Nearest Neighbor Queries in Metric Spaces

Kenneth L. Clarkson
Bell Laboratories, Lucent Technologies
Murray Hill, New Jersey 07974
`clarkson@research.bell-labs.com`
`http://cm.bell-labs.com/who/clarkson/`

Abstract

Given a set S of n sites (points), and a distance measure d, the *nearest neighbor searching* problem is to build a data structure so that given a query point q, the site nearest to q can be found quickly. This paper gives data structures for this problem when the sites and queries are in a metric space. One data structure, $D(S)$, uses a divide-and-conquer recursion. The other data structure, $M(S,Q)$, is somewhat like a skiplist. Both are simple and implementable. The data structures are analyzed when the metric space obeys a certain sphere-packing bound, and when the sites and query points are random and have distributions with an exchangeability property. This property implies, for example, that query point q is a random element of $S \cup \{q\}$. Under these conditions, the preprocessing and space bounds for the algorithms are close to linear in n. They depend also on the sphere-packing bound, and on the logarithm of the *distance ratio* $\Upsilon(S)$ of S, the ratio of the distance between the farthest pair of points in S to the distance between the closest pair. The data structure $M(S,Q)$ requires as input data an additional set Q, taken to be representative of the query points. The resource bounds of $M(S,Q)$ have a dependence on the distance ratio of $S \cup Q$. While $M(S,Q)$ can return wrong answers, its failure probability can be bounded, and is decreasing in a parameter K. Here $K \leq |Q|/n$ is chosen when building $M(S,Q)$. The expected query time for $M(S,Q)$ is $O(K \log n) \log \Upsilon(S \cup Q)$, and the resource bounds increase linearly in K. The data structure $D(S)$ has expected $O(\log n)^{O(1)}$ query time, for fixed distance ratio. The preprocessing algorithm for $M(S,Q)$ can be used to solve the all-nearest-neighbor problem for S in $O(n(\log n)^2(\log \Upsilon(S))^2)$ expected time.

1 Introduction

This paper addresses algorithmic questions related to the *post office* or *nearest neighbor* problem:

> Let (V,d) be a metric space, where V is a set and d is a distance measure on V. Given a set $S \subset V$ of n sites (points), build a data

structure so that given a query point $q \in V$, the nearest site to q can be found quickly.

Two data structures are given here for this problem. One data structure, $M(S,Q)$, requires an additional set Q of m points, taken to be representative of typical query points. The data structure $M(S,Q)$ may fail to return a correct answer, but the failure probability can be made arbitrarily small. This decrease requires a proportional increase in query time and data structure space. The other data structure, $D(S)$, always returns a correct answer and does not need the set Q, but its provable resource bounds are worse.

The data structure $M(S,Q)$ has been implemented, with some preliminary tests. For example, when the points of S, Q, and q are uniformly distributed in a square, and the distance measure is Euclidean (ℓ_2), a version of the data structure gives the correct answer for all of 500 tests. For this version, searching requires about 21 distance evaluations for $|S| = 2000$, and the space required is about 8 integers/site. (The data structure implemented uses $K = 1000$, $\gamma = 1.2$, and uses an "early termination" to a collection of sites to be searched by brute force.) Note that the algorithm uses the distance measure as a "black box." With 4000 similarly distributed points in $\Re^{20}$, an average search time of 604 distance evaluations gives a site for which about 0.6 sites are closer, on average, and at an average distance 2% farther than the true nearest neighbor distance.

The provable bounds require some general conditions on the data, and on the metric space. The failure probability bounds hold for $M(S,Q)$ when q is a random element of $Q \cup \{q\}$; we'll call this requirement on q and Q an *exchangeability* condition. When Q and q are each generated by some random process, each with their probability distribution, the exchangeability condition means that if we are given Q and q, and then take a random element of $Q \cup \{q\}$, then that element will have the same probability distribution as q.

The bounds on the query time hold when Q, S, and $\{q\}$ satisfy the exchangeability condition that they are each random subsets of $Q \cup S \cup \{q\}$.

This condition is satisfied when Q, S, and $\{q\}$ are random subsets of some $U \subset V$, or when the points in these sets are generated by random, independent random variates with the same probability distribution. (The probability distribution is arbitrary; in particular, it is not necessarily uniform.)

The proofs of the bounds also require that the metric spaces have certain *nearest neighbor bounds*, which are implied by *sphere-packing* bounds; such properties are described in §3.4. In particular, $\Re^k$ has these properties under L_p metrics. The constants associated with these properties appear in the bounds for the algorithms, and are in general exponential in k.

Many of the provable bounds also involve the *distance ratio*, denoted $\Upsilon(T)$ for a set T, which is the ratio of the distance between the farthest pair of points in T to distance between the closest pair in T. The quantity $\Upsilon(S)$ appears in bounds for $D(S)$, and the quantity $\Upsilon(S \cup Q)$ appears in bounds for $M(S,Q)$. The dependence is relatively mild, generally $O(\log \Upsilon)^{O(1)}$. It is reasonable in practice to assume the rough relation $\Upsilon = n^{O(1)}$, implying that "all logs are equal."

The preprocessing for $D(S)$ needs

$$O(n)(\lg n)^{O(\lg\lg \Upsilon(S))}$$

expected time, resulting in a data structure that answers queries q with $q \in_R S \cup \{q\}$ in expected

$$O((\lg n)^{O(1)+2\lg\lg(\Upsilon(S))})$$

time, and needing

$$O(n(\lg n)^{O(1)+2\lg\lg(\Upsilon(S))})$$

expected space. (The dependence on the sphere-packing bounds for the metric space is suppressed here.)

The preprocessing for $M(S, Q)$ needs

$$O(Kn(\log n)^2(\log \Upsilon(S \cup Q)^2)$$

expected time, yielding a data structure that answers queries in

$$O(K \ln n) \lg \Upsilon(S \cup Q)$$

expected time, with failure probability $O(\log^2 n)/K$, and needing space

$$O(Kn \log \Upsilon(S \cup Q)).$$

Either preprocessing method also solves the *all-nearest-neighbor* problem: find, for each site in S, the nearest other site. Algorithms for the all-nearest-neighbor problem that need $O(n \log n)$ time have been known since 1983 for $\Re^k$[Cla83, Vai89], but the algorithms given here are the first with near-linear bounds that use the distance measure alone; in particular, they do not use quadtrees or quadtree-like spatial subdivisions as in previous work.

1.1 Why metric spaces?

Recall that if (V, d) is a metric space, then for all $a, b, c \in V$, $d(a, a) = 0$, $d(a, b) = d(b, a)$, and the triangle inequality $d(a, c) \leq d(a, b) + d(b, c)$ holds.

The approach taken here is to find algorithms that can be applied to general metric spaces, but that have provable properties for important special cases. There are several reasons for considering the nearest neighbor problem in this generality. While closest-point problems have applications in statistics, data compression, information retrieval, and other areas, many such applications are in a high-dimensional setting, for which almost all known solutions either give a slowdown over the naive algorithm, or use too much space. On the other hand, high dimensional data often has useful structure. It may lie in a lower-dimensional hyperplane or flat, or manifold, for example. Still, such structure is not always readily apparent, as when the data lies in a manifold that isn't flat, or has fractal properties[FC96]. Hence it is desirable to seek algorithms whose complexity does not depend strictly on the dimension, or on using most coordinates, but rather on intrinsic properties of the data.

There are other reasons to consider general metric spaces: some distance measures are not applied to spaces with real coordinates, such as edit distance on strings. (We should note, however, that strings do not seem to have a useful sphere-packing bound.) Moreover, some spaces of sites and queries have coordinates that are restricted to small sets, so that intuitions or constructions from Euclidean space are not necessarily helpful. If all coordinates are 0 or 1, then distance evaluations between points are faster on real machines, using bit-twiddling, than other coordinate-wise operations. An algorithm that uses the distance measure, and only as a "black box," will probably not be too complicated: there are no operations on which such complexity can depend. Finally, from a theoretical point of view, it seems reasonable to strip the problem down to its essentials, and find the minimal properties needed for fast algorithms.

Algorithms operating in general metric spaces have been considered for some time,[FS82] and there are some more recent contributions as well[Bri95, Uhl91, Yia93]. Most of the proposed data structures are trees, and some are variations on the kd-tree, perhaps the most important practical method for nearest neighbor searching in $\Re^k$, for small dimension k. The data structure $D(S)$ uses divide-and-conquer in a somewhat similar way; in contrast, the data structure $M(S, Q)$ is somewhat akin to a skip list.

1.2 Why Q?

The need for the set Q for data structure $M(S, Q)$ contrasts unfavorably with most earlier work on the post office problem, where there is no such limitation. Also, the analysis requirement that q have the same distribution as Q and S is restrictive. However, in some significant applications, such as vector quantization or nearest neighbor classification, the queries have the same or similar distribution as the sites, and so the set Q is simply more training data.

In other cases, say for coordinate spaces, the representative queries that are needed can be at the least be generated uniformly within a region containing the sites. Also, such a set Q is available in a "deferred data structuring" setting[MR86]. Here no preprocessing is done, and a single query is answered with the naive algorithm. As more queries are answered, however, the set Q is built up, and the data structures described here can be used to speed up subsequent queries, under the assumption that future queries have a similar distribution to past ones.

2 Outline

After some motivating remarks, the next section describes $M(S, Q)$, and gives failure probability bounds, first for general metric spaces, and then for spaces that have a certain γ-*dominator* bound. Such a bound is implied by sphere-packing bounds, discussed in §3.4, which also discusses γ-*nearest neighbor* bounds, also implied by sphere-packing bounds. Such γ-nearest neighbor bounds are used in §3.5, and in the following subsection addressing space bounds for $M(S, Q)$.

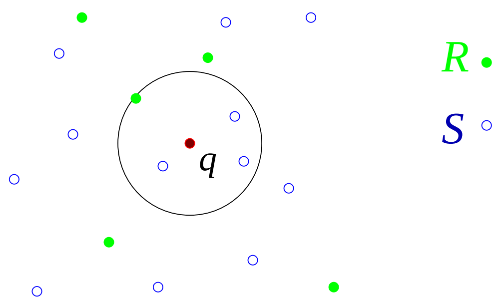

Figure 1: The nearest neighbor ball $B(q, R)$ of q.

