the value of $d(q_i, y)$ is independent of the value $i \in \{1, \ldots, k\}$, thus these queries don't help at all. Furthermore, the value of $d(q_i, y)$ for $y \in S' \setminus B(x_i, \frac{1}{2}d_{\min})$ is independent of the index $i \in \{1, \ldots, k\}$. Thus when asking the "important" questions $d(q_i, y)$ for $y \in S'$, the algorithm is simply told YES if $y \in B(x_i, \frac{1}{2}d_{\min})$ and NO otherwise. This completes the proof. □

Now we prove the main theorem of this section.

Proof of Theorem 2.1. Let (X, d) be any metric space. Note that as $\Phi_0 \to 2$ in Lemma 2.2, the lower bound value of ε to which the preceding theorem applies behaves like $\varepsilon < (\Phi_0 + \frac{1}{2})^{-1} \to \frac{2}{5}$. Thus for any fixed $\varepsilon < \frac{2}{5}$, there is a lower bound of $2^{\Omega(\dim(X))}$ on the number of calls to the distance function which are needed to answer some $(1+\varepsilon)$-ANN query. $\square$

We obtain the following corollary.

Corollary 2.4. *If* $\dim(X) = \omega(\log \log n)$ *and* $\varepsilon < \frac{2}{5}$*, then there is no efficient* $(1+\varepsilon)$*-ANN scheme for* X *in the black-box model, since* $2^{\Omega(\dim(X))}$ *is bigger than any* $\operatorname{polylog}(n)$*.*

2.1 Above the threshold

In this section, we show that when coarser approximations are desired, there are metrics of high dimension which nevertheless admit very efficient ANN algorithms, and thus the lower bounds of the previous section cannot be pushed too much further. Again, we do not seek to optimize constants.

Lemma 2.5. *There exists an* n*-point metric space* (X, d) *with* $\dim(X) = \Theta(\log n)$*, which admits a* 3*-ANN algorithm with query time* $O(\log n)$ *and space* $O(n)$*.*

Proof. Assuming that n is a power of 2, let $A = \{e_1, \ldots, e_n\}$ where $e_i \in \mathbb{R}^n$ is an n-dimensional vector with a 1 in the ith coordinate and zeros elsewhere. Additionally, let B consists of $\log n$ vectors; to construct the jth vector, partition the n coordinates into (consecutive) blocks of size 2^{j-1}, and let coordinates in odd blocks be -1 and those in even blocks be 0. We endow these points with the ℓ_∞ metric, i.e., for any two vectors $u, v \in \mathbb{R}^n$, let $d(u, v) = \|u - v\|_\infty = \max_{1 \le i \le n} |u_i - v_i|$ (where v_i is the ith coordinate of v).

Let $X = A \cup B$ be the set of points to be preprocessed. Clearly, X contains the n-point uniform metric A, and thus $\dim(X) \ge \dim(A) = \log n$. On the other hand, $|X| \le 2n$ and thus $\dim(X) \le 1 + \log n$.

However, it is not difficult to exhibit a 3-ANN algorithm for X. Let q be the query. First, note that since $|B| = O(\log n)$, we may efficiently compute $d(q, b)$ for every $b \in B$. Our algorithm will clearly report the closest point to q among the points of $A \cup B$ that were tested, and hence we may assume that $d(q, A) = d(q, X)$, i.e. the closest point to q is in A. Call this point e_k. Now, if $d(q, e_k) \ge \frac{1}{2}$, then for any $k' \ne k$, $d(q, e_{k'}) \le 1 + d(q, e_k) \le 3\, d(q, e_k)$, thus we may return $e_{k'}$ as a 3-approximate nearest neighbor. Hence we may assume that $d(q, e_k) < \frac{1}{2}$.

In what follows, we write b_i for the ith coordinate of b. Observe that for each vector $b \in B$,

$$d(b, e_i) = \begin{cases} 1 & \text{if } b_i = 0; \\ 2 & \text{if } b_i = -1. \end{cases}$$

By the triangle inequality, if $b_k = 0$ then $d(b, q) \le d(b, e_k) + d(e_k, q) < 3/2$, and if $b_k = -1$ the $d(b, q) \ge d(b, e_k) - d(e_k, q) > 3/2$. Hence, for each $b \in B$, when we check whether $d(b, q) < 3/2$ or not, we effectively find out whether $k \in \{i : b_i = 0\}$ or whether $k \in \{i : b_i = -1\}$. Now if b is the jth vector in B, then this simply determines the jth (least significant) bit in the binary representation of k. Repeating this for all $b \in B$ uniquely identifies the index k.

Let us reiterate the algorithm: We compute $d(q, b)$ for every $b \in B$. Let $b^* \in B$ be the closest point to q amongst those of B. The sequence of distance queries determines an index k' (as above); let $e_{k'}$ be the corresponding point. We then return the closest point among $\{b^*, e_{k'}\}$. $\square$

3 Efficient algorithms

We provide two algorithms for $(1+\varepsilon)$-approximate nearest neighbor search in a general metric space; the two have similar query time, but they differ in their space requirement. By the general reduction discussed in Section 1.2, it suffices to exhibit an $O(1)$-ANN algorithm. Our first algorithm (Section 3.1) is based on the existence of a certain ring-separator, which naturally yields a binary decision tree that can be used to solve 3-ANN. The decision tree's depth is $2^{O(\dim(X))}\log n$, so this algorithm has an optimal query time. However, its space requirement grows rapidly with $\dim(X)$. The second algorithm, which achieves space that is polynomial in n (independently of $\dim(X)$) is significantly more complex, and we refer the reader to Section 3.2 for a discussion of the subtle issues which arise. This algorithm proves the main result of this section, as follows.