A provably fast preprocessing algorithm for $M(S, Q)$ is discussed in §4, including some interesting relations holding for γ-nearest neighbors of β-nearest neighbors. The data structure $D(S)$ is discussed in §4, which also includes the use of the data structure for *inverse* queries, where the sites that have a given point as nearest neighbor are wanted.

3 Data structure $M(S, Q)$

3.1 Motivation

The original idea for the data structure goes back to a paper on the Euclidean version of the problem[Cla88], and probably further than that. The idea is to use a small subset $R \subset S$ to bound search: for a given query q, the distance of q to its nearest neighbor in R gives an upper bound on the distance to its nearest neighbor in S. That is, the nearest site to q in S is contained within the *nearest neighbor ball* $B(q, R)$ of q with respect to R.

Definition: $B(q, R)$**.** For metric space (V, d) with $R \subset V$, the *nearest neighbor ball* of q with respect to R is

$$\{x \in V \mid d(q, x) \le d(q, y) \text{ for all } y \in R \setminus \{q\}\}.$$

See Figure 3.1.

For a site $p \in R$, let $C(p, R)$ be the union of balls $B(q, R)$ over all (potential) query points q with p closest in R. Then for a given query point q, the knowledge that q is in the Voronoi region $\mathrm{Vor}(p)$ of p, that is, that p is closest to q in R, allows the search for nearest sites to q to be restricted to $C(p, R) \cap S$. (See Figure 2.)

This suggests a recursive construction, building a data structure for each $C(p, R) \cap S$ in a similar fashion. This is the idea of "RPO trees"[Cla88]. Such a construction is predicated on the hope that each $C(p, R) \cap S$ is small; this is false in general. However, in the Euclidean case, when R is a *random* subset of

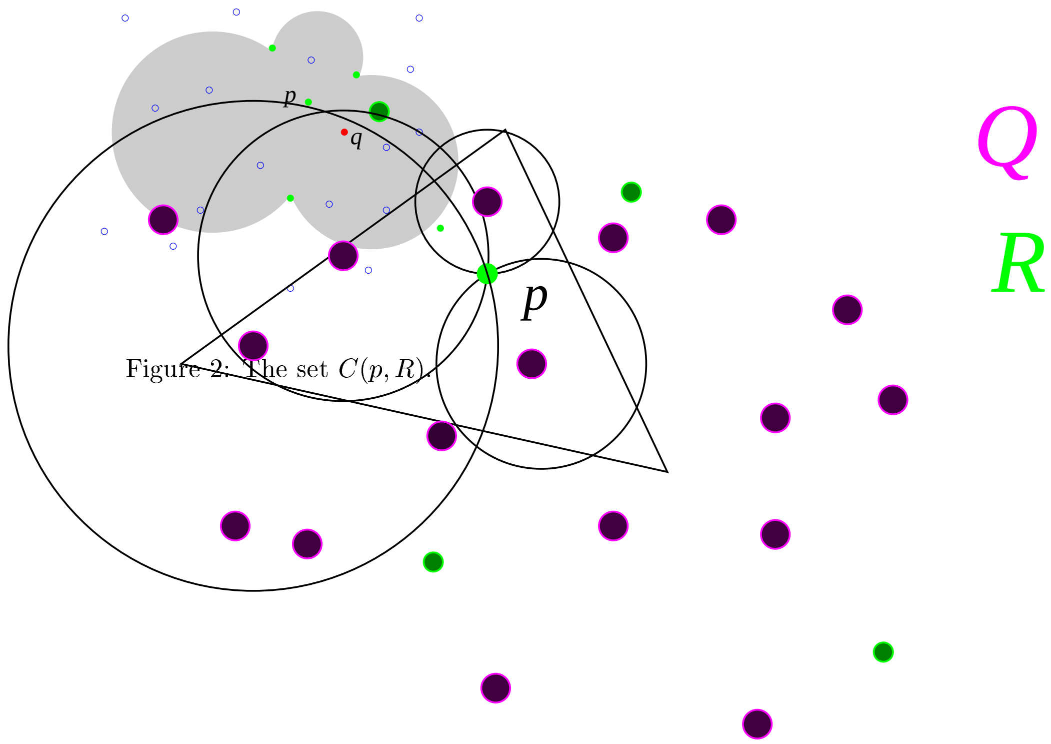

Figure 2: The set $C(p, R)$.

Figure 3: An approximation to $C(p, R)$; $\mathrm{Vor}(p)$ is within the triangle.

S, and the Voronoi region $\mathrm{Vor}(p)$ is split up into combinatorially simple regions, then such a region T will have $C(T) \cap S$ small, with high probability.[Cla88]

However, the latter construction seems to need a triangulation of the Voronoi diagram of R, which is hard to extend to arbitrary metric spaces. Here we take a simpler approach, and approximate $C(p, R)$ with the union

$$C'(p, R) \equiv \cup_{q' \in \mathrm{Vor}(p) \cap Q} B(q', R).$$

of nearest neighbor balls of members of Q. That is, we approximate the construction of the region $C(p, R)$ by approximating the Voronoi region of p with $\mathrm{Vor}(p) \cap Q$. Our hope is that for most of the query points q that will be encountered, if q is in $\mathrm{Vor}(p)$, then the set $C'(p, R) \cap S$ will contain the nearest site to q. (See Figure 3.)

To compensate for using $\mathrm{Vor}(p) \cap Q$ to find $C'(p, R)$, rather than $\mathrm{Vor}(p)$, we will expand each nearest neighbor ball by a given factor γ: we approximate $C(p, R)$ by $C'_\gamma(p, R)$, which is the union of the balls $B_\gamma(q, R)$. Such a ball is the expansion of $B(q, R)$ by the factor γ.

Definition: $B_\gamma(q, R)$**.** For metric (V, d) with $R \subset V$, the γ*-nearest neighbor*

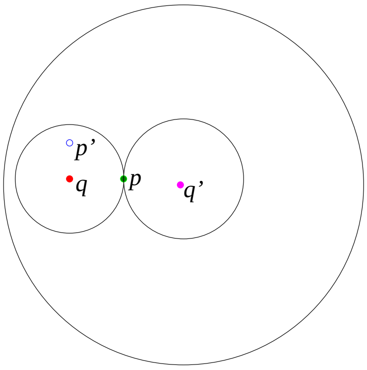

Figure 4: The 3-nearest neighbor ball of q' contains the nearest neighbor ball of q.

ball of q with respect to R is

$$\{x \in V \mid d(q,x) \leq \gamma d(q,y) \text{ for all } y \in R \setminus \{q\}\}.$$

Heuristically, this seems like a reasonable way to be more likely to get a correct answer. We can say more, however, as the following lemma states.

Lemma 1 *Suppose measure d satisfies the triangle inequality. If points q and q' have p nearest in R, and $d(q,p) < d(q',p)$, then $B(q,R) \subset B_3(q',R)$.*

PROOF. Suppose p' is closer to q than p. Then

$$d(p',q') \leq d(p',q) + d(q,p) + d(p,q') \leq 3d(p,q').$$

□

Thus, if for some query point q, the construction of $C_3'(p,R)$ included a point q' in the $\mathrm{Vor}(p) \cap S$, and $d(q',p) > d(q,p)$, then $C_3'(p,R) \cap S$ contains the nearest site to q. (See Figure 4.)

When the query points are distributed similarly to the points of Q, the probability that some $C_3'(p,R)$ will contain $B(q,R)$ is bounded by $r/|Q|$: roughly, a random member of Q has this probability of being "covered" in this sense, and the query point is will behave the same way.

3.2 The data structure

The data structure $M(S,Q)$ is based on these ideas, but uses a more incremental approach.

First, some definitions.

Definition: Nearest Neighbors. For metric space (V, d), set $R \subset V$, point $q \in V$, the *nearest neighbor distance* of q with respect to R is

$$d(q, R) \equiv \min_{p \in R \setminus \{q\}} \{d(q, p)\}.$$

A point $p \in R$ realizing that distance is a *nearest neighbor* of q in R.

Definition: γ-Nearest Neighbors. Say that $p \in V$ is a γ*-nearest neighbor* of $q \in V$ with respect to R if $d(q, p) \leq \gamma d(q, R)$.

That is, p is a γ-nearest neighbor of q with respect to R if and only if $p \in B_\gamma(q, R)$.

It will be convenient at times to use still another notation for proximity.

Definition: $q \xrightarrow{\gamma} p$. In a metric space (V, d), for $q, p \in V$, and $R \subset V$, let $q \xrightarrow{\gamma} p$ denote the condition that p is a γ-nearest neighbor of q with respect to R, that is, $d(q, p) \leq \gamma d(p, R)$, and so $p \in B_\gamma(q, R)$. Also write $p \xleftarrow{\gamma} q$ if and only if $q \xrightarrow{\gamma} p$.

Armed with all these ways of saying the same thing, here is a succinct definition of $M(S, Q)$.

Construction of $M(S, Q)$. The data structure $M(S, Q)$ for nearest neighbor searching is, for each site $p_j \in S$, a list of sites A_j. The data structure has a real parameter γ, picked before construction.

Let $(p_1, p_2, \ldots, p_n)$ be a random permutation of S, and let $R_i \equiv \{p_1, p_2, \ldots, p_i\}$, so that R_i is a random subset of S. Shuffle Q with a random permutation as well, so that Q_j is a random subset of Q of size j, for $j = 1 \ldots m = |Q|$. It will be helpful to define $Q_j \equiv Q$ for $j > m$.

Define A_j as

$$\{p_i \mid i > j, \text{there exists } q \in Q_{Ki} \text{ with } p_j \xleftarrow{1} q \xrightarrow{\gamma} p_i, \text{with respect to } R_{i-1}\}.$$

The sites in A_j are in increasing order of index i.

This is the whole definition of $M(S, Q)$.

While it is satisfying that $M(S, Q)$ can be so succinctly defined, a more detailed discussion may be helpful. Consider the subsets R_i to be built incrementally, by picking p_i randomly from $S \setminus R_{i-1}$, and adding it to R_{i-1}, yielding R_i. (Hereafter the numbering of the sites will be fixed in this random order.)

Each list A_j starts out empty. When adding site p_i to make R_i, append p_i to A_j if there is some $q \in Q_{Ki}$ with p_j nearest to q in R_{i-1}, and p_i is a γ-nearest neighbor of q. That is, for $q \in Q$, maintain the nearest neighbor to q in R_j, for $j = 1 \ldots n$. When adding p_i, check each $q \in Q$, and record p_i as nearest where appropriate, and also add p_i to A_j if $q \in Q_{Ki}$ and p_i is γ-nearest to q.

What does this construction mean? Note that in the terminology of the last subsection, we add p_i to A_j when $p_i \in C'_\gamma(p_j, R_{i-1})$, our approximation to the set of sites that may be nearer to some query point q if q has p_j nearest in R_{i-1}.

Consider the special case where $\gamma = 1$, $S \subset \Re^k$, $Q = \Re^k$, $K \to \infty$, and the distributions of S and Q are uniform in a box. Here the construction puts p_i on A_j just when p_i takes away part of the Voronoi region of p_j, as witnessed by a member of Q_{Ki}; for large enough K, p_i is a Delaunay neighbor of p_j in R_i.

The Search Procedure for $M(S,Q)$. The search procedure is as follows: given query point q, start with site p_1 as the candidate closest site to q. Walk down A_1, until a site p_j closer to q than p_1 is found. Now p_j is the candidate closest site to q; do a similar walk down A_j. Repeat until the A_k list for some site p_k is searched, and no site on that list is closer than p_k. Terminate the search and return p_k as closest site.

This is the search procedure.

Note that, when $K = \infty$, so that $Q_{Ki} = Q$ for all i, this search returns by construction the correct closest site for all elements of Q. When the search procedure is applied to some $q \in Q$, if p_i appears as a candidate closest site to q, then p_i is closest to q in R_i.

The query time and the probability of returning the correct nearest site increase with K and γ. The average number of members of Q_{Ki} with p_j nearest in R_i is K.

There are two uses of randomness here: for picking R_i, and for picking Q_{Ki}. The use of random subsets of the sites is similar to its use in some other randomized geometric algorithms[Mul93, Cla92], and helps to speed up the answering of queries. The random subset Q_{Ki}, on the other hand, serves as a proxy for members of some universe of queries, and aids correctness; intuitively, any possible query point q will have a member q' of Q_{Ki} not too far away, and hence a site nearest to q will be near to q'.

3.3 Failure probability analysis

The following bound on the failure probability holds without any restrictions on the metric space.

Theorem 2 *Suppose Q and q are such that q is a random element of $Q \cup \{q\}$. Assume $Kn < m = n^{O(1)}$. If $M(S,Q)$ is built with $\gamma = 3$, then the probability that it fails to return a nearest site to q in S is $O(\log^2 n)/K$.*

The proof depends on Lemma 1 and the following lemma.

Lemma 3 *Define $Q \equiv Q_{Ki} \cup \{q\}$. Assume $Kn < m = n^{O(1)}$. With probability $1 - 1/n^2$, for every $q' \in Q'$, the number of sites closer to q' than its nearest neighbor in R_i is $\kappa = O(\log n)n/i$.*

PROOF. The set Q' is fixed with respect to the random choice of R_i. Suppose $q' \in Q'$ and $p \in S$ are such that there are k points of S closer to q' than p is. The probability that $p \in R_i$ and also that p is the closest site in R_i to q' is no more than

$$\frac{\binom{n-k-1}{i-1}}{\binom{n}{i}} \leq \frac{i}{n} e^{-k(i-1)/(n-1)}.$$

Since the number of such pairs p and q' is no more than $(Ki+1)(n-k) < mn$, the claim follows for $k > \kappa \equiv (3\ln n + \ln m)(n-1)/(i-1) = O(\log n)n/i$. $\square$

Proof. (of Theorem 2.) Let q be a random query point. Consider the construction of the data structure when point p_{i+1} is added to make R_{i+1}, so p_{i+1} is possibly added to some A_j lists. Suppose p_j is closest to q in R_i, but p_{i+1} is closer. Then the query procedure should change the candidate closest site from p_j to p_{i+1}, but the procedure is not certain to do so if p_{i+1} is not in A_j. Conversely, if such appropriate entries in the A_j lists are present, for each j with $1 \leq j \leq n$, then the query for q will be answered correctly. This implies that the probability that the construction fails to produce a data structure that answers a query correctly is no more than the sum over i, for $i = 0 \ldots n-1$, of the probability that a failure occurs when p_{i+1} is added.

Thus we seek an upper bound on the probability that, when p_{i+1} is added:

1. With respect to R_i,
$$p_{i+1} \xleftarrow{<1} q \xrightarrow{1} p_j,$$
but
2. there is no point $v \in Q_{Ki}$ with
$$p_{i+1} \xleftarrow{3} v \xrightarrow{1} p_j.$$

Let $Q' \equiv Q_{Ki} \cup \{q\}$. For each $p_k \in R_i$, let v_k be the point in Q' that has p_k as nearest neighbor in R_i, and that has maximum distance to p_k among all such points. If (1) holds, and $q \neq v_j$, then (2) holds, by Lemma 1. (Put $q' = v_j$, $q = q$, and $p = p_j$ in the lemma.) To bound the probability that $q = v_j$ for some j, use the condition of the theorem that q is random element of Q'_{Ki}. The probability that a random element of Q' is one of the i points v_j is

$$i/|Q'| = i/(Ki+1) < 1/K,$$

and so the probability of (2), given (1), is at most $1/K$.

If some $q' \in Q'$ has more than $\kappa = (3\ln n + \ln m)(n-1)/(i-1)$ sites closer to it than its nearest neighbor in R_i, consider the addition of p_{i+1} a failure; this occurs with probability $1/n^2$, by Lemma 3. Suppose that the addition of p_{i+1} is not a failure for this reason, so that every $q' \in Q'$ has fewer than κ sites closer to it than its nearest neighbor in R_i. The probability of (1) is then κ/n, for q any member of Q'_{Ki}, and so the probability of failure when adding p_{i+1} is at most

$$1/n^2 + \frac{\kappa}{n}\frac{1}{K} = O(\log n)/iK,$$

for $i > 0$; using the trivial bound for small i, and adding this for i up to n bounds the overall failure probability at $O(\log^2 n)/K$. $\square$

A similar bound holds when $\gamma < 3$, for metric spaces that satisfy an additional condition, which can be expressed using the following definition.

Definition: γ-dominating. For $a, b, c \in V$, say that c *γ-dominates* b (with respect to a) if $B(b, \{a\}) \subset B_\gamma(c, \{a\})$.

The necessary assumption is:

Definition: γ-dominator bounds. Say that a metric space V has a *γ-dominator bound* if there is a value $\mathcal{D}_\gamma$ such that if for any $a \in V$ and $Q \subset V$, there is a set $\hat{Q} \subset Q$ of size no more than $\mathcal{D}_\gamma$, such that for every $b \in Q$, there is $c \in \hat{Q}$ such that c γ-dominates b.

In other words, a set $C(p, R)$, as described in §3.1, is contained in an approximation $C'_\gamma(p, R)$, generated by a finite set $\hat{Q}$.

For any metric space, $\mathcal{D}_3 = 1$. Also, as described in Theorem 6 below, a metric space that has a *sphere-packing* bound also has a γ-dominator bound.

For $\Re^k$ as an L_p space, the value $\mathcal{D}_\gamma$ is, in general, exponential in the dimension k. However, many "naturally occurring" point sets have structure, and that structure may imply that $\mathcal{D}_\gamma$ is much smaller than the worst-case bound for $\Re^k$.

Theorem 4 *Suppose metric space (V, d) has a γ-dominator bound, and Q and queries q are such that q is a random element of $Q \cup \{q\}$. Assume $Kn \leq m = n^{O(1)}$. Suppose $M(S, Q)$ is built for some value of γ. Then the probability that $M(S, Q)$ fails to return the nearest site in S is $O(\mathcal{D}_\gamma \log^2 n)/K$.*

Proof. The proof follows exactly as for Theorem 2, except for the probability of (2), that is, the probability that no member of Q γ-dominates q. Here the number of members of Q' needed to γ-dominate all members of Q' is $\mathcal{D}_\gamma i$, rather than just i, and so that probability that (2) holds is $\mathcal{D}_\gamma i/(Ki + 1) \leq \mathcal{D}_\gamma/K$. □

3.4 Sphere packing, γ-dominators, γ-nearest

The mysterious "γ-dominator" bound above is implied by less mysterious *sphere-packing* bounds discussed in this section.

The sphere-packing bounds also imply some *γ-nearest neighbor* bounds, which are needed to prove bounds on the query and preprocessing times. While the γ-dominator bounds suggest the interest of considering sites that are γ-nearest to points in Q, the γ-nearest neighbor bounds help show that the number of such γ-nearest sites is not too large.

The properties described in this section hold for L_p spaces. It's worth emphasizing that the properties are not used in the data structures, but only in their analysis.

Definition: Sphere packing. The space (V, d) has a *sphere-packing* bound if the following holds: for any real number ρ, there is an integer constant $\mathcal{S}_\rho$ such that for all $a \in V$ and $W \subset V$, if $|W| > \mathcal{S}_\rho$, and $d(w, a) \leq D$ for all $w \in W$ for some D, then there are $w, w' \in W$ such that $d(w, w') < D/\rho$.

There is a well-known relation between packing and covering, which we'll need; for completeness, a proof is given here.