Theorem 3.1. *For every n-point metric space (X,d) there exists an $O(1)$-ANN algorithm that consumes n^3 space and answers every query q in time $2^{O(\dim(X))}\log n$.*

3.1 The ring-separator tree

The basic notion introduced in this subsection is that of a ring-separator; this naturally yields a ring-separator tree, which can be used as a binary decision tree for 3-ANN. Throughout this section, we shall use the following definition. For $x \in S \subseteq X$ and $R_1, R_2 \geq 0$, define the *annulus about* x as

$$A_S(x, R_1, R_2) \stackrel{\text{def}}{=} B_S(x, R_2) \setminus B_S(x, R_1).$$

The ring-separator. Let (X,d) be an n-point metric space. A *δ-ring-separator* of a subset $S \subseteq X$ is a pair (x,R) consisting of a point $x \in S$ and a real $R > 0$, that satisfies the following condition: $|B_S(x,R)| \geq \delta|S|$ and yet $|B_S(x,2R)| \leq (1-\delta)|S|$. We now prove the main lemma of this subsection.

Lemma 3.2 (Ring separators). *For any metric space (X,d) and any subset $S \subseteq X$ with $|S| \geq 2$, there exists a δ-ring-separator of S with $\delta \geq (\frac{1}{2})^{O(\dim(X))}$.*

Proof. We proceed by contradiction. Fix some $0 < \delta < 1$ and assume that S does not have a δ-ring-separator; we will show that for a sufficiently large constant $c > 0$, $\delta > (\frac{1}{2})^{c\dim(X)}$, yielding the desired result.

Let $\bar{B}_S(x,r) = \{y \in S : d(x,y) \leq r\}$ be the *closed ball* of radius r around x (in S). For every point $x \in S$, let $R(x) \stackrel{\text{def}}{=} \min\{R \geq 0 : |\bar{B}_S(x,R)| \geq \delta|S|\}$. Since $|S| \geq 2$ is finite, $R(x)$ is defined and furthermore $|B_S(x,R(x))| < \delta|S|$. By our assumption, for all $x \in X$, $|B_S(x,2R(x))| > (1-\delta)|S|$, and hence each annulus $A_S(x_i, R(x_i), 2R(x_i))$ contains at least $(1-2\delta)|S|$ points.

Let $x_0 \in S$ be the point for which $R(x_0)$ is minimal, and iteratively for $t = 1, 2, \ldots$ choose $x_t \in S$ to be an arbitrary point of

$$\bigcap_{i=0}^{t-1} A_S(x_i, R(x_i), 2R(x_i)).$$

Clearly we can continue this process as long as the above intersection remains non-empty. Suppose we are forced to stop after selecting k points $x_0, x_1, \ldots, x_{k-1}$. On the one hand, we threw away at most $2\delta|S|$ points at every step, and thus $k \geq 1/2\delta$. On the other hand, the set $U = \{x_0, x_1, \ldots, x_{k-1}\}$ is contained in $B(x_0, 2R(x_0))$. Furthermore, for any pair x_i, x_j with $i < j$, we see that $d(x_i, x_j) \geq R(x_i)$ since $x_j \notin B_S(x_i, R(x_i))$. But by construction, $R(x_i) \geq R(x_0)$ for all $i \geq 0$. It follows that the set U has $\Phi(U) \leq 4$, and thus by Lemma 1.2, $k \leq 2^{O(\dim(X))}$. We conclude that $\delta \geq 1/2k \geq (\frac{1}{2})^{O(\dim(X))}$. □

The ring-separator tree. Given the above lemma, it is natural to define a δ-*ring-separator tree* for a metric space (X, d). This is a binary tree where each node has a label $S \subseteq X$, constructed recursively as follows. The root of the tree has the label $S = X$. A node labeled by S is a leaf if $|S| = 1$, and has two children if $|S| \geq 2$. In the latter case, we take (x, R) to be a δ-ring-separator of S and add it to the node's label, i.e., the label becomes $\langle S, (x, R)\rangle$ (where $S \subseteq X$, $x \in S$ and $R > 0$). The two children of the node are an *inside child*, whose label is $S_I = B_S(x, 2R)$, and an *outside child*, whose label is $S_O = S \setminus B_S(x, R)$. Note that S_I and S_O are not a partition of S, as their intersection is generally non-empty. Lemma 3.2 shows that if $|S| \geq 2$ then S admits a δ-ring-separator with $\delta \geq (\frac{1}{2})^{O(\dim(X))}$. Since every step (away from the root) decreases the size of S by a factor of at least $1 - \delta$, the height of the tree is at most $2^{O(\dim(X))} \log n$.

The 3-ANN algorithm. We now show how to use ring-separator trees to solve the 3-ANN problem on X in time $2^{O(\dim(X))} \log n$. Unfortunately, a bound of $2^{O(\dim(X))} \log n$ on the height of the ring-separator tree implies a possibly huge space requirement of $n^{2^{O(\dim(X))}}$. This problem will be remedied in Section 3.2.

Let q be the query against X. The algorithm proceeds along a root to leaf path, i.e., starts at the root and iteratively goes down the tree until a leaf node is met. Suppose that we are at a node $N = \langle S, (x, R)\rangle$. If $d(q, x) \leq 3R/2$, the algorithm proceeds to the inside child of N; otherwise, it proceeds to the outside child. The iterative process ends when a leaf node $N = \langle\{x\}\rangle$ is met. Let x_i be the point x seen in the ith node along this root to leaf path (either the point from the ring-separator or the only point in S at a leaf node). The algorithm outputs the point which is closest to q among the encountered points $\{x_i\}$.

This algorithm clearly runs in time linear in the height of the tree, i.e. $2^{O(\dim(X))} \log n$. We now proceed to show that the point x_i which is output is indeed a 3-approximate nearest neighbor to q.

Proposition 3.3. *The above algorithm outputs a* 3*-approximate nearest neighbor to* q*.*

Proof. Let $a^* \in X$ be a real nearest neighbor to q, i.e. $d(q, X) = d(q, a^*)$. Let $N_1, N_2, \ldots, N_k$ be the sequence of tree nodes seen by the algorithm on input q. For $i < k$ let $N_i = \langle S_i, (x_i, R_i)\rangle$, and let $N_k = \langle\{x_k\}\rangle$. Clearly $a^* \in S_1$ since $S_1 = X$. If $a^* \in S_k$, then $x_k = a^*$, and in this case the algorithm returns the exact nearest neighbor. Otherwise, there exists some j for which $a^* \in S_j$ but $a^* \notin S_{j+1}$. We claim that in this case, x_j is a 3-approximate nearest neighbor to q.

If N_{j+1} is the inside child of N_j, then $d(q, x_j) \leq 3R_j/2$, yet $d(a^*, x_j) \geq 2R_j$, so by the triangle inequality,

$$d(q, a^*) \geq d(a^*, x_j) - d(q, x_j) \geq 2R_j - 3R_j/2 = R_j/2 \geq d(q, x_j)/3.$$

If N_{j+1} is the outside child of N_j, then $d(q, x_j) > 3R_j/2$, yet $d(a^*, x_j) \leq R_j$. Again by the triangle inequality $d(q, a^*) \geq R_j/2$, and we conclude that

$$d(x_j, q) \leq d(x_j, a^*) + d(a^*, q) \leq R_j + d(a^*, q) \leq 3\, d(a^*, q).$$

The proof follows by recalling that the algorithm outputs the closest point to q among $x_1, \ldots, x_k$. $\square$

3.2 Polynomial storage

We now achieve a space requirement that is polynomial in n, regardless of $\dim(X)$, by modifying the ring-separator tree algorithm of Section 3.1. In a nutshell, we employ three techniques that, when applied together, "massage" the decision tree into a polynomial size directed acyclic graph (DAG) that can be used for $O(1)$-ANN. First, we obtain a more "canonical" form of the decision tree by snapping every δ-ring-separator to a suitable net of the metric. This step limits the number distinct radii that are used by the ring-separators in the data structure. Second, we eliminate altogether the outside children, essentially replacing each one by sufficiently many inside children (i.e., balls). The advantage in having only inside children is that now if a path in the tree (sequence of inside children) corresponds to properly nested balls, then the information in the entire sequence may be represented by a single ball (namely, the last one, considered as a ball in X). And indeed, the key idea in the third step is to maintain the invariant that whenever we go to an inside child corresponding to a ball $B_X(y, 2R)$, it holds that $d(y, q) \leq \beta R$ for a suitable constant $1 < \beta < 2$. This invariant is used to ensure that the next inside child in the sequence is properly nested, and hence we can merge nodes that have the same role (i.e., correspond to the same ball). As a consequence, the decision tree "folds" onto itself, creating a DAG of polynomial size.

We start by describing the first two techniques mentioned above, and then provide the actual scheme, which combines all three techniques. For clarity of exposition, we make no attempt to optimize the constants.

For the rest of this subsection, fix an r-net Y_r of X for every r in the set Γ of all powers of 2 in the range $[\frac{1}{128} d_{\min}, 4d_{\max}]$; here $d_{\min} = \inf\{d(x, y) : x, y \in X\}$ is the minimum interpoint distance in X and $d_{\max} = \sup\{d(x, y) : x, y \in X\}$ is the diameter of X. We further assume that $Y_r \subseteq Y_{r/2}$ for all $\min \Gamma < r \leq \max \Gamma$. Notice that $Y_{\max \Gamma}$ contains only one point and that $Y_{\min \Gamma} = X$.

Enhanced ring-separator. An *enhanced δ-ring-separator* of $S \subseteq X$ is a pair (x, t) consisting of $t/2 \in \Gamma$ and $x \in Y_{t/2}$, such that $|B_S(x, t)| \geq \delta|S|$ and yet $|B_S(x, 2t)| \leq (1-\delta)|S|$. Notice that it is a δ-ring-separator of S, except that x is allowed to be in $X \setminus S$. We first enhance the ring-separator lemma.

Lemma 3.4 (Enhanced ring separators). *For any metric space (X, d) and any subset $S \subseteq X$, there exists an enhanced δ-ring-separator (x, t) of S with $\delta \geq (\frac{1}{2})^{O(\dim(X))}$.*

Proof. By suitably modifying various constants in the proof of Lemma 3.2, we obtain that for $\delta \geq (\frac{1}{2})^{O(\dim(X))}$ there exist $x^* \in S$ and $R^* > 0$ such that $B_S(x^*, R^*) \geq \delta|S|$ and $|B_S(x^*, 10R^*)| \leq (1-\delta)|S|$. Now let t be the unique power of 2 in the range $[2R^*, 4R^*)$. It is clear that $d_{\min} \leq R^* \leq d_{\max}$, and thus $t/2 \in \Gamma$. By the definition of a net, there exists $x \in Y_{t/2}$ such that $d(x^*, x) \leq t/2$. The triangle inequality implies that $|B_S(x, t)| \geq |B_S(x^*, t/2)| \geq \delta|S|$ and $|B_S(x, 2t)| \leq |B_S(x^*, 5t/2)| \leq (1-\delta)|S|$, which proves the lemma. $\square$