Theorem 5 *If the metric space (V, d) has a sphere-packing bound, then for any set $W \subset V$ contained in a sphere of radius D, there is $\hat{W} \subset W$ of size $\mathcal{S}_\rho$ such that for all $w \in W \setminus \hat{W}$, $d(w, \hat{W}) < D/\rho$.*

PROOF. Build sets $W_i = \{w_1, w_2, \ldots w_i\}$ point by point, where $W_{\mathcal{S}_\rho} = \hat{W}$. Choose w_1 arbitrarily from W, and then for $i > 1$, pick for w_{i+1} the $w \in W$ which maximizes $d(w, W_i)$. Thus $d(p, W_i) \leq d(w_{i+1}, W_i)$ for all $p \in W \setminus \hat{W}$, and the distances $d(w_{i+1}, W_i)$ are nonincreasing in i. When $i = \mathcal{S}_\rho$, the sphere-packing bound implies that $d(p_{i+1}, W_i) \leq D/\rho$, and so $d(p, W_i) \leq D/\rho$ for all $p \in W$. $\square$

Theorem 6 *If the metric space (V, d) has a sphere-packing bound with constant $\mathcal{S}_\rho$, then the space has a γ-dominator bound with constant*

$$\mathcal{D}_\gamma \leq 1 + \lceil \log(\mu)/\log(1-\mu) \rceil \mathcal{S}_{1/\mu},$$

where $\mu \equiv (\gamma - 1)/(\gamma + 1)$.

PROOF. If $\gamma \geq 3$, we're done, so assume $1 < \gamma \leq 3$, and so $0 < \mu \leq 1/2$.

Given $\gamma > 1$, $a \in V$, and $Q \subset V$, let q_0 maximize $\{d(q, a) \mid q \in Q\}$. For $i = 0 \ldots k = \lceil \log(\mu)/\log(1-\mu) \rceil$, let $r_i \equiv (1-\mu)^i d(q_0, a)$. Let Q^i denote

$$\{q \in Q \mid r_{i+1} < d(q, a) \leq r_i\}.$$

Apply the previous lemma to each Q^i, with $\rho = 1/\mu$, obtaining sets $\hat{Q}^i$. Then the set $\hat{Q}$ needed for a γ-dominator bound is the union of the $\hat{Q}^i$, together with $\{q_0\}$, as we next show.

Suppose $q \in Q^k$, so $d(q, a) < r_k \leq \mu d(q_0, a)$. Then for any $z \in V$ with $d(q, z) \leq d(q, a)$, by the triangle inequality

$$d(q_0, z) \leq d(q_0, a) + d(a, q) + d(q, z) \leq (1 + 2\mu) d(q_0, a) \leq \gamma d(q_0, a),$$

and so $B(q, \{a\}) \subset \gamma B(q_0, \{a\})$.

Suppose $q \in Q^i$, so $r_{i+1} < d(q, a) \leq r_i$. Then $\hat{Q}^i$ contains a point q' such that $d(q, q') \leq alphar_i$, and so if z has $d(z, q) \leq d(a, q)$, then

$$d(q', z) \leq d(q', q) + d(q, z) \leq \mu r_i + r + i = (1 + \mu) r_i,$$

and since $d(q', a) \geq r_{i+1} = r_i(1 - \mu)$, we have

$$d(q', z) \leq \frac{1+\mu}{1-\mu} d(q', a) = \gamma d(q', a).$$

Hence $B(q, \{a\}) \subset \gamma B(q', \{a\})$.

Thus for any $q \in Q$, there is some $q' \in \hat{Q}$ with $B(q, \{a\}) \subset \gamma B(q', \{a\})$, and a γ-dominator bound holds. $\square$

We turn now to γ-nearest neighbor bounds, which weakly generalize the long-established nearest neighbor bounds of Euclidean spaces:

Definition: Nearest Neighbor bounds. Say that metric space (V, d) has a *nearest neighbor bound* if there is a constant $\mathcal{N}_1$ such that for all $a \in V$ and any $W \subset V$, $\mathcal{N}_1$ bounds the number of $b \in W$ such that a is a nearest neighbor of b with respect to W.

Remark. The $\Re^k$ spaces under L_p norms have nearest neighbor bounds. A construction using a fan of narrow cones about a can be used to prove this, or Lemma 7 below can be applied.

For the purpose of analyzing the algorithms given here, it would be ideal if a point a was γ-nearest neighbor to a constant number of points of a given set. Unfortunately, this may not be true, even for Euclidean spaces. However, a weaker condition does hold for spaces having a sphere-packing bound, and we'll use that condition to prove algorithm bounds. This is, roughly, that if a point a is γ-nearest neighbor to many points, then those points must be at a wide range of distances from a.

Definition: $\upsilon(x, R)$. For $x \in V$ and $R \subset V$, let $\upsilon(x, R)$ denote

$$\frac{\max\{d(x,y) \mid y \in R\}}{d(x,R).}$$

Definition: Neighbor sets. For $x \in V$, and $W \subset V$, let $N_\gamma(x, W)$ denote the points of W for which x is a γ-nearest neighbor with respect to W. Let $n_\gamma(x, W) \equiv |N_\gamma(x, W)|$. For $\gamma \geq 1$ and given $\upsilon > 0$, let

$$\mathcal{N}_{\gamma,\upsilon} \equiv \max\{n_\gamma(x, W) \mid x \in V, W \subset V, \upsilon(x, W) \leq \upsilon\},$$

if such a value exists.

Note that $\mathcal{N}_{1,\upsilon} \leq \mathcal{N}_1$ for any υ.

Definition: γ-nearest neighbor bounds. Say that a metric space (V, d) has a γ*-nearest neighbor* bound if it has a nearest neighbor bound, and also $\mathcal{N}_{\gamma,\upsilon}$ exists for all $\upsilon > 0$.

Such a bound is implied by a sphere-packing bound.

Lemma 7 *If a metric space (V, d) has a sphere-packing bound, then it also has a γ-nearest neighbor bound, with $\mathcal{N}_{\gamma,\upsilon} \leq \mathcal{S}_{2\gamma} \lg \upsilon$.*

Proof. Given $x \in V$ and $W \subset V$, for $i = 0 \ldots \lceil \upsilon(x, W) \rceil$, let

$$W^i \equiv \{y \in W \mid 2^i d(x, W) \leq d(x, y) \leq 2^{i+1} d(x, W)\}.$$

Now using Theorem 5, at most $\mathcal{S}_{2\gamma}$ points of W^i have x as a γ-nearest neighbor, and so at most $\mathcal{S}_{2\gamma} \lg \upsilon(x, W)$ points of W have x as nearest neighbor. $\square$

The following technical lemma will be useful.

Lemma 8 *For $x, y \in V$ and $R \subset V$,*

$$n_\gamma(x, R \cup \{y\}) \leq \mathcal{N}_{\gamma, \upsilon(x,R)} + 1.$$

That is, we can cheat in the quantity $\upsilon(,)$, and not pay too much.

PROOF. Trivial. □

3.5 Query time analysis

We can now analyze the running time of the search procedure for $M(S, Q)$.

First, a definition.

Definition: the distance ratio Υ. Given finite $W \subset V$, let $\Upsilon(W)$ be the *distance ratio*

$$\frac{\max\{d(x,y) \mid x, y \in W\}}{\min\{d(x,y) \mid x, y \in W, x \neq y\}}.$$

Of course, $\upsilon(x, W) \leq \Upsilon(W)$ for any $x \in W$. All our bounds could use $\max\{\upsilon(x, W) \mid x \in W\}$ instead of $\Upsilon(W)$, but the latter has a simpler and more familiar definition.

Theorem 9 *Suppose metric space (V, d) has a γ-nearest neighbor bound. Suppose Q, S, and $\{q\}$ are all random subsets of $Q \cup S \cup \{q\}$. Suppose $n = |S|$ and $Kn \leq m = |Q| = n^{O(1)}$. Then the expected work in answering a query for point q using $M(S, Q)$, built with parameter γ, given that the returned answer is correct, is*

$$O(\mathcal{N}_{\gamma,\Upsilon} \mathcal{N}_1 K \ln n),$$

where $\Upsilon \equiv \Upsilon(S \cup Q)$.

PROOF. The work done in answering a query is proportional to the number of members of A_j lists that are considered in the search. When adding site p_{i+1} to make R_{i+1}, an entry is made for p_{i+1} that will result in work for a query point q when there is $y \in R_i$ and $q' \in Q_{Ki}$ such that the following hold with respect to R_i:

1. y is nearest to q;
2. y is nearest to q';
3. p_{i+1} is γ-nearest to q'.

That is, in the "arrow" notation,

$$q \xrightarrow{1} y \xleftarrow{1} q' \xrightarrow{\gamma} p_{i+1}.$$

We want to bound the expected number of such configurations. First observe that this number is the sum, over $q'' \in Q_{Ki}$, of the expected number of such configurations with $q' = q''$. That is, the desired number is $|Q_{Ki}| = Ki$ times the expected number of such configurations, for a random choice of q' from Q_{Ki}.

The exchangeability assumptions for S, Q, and $\{q\}$ allow us to pick Q_{Ki}, R_i, and q in a few steps: choose a random subset $Q' \equiv Q_{Ki} \cup R_i \cup \{q\} \subset_R Q \cup S \cup \{q\}$, then choose $R'' \subset_R Q'$ of size $i+2$, then pick $q' \in_R R'$, and finally $q \in_R R' \setminus \{q'\}$. The set Q_{Ki} is then $Q' \cup \{q'\} \setminus R'$.

For a given p_{i+1}, the number of $q'' \in R'$ with p_{i+1} as γ-nearest with respect to R' is $\mathcal{N}_{\gamma, \upsilon(p_{i+1}, R')}$, by Lemma 7, or $1 + \mathcal{N}_{\gamma, \upsilon(p_{i+1}, R' \setminus \{q\})}$, by Lemma 8, and so the probability that a random $q' \in R'$ has p_{i+1} γ-nearest is $(1 + \mathcal{N}_{\gamma, \upsilon(p_{i+1}, R' \setminus \{q\})})/(i+2)$. Note that $\upsilon(p_{i+1}, R' \setminus \{q\} \leq \Upsilon(S \cup Q)$, so the probability is at most $(1 + \mathcal{N}_{\gamma, \Upsilon})/(i+2)$.