How to avoid outside children. Let $S \subseteq X$ and let (x, t) be an enhanced δ-ring-separator for S, i.e., with $t/2 \in \Gamma$ and $x \in Y_{t/2}$. Instead of constructing an inside child (which corresponds to a ball) and an outside child (which corresponds to the complement of ball) as we did in Section 3.1, we shall have two families of only inside children, as follows. The first family (see Figure 2) will handle queries q for which $d(x, q) \leq 2t$. For each point $y \in Y_{t/4}$ which is within distance $3t$ from x we create an inside child by taking $S_y = B_S(y, t/2)$. Since S_y has diameter at most t, it must be disjoint from either $B_S(x, t)$ or $S \setminus B_S(x, 2t)$, and thus $|S_y| \leq (1-\delta)|S|$. Using Lemma 1.2, this creates at most $2^{O(\dim(X))}$ inside children.

The second family (see Figure 3) will handle queries q such that $d(x, q) > 2t$. For every $R \in \Gamma$ and $R \geq 2t$ we do the following. For each point $y \in Y_{R/8}$ which is in the $(\frac{3}{4}R, \frac{9}{4}R)$-annulus around x, we create an inside child by taking $S_y = B_S(y, R/4)$. Since $d(x, y) \geq \frac{3}{4}R \geq \frac{1}{4}R + t$, this child S_y must be disjoint from $B_S(x, t)$, and thus contains at most $(1-\delta)|S|$ points. By Lemma 1.2, this creates at most $2^{O(\dim(X))}$ inside children. It is not difficult to see that the number of inside children is $O(n)$, because each point $y' \in X$ appears in at most $\lceil \log_2(\frac{9}{4}/\frac{3}{4}) \rceil = 2$ of these sets.

This modification yields a decision tree similar to the ring-separator tree, except that each node may have $O(n)$ inside children instead of two. The algorithm changes accordingly–at each node,

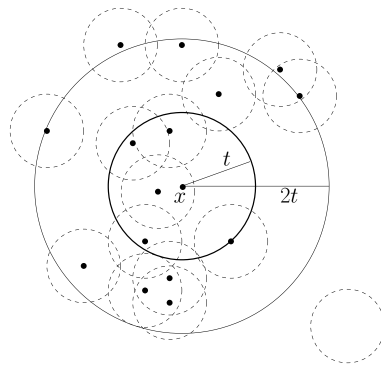


Figure 2: Balls $B(y, t/2)$ forming the first family of inside children.

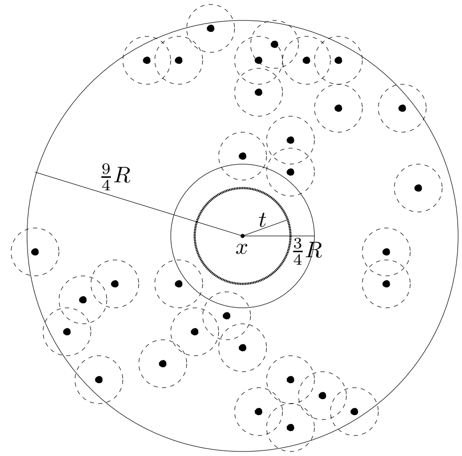


Figure 3: Balls $B(y, R/4)$ forming the subcase $R = 2t$ of the second family of inside children

we compute $d(x, q)$, recover the list of corresponding inside children (given by $Z_{t/4}$ or $Z_{R/4}$), and proceed to the ball whose center is closest to q. An $O(1)$-approximation guarantee follows quite easily, since the balls are defined so that they have a certain overlap. However, the depth of the tree is, as before, at most $2^{O(\dim(X))} \log n$, which provides a good (actually optimal) bound on the number of distance computations by the query procedure, but does not lead to a polynomial bound on the space requirement. So our next goal is to "compress" the decision tree into a DAG that has a small size. This requires a subtle modification, which will guarantee that along any path in the tree it actually suffices to "remember" only the last ball; namely, if our last child was $B_S(y, R)$ for S defined by the previous children along the path, then we can actually consider instead $B_X(y, R)$ (with no dependence on S). Since this introduces additional technical complications, we shall have to modify the constants in the above construction of inside children.

The ring-separator DAG. We start by defining the vertex set of the DAG. For every $R \in \Gamma$ and every point $y \in Y_R$ the DAG contains a vertex $\langle y, R\rangle$ that is associated with the set $B_X(y, 2R)$. Clearly, this vertex set size is bounded by $O(n \log \frac{d_{\max}}{d_{\min}})$. To achieve a bound that is polynomial in n, independently of $\frac{d_{\max}}{d_{\min}}$, we do not explicitly store vertices $\langle y, R\rangle$ for which $R > \min \Gamma$ and $B_X(y, 2R) = B_X(y, R/2)$. We call these *implicit vertices*. It is easy to see that for every point $y \in X$ there are only $O(n)$ values R for which the vertex $\langle y, R\rangle$ is stored explicitly. Therefore, the overall number of vertices is $O(n^2)$. To make it easier to track certain constants, we shall identify them as the following parameters $\alpha = \frac{3}{2}$, $\beta = \frac{9}{8}$, and $\gamma = \frac{5}{4}$.