For any given such q', with $y \in R_i$ nearest to q' in R_i, the number of $b \in R' \setminus \{q'\}$ that have y as nearest neighbor is at most $\mathcal{N}_1$, since (V, d) has a nearest neighbor bound. The probability that a random $q \in R' \setminus \{q'\}$ has y nearest is $\mathcal{N}_1/(i+1)$.

The probability that (1), (2), and (3) hold for random q and q' is thus $\mathcal{N}_{\gamma, \Upsilon} \mathcal{N}_1/(i+2)(i+1)$, and so the expected number of configurations is Ki times this quantity, or $O(K/i)$. This bounds the work for random query q, associated with p_{i+1} on the list A_j list for site $y \in R_i$.

Summing this bound for $i+1$ from 1 to n yields the result.

□

3.6 Space bounds for $M(S, Q)$

The following lemma is needed for proving space bounds, and for proving bounds on the preprocessing time.

Lemma 10 *Suppose (V, d) is a metric space that has a sphere-packing bound, and $|V| = n$, with distance ratio $\Upsilon \equiv \Upsilon(V)$. For R a random subset of V of size i, $p \in V \setminus R$, and $\beta > 1$, the expected number of $q \in V \setminus R$ with p as β-nearest neighbor in R is*

$$O(\mathcal{N}_{\beta, \Upsilon}) n/i = O(\mathcal{S}_{2\beta} \log \Upsilon) n/i.$$

PROOF. The proof is similar to that of Theorem 9.

We observe that the desired quantity is $n - i$ times the probability that a random choice $q \in_R V \setminus R$ has $q \xrightarrow{\beta} p$.

The random choice of $R \subset_R V$ and $q \in_R V \setminus R$ is equivalent to picking random $R' \subset_R V$ of size $i+1$, then picking $q \in_R R'$, and finally setting $R \equiv R' \setminus \{q\}$.

By Lemma 7, the number of $x \in R'$ with p as β-nearest with respect to R' is at most $\mathcal{N}_{\beta, \Upsilon}$. The probability that $q \in_R R'$ is one such point is at most $\mathcal{N}_{\beta, \Upsilon}/(i+1)$. The result follows, multiplying by $n - i$.

□

The following lemma will be helpful later.

Lemma 11 *For random $q \in V$, and random $R \subset_R V$, the expected number of β-nearest neighbors of q with respect to R is*

$$O(\mathcal{N}_{\beta,\Upsilon}) = O(\mathcal{S}_{2\beta}) \log \Upsilon.$$

PROOF. From Lemma 10, for each $p \in R$ the number of $q \in S$ with p as a β-nearest neighbor is $O(\mathcal{N}_{\beta,\Upsilon} n/i)$, where $i = |R|$. Multiplying by i, which is the number of such p, and dividing by n, the number of such q, gives the result. The last equality in the lemma statement is from Lemma 7. □

Theorem 12 *Suppose Q and S are random subsets of $Q \cup S$. When (V, d) has a sphere-packing bound, the expected space used by the data structure $M(S, Q)$ is $O(\mathcal{S}_{2\gamma} \log \Upsilon(S \cup Q))Kn$.*

PROOF. When p_{i+1} is added, it is added to the A_j lists of those p_j for which there is some $q \in Q_{i+1}$ that has p_j as nearest neighbor in R_i and p_{i+1} as γ-nearest neighbor in R_{i+1}. Each such $q \in Q_{i+1}$ yields at most one entry in some A_j list; the expected number of such entries is bounded by applying the previous lemma to R_i as a random subset of $V = R_i \cup Q_{i+1}$. Summing over $i = 1 \ldots n$ yields the result. □

4 Faster preprocessing for $M(S, Q)$

This section gives a preprocessing algorithm for $M(S, Q)$. The algorithm requires $O(Kn)(\log n)^2(\log \Upsilon(S \cup Q)^2)$ expected time when S is a random subset of $S \cup Q$.

The problem in the basic construction of $M(S, Q)$ is to maintain, for each member of Q, its γ-nearest neighbors in R_i. As we'll see, it will be helpful to maintain, in addition, the γ-nearest neighbors of S in R_i: these will help speed up the computation of nearest neighbors when a site is added to R_i. (Of course, some members of S will be in R_i; for $p \in R_i$, as before, we'll consider its nearest neighbor to be the closest site in $R_i \setminus \{p\}$.) Suppose Q and S are random subsets of $Q \cup S$. Then $R_i \subset_R Q \cup S$, and so here the basic problem is to maintain the γ-nearest neighbors of a set with respect to a random subset of that set. Thus under this assumption, it's no loss of generality to solve this maintenance problem for a set S.

The general step of this construction is to update the γ-nearest neighbor relation for R_{i+1}, given that random $p_{i+1} \in S \setminus R_i$ has been added to random $R_i \subset S$, yielding R_{i+1}. To save i's, we'll refer in this section to random $R \subset S$ and random $p \in S \setminus R$. We'll assume that the metric space (V, d) has a sphere-packing bound.

The algorithm will maintain a neighborhood relation $\overset{\gamma'}{\Longrightarrow}$ between points in R. This relation will be used to find the points of S for which a point p is γ-nearest neighbor. This information, in turn, will help update the $\overset{\gamma'}{\Longrightarrow}$ relation.

4.1 Outline

The first subsection gives some probabilistic lemmas that are used in the analysis, and may help motivate the algorithm that follows. Next follows some geometric notation, and the basic relations that are maintained by the algorithm. The algorithm is given in two main parts, M.1 and M.2, with two auxiliary algorithms then described; finally, the analysis of the running time is completed.

Remember that n is the number of elements of S and $\Upsilon(S)$ is the ratio

$$\frac{\max\{d(x,y) \mid x,y \in S\}}{\min\{d(x,y) \mid x,y \in S, x \neq y\}.}$$

Hereafter, we'll refer to $\Upsilon(S)$ simply as Υ.

4.2 Probabilistic lemmas

In these lemmas, fix $\beta > 1$.

Definition: A_β**.** Let A_β denote $\mathcal{N}_{\beta,\Upsilon} n/i$.

Thus Lemma 10 can be restated as follows:

Lemma 13 *The expected number of $s \in S$ with p as a β-nearest neighbor in R is $O(A_\beta)$.*

Lemma 14 *Let $R \subset_R S$ of size i and $p \in_R S$. The expected number of configurations (p,q,p') with $q \in S$, $p' \in R$, and*

$$p \overset{<1}{\longleftarrow} q \overset{\beta}{\longrightarrow} p'$$

is

$$O(A_{2\beta}) = O(\mathcal{S}_{4\beta}(\log \Upsilon))n/i.$$

That is, when p is added, we can bound the expected number of β-nearest neighbors of points $q \in S$ that have p as new nearest neighbor.

Proof. The number of such configurations is the sum, over $p' \in R$, of the expected number involving p', or i times a bound that holds for any given p'. As in the proof of Theorem 9, the expected number of configurations is n times the expected number involving a random $q \in S$.

Thus we consider the expected number of configurations for some fixed p', and for random $q \in S$ and $p \in R$. We assume that $q \notin R$; the argument when $q \in R$ is similar.

Under these conditions, q and p are random elements of $R' \equiv R \cup \{q,p\}$, which we can view as chosen by first picking q from R', and then picking p from $R' \setminus \{q\}$. Let $n_2(q)$ denote the *second* nearest neighbor of q in R'. The problem becomes: what is the probability that random $q \in R'$ has $d(q,p') \leq \beta d(q,n_2(q))$, and that the nearest neighbor of q is picked to be p? The latter probability is

$1/(i+1)$; the former probability is $1/(i+2)$ times the number of points $q' \in R'$ that have $d(q',p') \le \beta d(q', n_2(q'))$.

Putting these considerations together, the expected number of configurations of the lemma statement is $in/(i+1)/(i+1)$ times the size of

$$\{q' \in R' \mid d(q',p') \le \beta d(q', n_2(q'))\}.$$

The number of such q' can be bounded by $\mathcal{S}_{4\beta} \lg \Upsilon$, with a proof similar to that of Lemma 7: separate the sites of R into groups according to their distance from p', and apply Theorem 5, so that at most $\mathcal{S}_{4\beta}$ sites in a group have a nearest neighbor in the group at a distance larger than $\beta/2$ times their distance to p'. Since V is a metric space, all but $\mathcal{S}_{4\beta}$ sites in the group have second nearest neighbors within a distance at most β times their distance to p'. The lemma follows.

□

4.3 Geometric notation and lemmas

Before stating the geometric starting point for the algorithm, some notation will be helpful:

Definition: $d(c)$**.** In the remainder of this section, abbreviate $d(c, R)$ by $d(c)$.

Definition: M_a**.** For $a \in R$, let $M_a \equiv \max\{d(s,a) \mid s \xrightarrow{1} a\}$.

Definition: $a \overset{\beta}{\Longrightarrow} b$**.** For $a, b \in R$, let $a \overset{\beta}{\Longrightarrow} b$ denote the condition that $d(a,b) \le \beta M_a$.

All these definitions are with respect to R, which is left implicit. When considering a site $p \notin R$, the sites $s \in S$ with $s \xrightarrow{<1} p$ are those which have p as nearest neighbor in $R \cup \{p\}$.

For $a, b \in R$, the relation $a \overset{2}{\Longrightarrow} b$ is akin to a and b being Delaunay neighbors: if there is some $s \in S$ with $a \xleftarrow{1} s \xrightarrow{1} b$, then a and b are Delaunay neighbors and both $a \overset{2}{\Longrightarrow} b$ and $b \overset{2}{\Longrightarrow} a$.

Note that if $a \overset{\beta}{\Longrightarrow} b$, then the site s realizing M_a has $a \xleftarrow{1} s \xrightarrow{\beta+1} b$, since by the triangle inequality, $d(s,b) \le d(s,a) + d(a,b) \le (1+\beta)d(s)$.

The following lemmas will be helpful.