The directed edges leaving a vertex $\langle y, R\rangle$ are as follows. If the set $S = B_X(y, 2R)$ associated with the vertex contains only one point, then this vertex $\langle y, R\rangle$ has no outgoing edges. Otherwise, let (x, t) be an enhanced δ-ring-separator for S, as guaranteed by Lemma 3.4. Notice that the ring-separator implies that S contains two points at least t apart, and thus $t \le \mathrm{diam}(S) \le 4R$. We have two families of edges outgoing from $\langle y, R\rangle$:

(a). For every $z \in Y_{t/32}$, we add an edge to the node $\langle z, t/32\rangle$ if

$$d(x, z) \le 3t \quad \text{and} \quad d(y, z) \le \alpha R. \tag{1}$$

(b). For every $R' \in \Gamma$, $R' \ge 2t$ and every $y' \in Y_{R'/32}$, we add an edge to the node $\langle y', R'/32\rangle$ if

$$\frac{3}{4}R' \le d(x, y') < \frac{9}{4}R' \quad \text{and} \quad d(y, y') \le \alpha R. \tag{2}$$

Note: $R' \le 8R$, because $\frac{3}{4}R' \le d(x, y') \le d(x, S) + 2R + d(y, y') \le t + 2R + \alpha R \le 7.5R$ (by the triangle inequality and since $|B_S(x, t)| \ge \delta|S| > 0$), which is just $R' \le 10R$, but both sides are integral powers of 2.

If an edge is supposed to go to an implicit vertex $\langle y, R\rangle$, we "redirect" it so that it leads instead to $\langle y, r\rangle$, where $r < R$ is the maximal possible value such that $\langle y, r\rangle$ is explicitly stored. Such a vertex always exists, because for every $y \in X$, the vertex $\langle y, \min \Gamma\rangle$ has an explicit representation. Note that this redirection is done at the preprocessing stage.

Lemma 3.5. *If an edge of the DAG goes from a vertex with point set S to a vertex with point set S', then $S' \subseteq S$, and furthermore $|S'| \le (1-\delta)|S|$ for $\delta \ge (\frac{1}{2})^{O(\dim(X))}$.*

Proof. Fix a DAG vertex $\langle y, R\rangle$, and let $S = B_X(y, 2R)$. Consider a directed edge of the family (a), going to $\langle z, t/32\rangle$. Notice that $S' = B_X(z, t/16) \subseteq B_X(y, 2R) = S$ because (1) implies that $d(y, z) + t/16 \le \alpha R + t/16 \le 2R$. Furthermore, S' has diameter at most $t/8$, must be disjoint from either $B_S(x, t)$ or $S \setminus B_S(x, 2t)$. Since (x, t) is an enhanced δ-ring-separator, we get that $|S'| \le (1-\delta)|S|$.

Consider a directed edge of the second family (b), going to $\langle y', R'/32\rangle$. Notice that $S' = B_X(y', R'/16) \subseteq B_X(y, 2R) = S$ because (2) implies that $d(y, y') + R'/16 \le \alpha R + R'/16 \le 2R$. Furthermore, S' is disjoint from $B_S(x, t)$ since the distance between their centers is $d(x, y') \ge \frac{3}{4}R' > R'/16 + R'/2 \ge R'/16 + t$, and thus $|S'| \le (1-\delta)|S|$. ☐

The $O(1)$-ANN algorithm. Given a query q against X, the algorithm traverses the DAG along the directed edges, constructing along the way a set $L \subset X$. Each step in the traversal will add to L at most two points; namely, at a vertex $\langle y, R\rangle$ the point y is added to L, and if this vertex has a ring-separator (x, t), then x is added to L as well. Once the traversal ends, the algorithm reports the point in L which is closest to q.

The DAG traversal starts at the vertex $\langle \hat{y}, \hat{R}\rangle$ where $\hat{R} \stackrel{\text{def}}{=} \max \Gamma \ge 2\,\mathrm{diam}(X)$ and $\hat{y}$ is the single point in $Y_{\hat{R}}$. To specify how one step in the traversal proceeds, denote the current vertex by $\langle y, R\rangle$. If $S = B_X(y, 2R)$ contains only one point, the traversal ends. Otherwise, let (x, t) be the enhanced δ-ring-separator for this vertex. We now have two cases, depending on $d(x, q)$.

- If $d(x, q) \le 2t$, we examine family (a) of outgoing edges of $\langle y, R\rangle$. If none of them goes to a vertex $\langle z, t/32\rangle$ for which $d(z, q) \le \beta t/32$ (this includes the case $t/32 \notin \Gamma$), the traversal ends; otherwise, we proceed along one such edge, breaking ties arbitrarily.

- If $d(x,q) > 2t$, we compute the value R' which is a power of 2 such that $R' < d(x,q) \le 2R'$, and examine family (b) of outgoing edges of $\langle y, R\rangle$. If none of them goes to a vertex $\langle y', R'/32\rangle$ for which $d(y',q) \le \beta R'/32$ (this includes the case $R'/32 \notin \Gamma$), the traversal ends; otherwise, we proceed along one such edge, breaking ties arbitrarily.

Finally, if the vertex $\langle z, t/32\rangle$ above is implicit and the edge to it is redirected to some $\langle z, \tilde{t}\rangle$, then we still check only whether $d(z,q) \le \beta t/32$. Similarly, if the $\langle y', R'/32\rangle$ as above is implicit and the edge to it is redirected to another vertex then we still check only whether $d(y',q) \le \beta R'/32$.

We are now ready to complete the proof of Theorem 3.1. Below, we bound the space consumption (Lemma 3.6) and the query time (Lemma 3.7), and show that the algorithm's output is indeed an $O(1)$ approximation (Lemma 3.8).

Lemma 3.6. *The above algorithm consumes $O(n^3)$ space.*

Proof. The query algorithm only needs to have access to the DAG, and we already argued above that the DAG contains $O(n^2)$ vertices. It thus remains to upper bound the number of edges in the DAG. Fix a single vertex $\langle y, R\rangle$ and let us bound the number of its outgoing edges. If $S = B_X(y, 2R)$ contains only one point, the vertex has no outgoing edges. Otherwise, let (x,t) be the enhanced δ-ring-separator for S. The number of outgoing edges in family (a) is clearly bounded by the size of $Y_{t/32} \subseteq X$, and thus by n. To bound the number of outgoing edges in family (b), observe that for every point $y' \in X$, there are only $O(1)$ values $R' \in \Gamma$ for which there is an edge to $\langle y', R'\rangle$, because we must have $R' = \Theta(d(x,y'))$. We conclude that there are only $O(n)$ outgoing edges from every vertex, and hence the total number of edges in the DAG is $O(n^3)$. □

Lemma 3.7. *The above algorithm runs, for any query q, in time $2^{O(\dim(X))} \log n$. In particular, this bounds the number of distance computations.*

Proof. By Lemma 3.5, any traversal that proceeds along directed edges of the DAG has length at most $2^{O(\dim(X))} \log n$. Each step in the traversal computes $d(y,q)$ to decide between the two cases. In the first case, we enumerate over family (a) of edges outgoing to vertices of the form $\langle z, t/32\rangle$, computing each time $d(z,q)$, and determine along which edge (if at all) to proceed in the traversal. Using Lemma 1.2 we can bound the number of edges in this family, and hence the running time of this step of the traversal, by $2^{O(\dim(X))}$. In the second case, we compute the appropriate value R', and retrieve the corresponding list of edges from family (b), namely, those outgoing to vertices of the form $\langle y', R'/32\rangle$. There are only $O(n)$ values of R' for which this list is nonempty; binary search would locate the desired one in time $O(\log n)$, but we can do it in $O(1)$ time using hashing, see e.g. [MR95, Section 8.5]. By doing one distance computation for each edge in the list (if the edge goes to $\langle y', R'/32\rangle$ we compute $d(y',q)$), we can determine along what edge (if at all) to proceed in the traversal. Again, using Lemma 1.2 we can bound the number of edges in this list, and hence the running time of this step of the traversal, by $2^{O(\dim(X))}$.

We conclude that each step of the DAG traversal runs in time $2^{O(\dim(X))}$, and therefore the overall running time is at most $2^{O(\dim(X))} \log n$. □

Lemma 3.8. *The above algorithm outputs an $O(1)$-approximate nearest neighbor to any query q.*

Proof. Let $a^* \in X$ be the real nearest neighbor to q, i.e., $d(q,X) = d(q,a^*)$. Consider the DAG traversal made by the algorithm. Let $\langle y_i, R_i\rangle$ be the ith vertex in the traversal, and let $S_i = B_X(y_i, 2R_i)$ be the point set associated with it. To ease notation (and only for the sake of analysis), we let this sequence contain also the implicit vertices which the traversal did not really visit. That is, an implicit vertex and the explicitly stored vertex the traversal was redirected will be represented here by $\langle y_i, R_i\rangle$ and $\langle y_{i+1}, R_{i+1}\rangle$, respectively, with $y_i = y_{i+1}$.

Let j be the largest (i.e., last vertex in the traversal) such that $a^* \in B_X(y_j, \gamma R_j)$. This j is well-defined because for the first vertex in the traversal we have $\gamma R_1 = \gamma \hat{R} \geq \mathrm{diam}(X)$.

We can rule out two cases that are easy to handle. First, if $|S_j| = 1$, then since $a^* \in B_X(y_j, \gamma R_j) \subseteq S_j$, the traversal must end at $\langle y_j, R_j \rangle$ with a^* being added to L, and hence the algorithm returns the exact nearest neighbor to q. So in the rest of the proof we assume that $|S_j| > 1$. Second, if $\langle y_j, R_j \rangle$ is an implicit vertex then we know that the traversal is redirected to vertex $\langle y_{j+1}, R_{j+1} \rangle$, which satisfies $y_j = y_{j+1}$ and furthermore

$$B_X(y_j, 2R_j) = B_X(y_j, R_j) = B_X(y_j, R_j/2) = \cdots = B_X(y_{j+1}, R_{j+1}). \tag{3}$$

But by the definition of j, $a^* \in B_X(y_j, \gamma R_j) \subseteq B_X(y_j, 2R_j)$ and $a^* \notin B_X(y_{j+1}, \gamma R_{j+1}) \supseteq B_X(y_{j+1}, R_{j+1})$, and this contradicts (3). So from now on we assume that $\langle y_j, R_j \rangle$ is not an implicit vertex.

Consider the algorithm's operation upon entering $\langle y_j, R_j \rangle$. To ease notation, denote $y = y_j$, $R = R_j$ and $S = S_j$. We assumed $|S| \geq 2$, hence the algorithm computes an enhanced δ-ring-separator (x, t) for S. Recall that $t \leq \mathrm{diam}(S) \leq 4R$. We have two cases, depending on $d(x, q)$.