Lemma 15 *If $x \xleftarrow{\beta} x' \xrightarrow{\delta} y'$, then $d(x,y') \le (\beta+\delta)d(x')$.*

Proof. Using the triangle inequality,

$$d(x,y') \le d(x,x') + d(x',y') \le \beta d(x') + \delta d(x') = (\beta+\delta)d(x').$$

□

Lemma 16 *If $x' \xrightarrow{\beta} y \xrightarrow{\delta} z$ for $z \in R$, then $x' \xrightarrow{\beta+\delta+\beta\delta} z$.*

PROOF. Let $a \in R$ have $a \xleftarrow{1} x'$. From the previous lemma,

$$d(y) \leq d(a, y) \leq (1+\beta)d(x'),$$

and so

$$d(x', z) \leq d(x', y) + d(y, z) \leq \beta d(x') + \delta(1+\beta)d(x'),$$

and the result follows. $\square$

Lemma 17 *The number of $a \in R$ with $a \stackrel{\beta}{\Longrightarrow} p$ is no more than the expected number of $s \in S$ with $s \xrightarrow{\beta+1} p$, which is $O(A_{\beta+1})$.*

PROOF. For each $a \in R$ with $a \stackrel{\beta}{\Longrightarrow} p$, as noted there is some s with $s \xrightarrow{\beta+1} p$. Since $s \xrightarrow{1} a$ for only one $a \in R$, the lemma follows. $\square$

4.4 Algorithm Structure

Lemma 16 may help motivate the following definition.

Definition: γ'. Fix a parameter α with $1 \leq \alpha \leq \gamma$, and let $\gamma' \equiv 1+\alpha+\gamma+\alpha\gamma$.

The algorithm will maintain the nearest neighbor relation $s \xrightarrow{1} f$ for each $f \in R$, and a subset of the relation $a \stackrel{\gamma'}{\Longrightarrow} f$ for each $a, f \in R$. That is, $a \stackrel{\gamma'}{\Longrightarrow} f$ will be known for all $a, f \in R$ for which there are $s, c \in S$ with

$$a \xleftarrow{1} s \xrightarrow{\gamma} c \xrightarrow{\alpha} f.$$

This condition implies $a \stackrel{\gamma'}{\Longrightarrow} f$, by Lemmas 16 and 15.

The goal is to update these relations when p is added to R, in roughly $O(n/i)$ time, where $i = |R|$.

Algorithm M.1, given in §4.6, finds s with $s \xrightarrow{<1} p$ for a given p, using the $\stackrel{\gamma'}{\Longrightarrow}$ relation. Algorithm M.2, given in §4.7, updates the $\stackrel{\gamma'}{\Longrightarrow}$ relation, and uses a set $\hat{G} \subset R$ whose construction is given in §4.9. The maintenance of the data structures representing $\xrightarrow{1}$ and $\stackrel{\gamma'}{\Longrightarrow}$ is described in §4.8.

4.5 Heaps of relations

In large part, the algorithm is simply the maintenance of data structures representing the $\longrightarrow$ and $\Longrightarrow$ relations. These will comprise, for each $f \in R$, some sets of sites of S, each in a heap (priority queue) with maximum key value on top. The heaps are:

$H_{f\leftarrow}$: $\{s \in S \mid s \xrightarrow{1} f\}$, with key values $d(f, s)$;

$H_{f\Rightarrow}$: $\{a \in R \mid f \stackrel{\gamma'}{\Longrightarrow} a\}$, with key values $-d(a, f)$;

$H_{f\Leftarrow}$: $\{a \in R \mid a \stackrel{\gamma'}{\Longrightarrow} f\}$, with key values M_a;

The heap $H_{f\Rightarrow}$ stores the a with minimum $d(a,f)$ on top of the heap.

Heaps are useful here because a simple procedure allows entries with key values greater than a given value X to be found in constant time for each such entry. The procedure simply checks if the top of the heap has key value greater than X; if so, that value is reported, and recursively the children of the top are checked.

Note that the keys of entries in $H_{f\Leftarrow}$ are the keys of the tops of heaps $H_{a\leftarrow}$, when $a \stackrel{\gamma'}{\Longrightarrow} f$. (A fine point here is that the key for a in $H_{f\Leftarrow}$ is the value of M_a *when a is inserted in $H_{f\Leftarrow}$*; that is, the value of M_a may be allowed to change without updating the key for a immediately. This issue is discussed in §4.8.)

4.6 Finding γ neighbors using $\stackrel{\gamma'}{\Longrightarrow}$

We can now discuss how to obtain all $s \in S$ with $s \xrightarrow{\gamma} p$, and in particular, those with $s \xrightarrow{\alpha} p$ and even $s \xrightarrow{<1} p$.

Algorithm M.1: find s with $s \xrightarrow{\gamma} p$. Look up f, the nearest neighbor of p in R. For each entry a in $H_{f\Leftarrow}$, use $H_{a\leftarrow}$ to find all s with

$$s \xrightarrow{1} a \text{ and } d(s) \geq d(a,p)/(1+\gamma).$$

Check if $d(s,p) \leq \gamma d(s)$, so that $s \xrightarrow{\gamma} p$.

Lemma 18 *Algorithm M.1 finds the sites with p as γ-nearest neighbor in R.*

Proof. Suppose that

$$a \xleftarrow{1} s \xrightarrow{\gamma} p \xrightarrow{1} f \tag{1}$$

holds. The relation in the middle, $s \xrightarrow{\gamma} p$, must be found using stored information. By Lemma 16, the above implies $s \xrightarrow{2\gamma+1} f$, and so $d(a,f) \leq (2+2\gamma)d(s)$ using Lemma 15. Hence $d(a,f) \leq \gamma' d(s) \leq \gamma' M_a$, and so $a \stackrel{\gamma'}{\Longrightarrow} f$. Thus, $a \stackrel{\gamma'}{\Longrightarrow} f$ is necessary for $s \xrightarrow{\gamma} p$. Finally, $a \xleftarrow{1} s \xrightarrow{\gamma} p$ implies that $d(a,p) \leq (1+\gamma)d(s)$, using Lemma 15. $\square$

Lemma 19 *The expected work by Algorithm M.1 is $O(A_{\gamma'+1})$.*

Proof. The number of sites $a \in R$ inspected by the algorithm is bounded using Lemmas 17 and 13. The s examined all have

$$d(s,p) \leq d(s,a) + d(a,p) \leq (2+\gamma)d(s),$$

and so all s examined have $s \xrightarrow{\gamma+2} p$. The result follows. $\square$

4.7 Finding $\overset{\gamma'}{\Longrightarrow}$ relations of p

With knowledge of the sites for which p is a γ-nearest neighbor comes the knowledge of those sites for which p is a nearest neighbor. (Also the value of M_p is found.) These sites must have their nearest neighbor sites changed to p; however, it will be essential to retain knowledge of their former nearest neighbors when finding the sites $a \in R$ for which $p \overset{\gamma'}{\Longrightarrow} a$ or $a \overset{\gamma'}{\Longrightarrow} p$ after p is added. (Note that the exposition below retains the $\longrightarrow$ and $\Longrightarrow$ relations with respect to R, before p is added.)

Since the relation $g \overset{\gamma'}{\Longrightarrow} a$ depends on M_g, and M_g may change when p is added, the heaps for g must be updated for such changes. This issue is discussed in §4.8 below.

Algorithm M.2. Find all a with $p \overset{\gamma'}{\Longrightarrow} a$ or $a \overset{\gamma'}{\Longrightarrow} p$.
For each $g \in R$ and $c \in S$ with $g \xleftarrow{1} c \xrightarrow{<1} p$, use $H_{g\Rightarrow}$ to find all a with

$$g \overset{\gamma'}{\Longrightarrow} a \text{ and } d(a,g) \leq \gamma' d(c,g),$$

and check if $d(c,a) \leq (\gamma'-1)d(c,p)$. If so, record $p \overset{\gamma'}{\Longrightarrow} a$. (This may not record all a for which $p \overset{\gamma'}{\Longrightarrow} a$, but it does record all those needed for the correctness condition; heuristically, the fewer a recorded, the better.)

Make the set G where

$$G \equiv \{g \in R \mid g \xleftarrow{1} c \xrightarrow{\alpha} p, c \in S\}.$$

Using the algorithm of §4.9, build a set $\hat{G} \subset G$, of size independent of n, with the property that for all c with $c \xrightarrow{\alpha} p$ there is $\hat{g} \in \hat{G}$ with $c \xrightarrow{\alpha} \hat{g}$. Now for each $\hat{g} \in \hat{G}$, use $H_{\hat{g}\Leftarrow}$ to find all a with

$$a \overset{\gamma'}{\Longrightarrow} \hat{g} \text{ and } M_a \geq d(p,\hat{g})/2\alpha(1+\gamma),$$

and check if $a \overset{\gamma'}{\Longrightarrow} p$.

(The test for $a \overset{\gamma'}{\Longrightarrow} p$ could use the stringent condition that there is some s with $a \xleftarrow{1} s \xrightarrow{\gamma'-1} p$, and still be correct; moreover, all $s \xrightarrow{1} a$ with $d(a,s) \geq d(a,p)/\gamma'$ could be tested for $s \xrightarrow{\gamma'-1} p$ within the desired time bounds. This might give a heuristic improvement.)

This algorithm updates the subset of $\overset{\gamma'}{\Longrightarrow}$ that is promised to be maintained.

Lemma 20 Correctness of Algorithm M.2.
The above algorithm finds all a such that there are s and c with

$$a \xleftarrow{\alpha} s \xleftarrow{\gamma} c \xrightarrow{<1} p,$$

and records $p \stackrel{\gamma'}{\Longrightarrow} a$, *or*

$$a \stackrel{1}{\longleftarrow} s \stackrel{\gamma}{\longrightarrow} c \stackrel{\alpha}{\longrightarrow} p,$$

and records $a \stackrel{\gamma'}{\Longrightarrow} p$. *This maintains the condition that for all* $a, g \in R$, *if there are* $s, c \in S$ *with*

$$a \stackrel{1}{\longleftarrow} s \stackrel{\gamma}{\longrightarrow} c \stackrel{\alpha}{\longrightarrow} g,$$

then $a \stackrel{\gamma'}{\Longrightarrow} g$ *is known.*

PROOF. There are two cases.

Case I: $a \stackrel{\alpha}{\longleftarrow} s \stackrel{\gamma}{\longleftarrow} c \stackrel{<1}{\longrightarrow} p$. Here the site $g \in R$ with $c \stackrel{1}{\longrightarrow} g$ has $g \stackrel{\gamma'}{\Longrightarrow} a$ known, by induction. Moreover, $d(c,a) \leq (\gamma' - 1)d(c,g)$ by Lemma 16, and so $d(g,a) \leq \gamma' d(c,g)$ by Lemma 15. Hence the conditions under which a is examined for $p \stackrel{\gamma'}{\Longrightarrow} a$ are necessary for that condition to hold. Also, if $d(c,a) \leq (\gamma' - 1)d(c,p)$, then $p \stackrel{\gamma'}{\Longrightarrow} a$ using Lemma 15 and $M_p \geq d(c,p)$.

Case II: $a \stackrel{1}{\longleftarrow} s \stackrel{\gamma}{\longrightarrow} c \stackrel{\alpha}{\longrightarrow} p$. By induction, any $g \in R$ with $c \stackrel{\alpha}{\longrightarrow} g$ has $a \stackrel{\gamma'}{\Longrightarrow} g$ recorded. By construction, there is some $\hat{g} \in \hat{G}$ examined with $c \stackrel{\alpha}{\longrightarrow} \hat{g}$; also, using Lemma 17,

$$d(c) \leq d(c,a) \leq (1+\gamma)d(s),$$

and using the triangle inequality,

$$d(\hat{g},p) \leq 2\alpha d(c) \leq 2\alpha(1+\gamma)d(s) \leq 2\alpha(1+\gamma)M_a.$$

Hence the conditions under which a is examined for $a \stackrel{\gamma'}{\Longrightarrow} p$ are necessary. □

Lemma 21 *Algorithm M.2 requires* $O(A_{2(\gamma'+1)})$ *expected time to find all* $a \in R$ *with* $p \stackrel{\gamma'}{\Longrightarrow} a$, *and* $O(A_{\gamma''}|\hat{G}|)$ *to find all* $a \in R$ *with* $a \stackrel{\gamma'}{\Longrightarrow} p$, *where* $\gamma'' \equiv 2 + \gamma + 3\alpha(1+\gamma)$.

PROOF. The points $a \in R$ examined for $p \stackrel{\gamma'}{\Longrightarrow} a$ have

$$p \stackrel{<1}{\longleftarrow} c \stackrel{1}{\longrightarrow} g \stackrel{\gamma'}{\Longrightarrow} a$$

for some c and g, and have $d(a,g) \leq \gamma' d(c)$. By the triangle inequality, $d(c,a) \leq (\gamma'+1)d(c)$, and so the first claim follows from Lemma 14.

Since $a \stackrel{\gamma'}{\Longrightarrow} \hat{g}$, $d(a,\hat{g}) \leq \gamma' M_a$, and since $d(p,\hat{g}) \leq 2\alpha(1+\gamma)M_a$, it follows that for s realizing M_a,

$$d(s,p) \leq d(s,a) + d(a,\hat{g}) + d(\hat{g},p) \leq (1 + \gamma' + 2\alpha(1+\gamma))d(s,a),$$

and the second claim follows from simplifying this expression, and applying Lemmas 17 and 13. □

4.8 Maintaining heaps

As discussed in §4.5, the relations $a \xleftarrow{1} s$ and $a \overset{\gamma'}{\Longrightarrow} b$ are represented using heaps for $a \in R$. When p is added, the heaps H_{p*} are created and some heaps H_{a*} are updated to reflect relations involving p. First, a bound on the cost of this operation.

Lemma 22 *The expected cost of adding relations $s \xrightarrow{1} p$, $a \overset{\gamma'}{\Longrightarrow} p$, or $p \overset{\gamma'}{\Longrightarrow} a$, is $A_{2(\gamma'+1)}O(\log n)$.*

Proof. Each relation $p \overset{\gamma'}{\Longrightarrow} a$ implies that for $s \xrightarrow{<1} p$ realizing M_p,

$$p \xleftarrow{<1} s \xrightarrow{\gamma'+1} a.$$

The bound follows by Lemma 14, and using a heap with $O(\log n)$ per insertion.

From Lemmas 17 and 13, and using a heap with insertion time $O(\log n)$, the time for adding the first two types of relations is $A_{\gamma'+1}O(\log n)$. $\square$

The relations $\overset{\gamma'}{\Longrightarrow}$ continue to hold when p is added, except possibly for $g \in R$ with $g \xleftarrow{1} c \xrightarrow{<1} p$ for some $c \in S$. We next look at what heap maintenance must be done to reflect this.

The entry for c in $H_{g\leftarrow}$ can be deleted in $O(\log n)$ time.

The heaps $H_{g\Rightarrow}$ and $H_{a\Leftarrow}$ have entries a with $g \overset{\gamma'}{\Longrightarrow} a$, and this relation may become false when p is added. When $H_{g\Rightarrow}$ is used in Algorithm M.2, the a examined are close to g, and the analysis of Lemma 21 holds even when $g \overset{\gamma'}{\Longrightarrow} a$ is false. Hence, no updating of $H_{g\Rightarrow}$ need be done.

Finally, a change in $H_{a\Leftarrow}$ in Algorithm M.2 must be considered. The approach we take is to do nothing to update $H_{a\Leftarrow}$, but instead to verify relations when they are used. That is, we verify the relation $a \overset{\gamma'}{\Longrightarrow} f$ claimed by $H_{f\Leftarrow}$ in Algorithm M.1, and the relation $a \overset{\gamma'}{\Longrightarrow} \hat{g}$ claimed by $H_{\hat{g}\Leftarrow}$ in Algorithm M.2. If the key for a in $H_{f\Leftarrow}$ is still M_a, then no change in $H_{f\Leftarrow}$ is needed. If M_a is currently smaller than that key value, but still $M_a \geq d(a, \hat{g})/\gamma'$, then the relation still holds, but the key value must be updated and its heap position changed. If $M_a < d(a, \hat{g})/\gamma'$, then a is deleted from $H_{\hat{g}\Leftarrow}$. Similar maintenance can be done for $H_{f\Leftarrow}$ in Algorithm M.1.

Lemma 23 *Verifying and updating the relations in heaps $H_{f\Leftarrow}$ and $H_{\hat{g}\Leftarrow}$ needs $O(A_{\gamma''}|\hat{G}|\log n)$ expected time. Here γ'' is defined as in Lemma 21.*

This ignores the work in deleting entries a for which $a \overset{\gamma'}{\Longrightarrow} \hat{g}$ is no longer true. However, this work is done once for each such relation, and therefore can be charged to the insertion time for a in $H_{\hat{g}\Leftarrow}$.

Proof. This is simply the $O(\log n)$ work for each a examined by some $H_{\hat{g}\Leftarrow}$. The time to verify $H_{f\Leftarrow}$ is dominated by this. $\square$

4.9 Finding $\hat{G}$

Next we'll see that if a sphere-packing bound holds, as discussed in §3.4, then the set $\hat{G}$ exists, and can be found efficiently. The algorithm is described within the proof of the following lemma.

Lemma 24 *Suppose (V, d) has a sphere-packing bound. For $p \in S$ and $R \subset S$, let S' denote the set of $c \in S$ with $c \xrightarrow{\alpha} p$ with respect to R, where $\alpha > 4/3$. Let*

$$G \equiv \{g \in R \mid c \xrightarrow{1} g, c \in S'\}.$$

Then there is a set $\hat{G} \subset G$ of size $O(\mathcal{S}_{3\alpha} \log_{3\alpha/4} \upsilon(p, S'))$ such that for any $c \in S'$, there is some $\hat{g} \in \hat{G}$ with $c \xrightarrow{\alpha} \hat{g}$.

Proof. Let

$$d_l \equiv \min\{d(c) \mid c \in S'\},$$

and let

$$d_h \equiv \max\{d(c) \mid c \in S'\}.$$

Let

$$\hat{\alpha} \equiv 3\alpha/4.$$

Divide the interval (d_l, d_h) into ranges

$$(\hat{\alpha}^k d_l, \hat{\alpha}^{k+1} d_l), \text{ for } k = 0 \ldots \lceil \log_{3\alpha/4} \upsilon(p, S') \rceil.$$

For each such k, consider the set S_k of $c \in S'$ such that $d(c) \in (\hat{\alpha}^k d_l, \hat{\alpha}^{k+1} d_l)$. Construct a packing $S'_k \subset S_k$ using Theorem 5, such that $d(c, S'_k) \leq \hat{\alpha}^{k+1} d_l/3$ for all $c \in S_k$. Since $d(c, p) \leq \alpha d(c) \leq \alpha \hat{\alpha}^{k+1} d_l$, the size of S'_k need be at most $\mathcal{S}_{3\alpha}$.

Now for each $c \in S_k$, there is some $c' \in S'_k$ with

$$d(c, c') \leq \hat{\alpha}^{k+1} d_l/3 \leq \alpha d(c)/4,$$

and so if $c' \xrightarrow{1} g'$, then

$$d(c, g') \leq d(c, c') + d(c', g') \leq \alpha d(c)/4 + \hat{\alpha}^{k+1} d_l \leq \alpha d(c)/4 + 3\alpha d(c)/4 = \alpha d(c).$$

Then

$$\hat{G} \equiv \cup_k \{g' \in R \mid c' \xrightarrow{1} g', c' \in S'_k\}$$

satisfies the conditions of the lemma. □

Lemma 25 *The algorithm for finding $\hat{G}$ needs*

$$O(|G||\hat{G}|) = O(A_\alpha |\hat{G}|)$$

expected time.

Proof. This is clear from the algorithm, and the bound on $|G|$ is from the definition of A_α. □

4.10 Concluding Analysis

Before considering the time complexity, we should note correctness.