- Suppose $d(x, q) \leq 2t$. We can assume $d(q, a^*) < t/300$, as otherwise $x \in L$ is a 600-approximate nearest neighbor. We may assume further that $t/32 \in \Gamma$, as otherwise $t/32 < d_{\min}/128$ and $d(x, q) \leq 2t < d_{\min}/2$, hence x is an exact nearest neighbor to q. We now aim to show that the traversal proceeds from this vertex to a vertex that contradicts the definition of j. As a net, $Y_{t/32}$ must contain a point z such that $d(a^*, z) \leq t/32$. Notice that there must be an outgoing edge from $\langle y, R \rangle$ to $\langle z, t/32 \rangle$, because (1) holds:

  $$\begin{aligned} d(x, z) &\leq d(x, q) + d(q, a^*) + d(a^*, z) \leq 2t + t/300 + t/32 < 3t, \\ d(y, z) &\leq d(y, a^*) + d(a^*, z) \leq \gamma R + R/8 \leq \alpha R. \end{aligned}$$

  Furthermore, the traversal is allowed to proceed to $\langle z, t/32 \rangle$, because $d(z, q) \leq d(z, a^*) + d(a^*, q) \leq t/32 + t/300 < \beta t/32$.

  Due to ties, the next vertex in the DAG traversal, $\langle y_{j+1}, R_{j+1} \rangle$, does not have to be $\langle z, t/32 \rangle$. But by definition, $R_{j+1} = t/32$ and $d(y_{j+1}, q) \leq \beta t/32$. Thus, $d(y_{j+1}, a^*) \leq d(y_{j+1}, q) + d(q, a^*) \leq \beta t/32 + t/300 \leq \frac{5}{4} t/32 = \gamma R_{j+1}$, which contradicts the definition of j.

- Suppose $d(x, q) > 2t$ and let R' be the a power of 2 such that $R' < d(x, q) \leq 2R'$. We can assume that $d(q, a^*) < R'/300$, as otherwise $x \in L$ is a 600-approximate nearest neighbor. We may assume further that $R'/32 \in \Gamma$, as otherwise $R'/32 < d_{\min}/128$ and $d(x, q) \leq 2R' < d_{\min}/2$, hence x is an exact nearest neighbor to q. We now aim to show that the traversal proceeds from this vertex to a vertex that contradicts the definition of j. As a net $Y_{R'/32}$ must contain a point y' such that $d(a^*, y') \leq R'/32$. It follows that there is an outgoing edge from $\langle y, R \rangle$ to $\langle y', R'/32 \rangle$, because (2) holds:

  $$\begin{aligned} d(x, y') &\leq d(x, q) + d(q, a^*) + d(a^*, y') \leq 2R' + R'/300 + R'/32 \leq \frac{9}{4} R', \\ d(x, y') &\geq d(x, q) - d(q, y') \geq R' - (R'/300 + R'/32) \geq \frac{3}{4} R', \\ d(y, y') &\leq d(y, a^*) + d(a^*, y') \leq \gamma R + R'/32 \leq \gamma R + 8R/32 \leq \alpha R. \end{aligned}$$

  Furthermore, the traversal is allowed to proceed to $\langle y', R'/32 \rangle$, because $d(y', q) \leq d(y', a^*) + d(a^*, q) \leq R'/32 + R'/300 < \beta R'/32$.

Due to ties, the next vertex in the DAG traversal, $\langle y_{j+1}, R_{j+1}\rangle$, does not have to be $\langle y', R'/32\rangle$. But by definition, $R_{j+1} = R'/32$ and $d(y_{j+1}, q) \leq \beta R'/32$. Thus, $d(y_{j+1}, a^*) \leq d(y_{j+1}, q) + d(q, a^*) \leq \beta R'/32 + R'/300 \leq \frac{5}{4}R'/32 = \gamma R_{j+1}$, which contradicts the definition of j.

We conclude that in either case, the algorithm outputs an $O(1)$-approximate nearest neighbor of q. $\square$

Acknowledgments

The authors would like to thank the Weizmann Institute, and in particular Uriel Feige, for hosting them in December 2003.

References

[Ass83] P. Assouad. Plongements lipschitziens dans $\mathbf{R}^n$. *Bull. Soc. Math. France*, 111(4):429–448, 1983.

[BLMN03] Y. Bartal, N. Linial, M. Mendel, and A. Naor. On metric Ramsey-type phenomena. *Annals of Mathematics*, 2003. To appear.

[Cla99] K. L. Clarkson. Nearest neighbor queries in metric spaces. *Discrete Comput. Geom.*, 22(1):63–93, 1999.

[GKL03] A. Gupta, R. Krauthgamer, and J. R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Proceedings of the 44th annual Symposium on the Foundations of Computer Science*, 2003.

[Hei01] J. Heinonen. *Lectures on analysis on metric spaces*. Universitext. Springer-Verlag, New York, 2001.

[HKMR04] K. Hildrum, J. Kubiatowicz, S. Ma, and S. Rao. A note on finding nearest neighbors in growth-restricted metrics. In *Proceedings of the 15th annual ACM-SIAM Symposium on Discrete Algorithms*, 2004.

[HP01] S. Har-Peled. A replacement for Voronoi diagrams of near linear size. In *42nd IEEE Symposium on Foundations of Computer Science*, pages 94–103. IEEE Computer Soc., 2001.

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *30th Annual ACM Symposium on Theory of Computing*, pages 604–613, May 1998.

[KKL03] S. Kakade, M. Kearns, and J. Langford. Exploration in metric state spaces. In *Proceedings of the 20th International Conference on Machine Learning*, 2003.

[KL04] R. Krauthgamer and J. R. Lee. Navigating nets: Simple algorithms for proximity search. In *Proceedings of the 15th annual ACM-SIAM Symposium on Discrete Algorithms*, 2004.

[KOR98] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. In *30th Annual ACM Symposium on the Theory of Computing*, pages 614–623, 1998.

[KR02] D. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *34th Annual ACM Symposium on the Theory of Computing*, pages 63–66, 2002.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[Tal04] K. Talwar. Bypassing the embedding: Approximation schemes and distance labeling schemes for growth restricted metrics. To appear in the procedings of the *36th annual Symposium on the Theory of Computing*, 2004.

A Triangle inequality for the lower bound

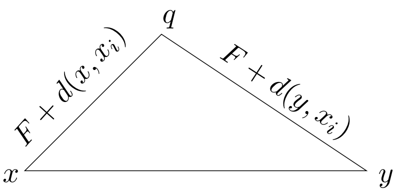
Lemma A.1. *For every $i \in \{1, \dots, k\}$, the space $(X \cup \{q_i\}, d)$ satisfies the triangle inequality.*

Proof. Fix $x, y \in X$. We need to verify that all triangle inequalities on q, x, y are satisfied, i.e., that

$$d(x,y) \leq d(x,q) + d(q,y) \quad \text{and} \quad |d(x,q) - d(y,q)| \leq d(x,y).$$

To ease notation, let $S' = \{x \in X : d(x,S) \leq \frac{1}{2}d_{\min}\}$. Then each of x,y can be either in (i) $B(x_i, \frac{1}{2}d_{\min})$, (ii) $S' \setminus B(x_i, \frac{1}{2}d_{\min})$, or (iii) in $X \setminus S'$. We have to consider six cases for x, y (because of symmetry). In the sequel, let $F = \frac{1}{2}d_{\max} + \frac{1}{4}d_{\min}$.

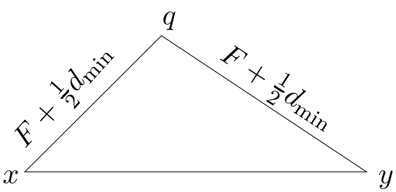
1. $x, y \in B(x_i, \frac{1}{2}d_{\min})$:



$$d(x,y) \leq \tfrac{1}{2}d_{\min} + \tfrac{1}{2}d_{\min} < 2F \ \checkmark$$

$$\Delta = |d(x,x_i) - d(y,x_i)| \leq d(x,y) \ \checkmark$$

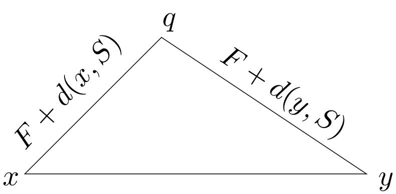
2. $x, y \in S' \setminus B(x_i, \frac{1}{2}d_{\min})$:



$$d(x,y) \leq \tfrac{1}{2}d_{\min} + d_{\max} + \tfrac{1}{2}d_{\min} = 2F + \tfrac{1}{2}d_{\min} \ \checkmark$$

$$\Delta = 0 \ \checkmark$$

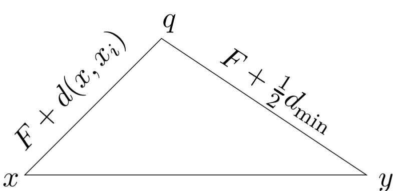
3. $x, y \notin S'$:



$$d(x,y) \leq d(x,S) + d_{\max} + d(y,S) < d(x,S) + 2F + d(y,S) \ \checkmark$$

$$\Delta = |d(x,S) - d(y,S)| \leq d(x,y) \ \checkmark$$

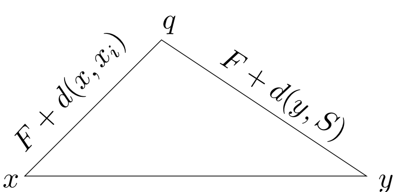
4. $x \in B(x_i, \frac{1}{2}d_{\min})$, $y \in S' \setminus B(x_i, \frac{1}{2}d_{\min})$:



$$d(x,y) \leq \tfrac{1}{2}d_{\min} + d_{\max} + \tfrac{1}{2}d_{\min} = 2F + \tfrac{1}{2}d_{\min} \ \checkmark$$

$$\Delta = \tfrac{1}{2}d_{\min} - d(x,x_i) \leq d(y,x_i) - d(x,x_i) \leq d(x,y) \ \checkmark$$

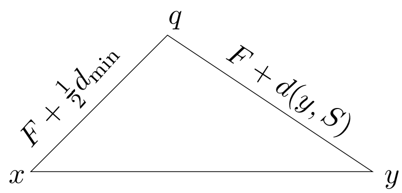
5. $x \in B(x_i, \frac{1}{2}d_{\min})$, $y \notin S'$:



$$d(x,y) \leq \tfrac{1}{2}d_{\min} + d_{\max} + d(y,S) = 2F + d(y,S) \ \checkmark$$

$$\Delta = d(x,x_i) - d(y,S) = d(x,S) - d(y,S) \leq d(x,y) \ \checkmark$$

6. $x \in S' \setminus B(x_i, \frac{1}{2}d_{\min})$, $y \notin S'$:



$$d(x, y) \leq \frac{1}{2}d_{\min} + d_{\max} + d(y, S) = 2F + d(y, S) \checkmark$$

$$\Delta = d(y, S) - \frac{1}{2}d_{\min} \leq d(y, S) - d(x, S) \leq d(x, y) \checkmark$$

$\square$