Theorem 26 *Algorithms M.1 and M.2 find the points for which a site p is γ-nearest in R.*

PROOF. This follows from Lemmas 18 and 20. $\square$

Theorem 27 *The all-nearest-neighbors problem can be solved for a metric space with a sphere-packing bound in*

$$O(n)(\log n)^2(\log \Upsilon)^2$$

expected time, for parameters α and γ satisfying $4/3 < \alpha \leq \gamma$.

PROOF. The bounds given by Lemmas 19, 21, 22, 23, and 25 are dominated by $O(A_{\max\{\gamma'',2(\gamma'+1)\}}|\hat{G}|\log n)$.

Using the definition of A_β, we obtain an expected bound on the order of

$$\frac{n}{i}\left(\mathcal{N}_{\max\{\gamma'',2(\gamma'+1)\},\Upsilon}|\hat{G}|\log n\right)$$

for the cost of adding p to R of size i. The size of $\hat{G}$ is $O(\mathcal{S}_{3\alpha}(\log_{3\alpha/4}\Upsilon))$ from Lemma 24. Using this bound, the hypothesis that $\alpha > 4/3$ is fixed, and summing over i up to n gives a total expected cost

$$O(n\log^2 n)\mathcal{N}_{\max\{\gamma'',2(\gamma'+1)\},\Upsilon}\mathcal{S}_{3\alpha}\log\Upsilon.$$

Finally, from Lemma 7, the total expected cost is

$$O(n\log^2 n)(\mathcal{S}_{\max\{\gamma'',2(\gamma'+1)\}}\mathcal{S}_{3\alpha}\log^2\Upsilon),$$

or, neglecting constants, $O(n(\log n)^2(\log\Upsilon)^2)$. $\square$

Theorem 28 *If Q and S are random subsets of $Q \cup S$, and V has a sphere-packing bound, then $M(S,Q)$ can be built in*

$$O(Kn)(\log n)^2(\log\Upsilon(S\cup Q))^2$$

expected time.

PROOF. Apply the algorithm of this section to $S \cup Q_{Kn}$, adding the elements of S first. The resulting γ-nearest neighbor information is exactly what is needed for the data structure. $\square$

5 Data Structure $D(S)$

This section will describe another approach to the post office problem, in this same setting, but with some differences: the algorithm always returns the correct answer, and does not require Q, but requires somewhat more space and query time. The algorithm can be analyzed using the exchangeability assumption, where a query point q is a random element of $\{q\} \cup S$.

The algorithm uses a divide-and-conquer scheme apparently somewhat like Brin's[Bri95].

In this section, the values $\mathcal{N}_{\gamma,\Upsilon(R)}$ will be considered for various subsets R of S. We'll use the uniform bound $\mathcal{N}_{\gamma,\Upsilon(S)}$, abbreviated just as $\mathcal{N}_\gamma$. Recall Lemma 7, that if (V, d) obeys a sphere-packing bound, then $\mathcal{N}_{\gamma,\Upsilon(S)} \leq \mathcal{S}_{2\gamma} \lg \Upsilon(S)$.

Preprocessing. Given a set of n sites $S \subset V$, the data structure $D(S)$ is built as follows: take a random subset R of S. Recursively build the data structure $D(R)$ for R, and then find, for each $p \in S \setminus R$, its nearest neighbor in R. Let S_a^1 denote the sites in $S \setminus R$ that have $a \in R$ as nearest neighbor. For each $a \in R$, build a list L_a of sites of R, in nondecreasing order of their distance to a. For each $c \in S_a^1$, find the 3-nearest neighbors of c in R as follows: Walk down the list L_a, checking for each site $b \in R$ if b is 3-nearest to c in R. Stop this traversal of L_a when $d(a, b) > 4d(c, a)$. For $a \in R$, let S_a^3 denote the set of sites in S with a as 3-nearest neighbor. Recursively build $D(S_a^3)$.

This completes the preprocessing. The data structure $D(S)$ comprises $D(R)$, the lists L_a for $a \in R$, and the recursively built data structures $D(S_a^3)$ for $a \in R$.

For convenience of analysis, the construction will use random subsets of size $r_k \equiv n^{1/2^{k+1}}$ at recursion depth k, and terminate the recursion at $k = l_n \equiv \lceil \lg \lg n \rceil$. (The construction of $D(S)$ for the original input is at depth 0; when constructing $D(S)$ at depth k, the construction of the data structures $D(S_a^3)$ and $D(R)$ are at depth $k + 1$.) The recursion bottoms out by simply storing the sites in a list, at recursion depth l_n, or when the set has fewer than some constant number of members.

Answering Queries. To find the nearest site to a point q, use the data structure $D(R)$ to find the nearest neighbor a of q in R. Next find the 3-nearest neighbors of q in R as in the preprocessing. For each b that is 3-nearest neighbor to q in R, recursively search $D(S_b^3)$ for the closest site in S_b^3 to q. Return the closest site found by all such searches.

The recursion bottoms out by searching the list of sites in linear time.

There are a few correctness conditions to verify.

Theorem 29 *The query and preprocessing algorithms find all 3-nearest neighbors of q and of sites in $S \setminus R$. The query procedure returns the closest site in S to q.*

PROOF. Suppose b is a 3-nearest neighbor of a site p with nearest neighbor a

in R, or $a \xleftarrow{1} p \xrightarrow{3} b$ with respect to R. Then $d(a,b) \leq 4d(p,a)$ by Lemma 15. Thus $d(b,a) \leq 4d(p,a)$ is necessary for b to be a 3-nearest neighbor of p.

The query procedure is correct: if $c \in S$ and $b \in R$ have $q \xrightarrow{<1} c \xrightarrow{1} b$ with respect to R, then $q \xrightarrow{3} b$ by Lemma 16. In othe words, the nearest neighbor b to c is a 3-nearest neighbor of q, and the query procedure will find c when recursively searching S_b^3. $\square$

Rather than store L_a as a sorted list, as described above, it is enough to store the entries in a heap. With appropriate heap ordering relations, all members of L_a at a distance closer than a value X can be found in constant time per reported member, using a recursive traversal of the heap, just as for sorted lists. The heaps for sites $a \in R$ can be built in constant time per entry, however, unlike sorted lists; this might be a bit faster in practice, and simplifies the analysis.

Lemma 30 *The expected size of a set of sites considered at recursion depth k is*

$$O(\mathcal{N}_3)^k n^{1/2^k}.$$

PROOF. Suppose inductively that the expected size of the set W in the parent recursion, at depth $k-1$, is $m = O(\mathcal{N}_3)^{k-1} n^{1/2^{k-1}}$, as is plainly true for $k = 1$. One branch of recursion considers $R \subset W$ of size $n^{1/2^k}$, which satisfies the claim. Other branches of recursion apply to sets W_a^3, which have expected sizes $O(\mathcal{N}_3)m/r_{k-1}$ using Lemma 10. (Note that the expectation for W_a^3 is with respect to the random choice of R, while the expectation for $|W|$ is from random choices at lesser recursion depth; hence the expectations can be "mixed" in this fashion.) Since $r_{k-1} \equiv n^{1/2^k}$, the result follows. $\square$

Theorem 31 *Suppose metric space (V,d) has a γ-nearest neighbor bound, and $S \subset V$ is a set of n sites. Suppose q is a random element of $\{q\} \cup S$. Then the expected time to find the closest site in S to q using $D(S)$ is*

$$\mathcal{N}_5 (\lg n)^{O(1) + 2 \lg \mathcal{N}_3}$$

as $n \to \infty$.

PROOF. Consider the query procedure for a set T at depth k in the recursion. Let $Z(k)$ denote the expected time for the query procedure applied to a set at depth k, so that the expected time overall is $Z(0)$. Let R be the random subset of T used at that step. The set T is a random subset of S, subject to the 3-nearest neighbor conditions used in passing down the recursion. Since q is a random element of $\{q\} \cup S$, and subject to the same 3-nearest neighbor conditions (or else T would not be considered for q), q is a random element of $\{q\} \cup R$. Thus the expected time to search $D(R)$ is $Z(k+1)$. Using the triangle inequality, each site b examined in finding 3-nearest neighbors of q satisfies

$$d(b,q) \leq d(b,a) + d(a,q) \leq 5d(a,q).$$

From Lemma 11, the expected work in examining such b is $O(\mathcal{N}_5)$. The expected work done recursively can be expressed as the sum, over $b \in T$, of $Z(k+1)$, times the probability that b is in R and is a 3-nearest neighbor of q. The correctness of this claim requires the observation that q is a random element of $\{q\} \cup T_b^3$. The probability that b is a 3-nearest neighbor of q in R is $O(\mathcal{N}_3)/|T|$, using Lemma 11. The expected query time is therefore

$$Z(k) \leq Z(k+1) + O(\mathcal{N}_5) + O(\mathcal{N}_3)/|T| \sum_{a \in T} Z(k+1),$$

or

$$Z(k) \leq Z(k+1) + O(\mathcal{N}_5) + O(\mathcal{N}_3) Z(k+1),$$

for k smaller than the recursion depth l_n and $|T|$ large. From Lemma 30, the expected size of the sets T at the bottom level is $O(\mathcal{N}_3)^{l_n}$, and this bounds the expected cost at the bottom level of recursion. The overall expected cost is

$$O(\mathcal{N}_5) O(\mathcal{N}_3 + 1)^{l_n} O(\mathcal{N}_3)^{l_n},$$

which implies the expected time bound claimed. □

Theorem 32 *Suppose metric space (V, d) has a γ-nearest neighbor bound, and $S \subset V$ is a set of n sites. Then the expected time to build $D(S)$ is*

$$O(n) \mathcal{N}_5 (\lg n)^{O(1) + 2 \lg \mathcal{N}_3}$$

as $n \to \infty$.

PROOF. The general step of the algorithm builds a data structure for a set T at recursion depth k. Let $Y(k)$ denote the expected time needed to build the data structure for such a set T. The expected cost of building $D(R)$ is $Y(k+1)$; the expected cost of finding the nearest neighbors in R of each site can be bounded at $Z(k+1)$ per site; the expected cost of finding 3-nearest neighbors in R of each site is $O(r_k^2) + O(\mathcal{N}_5)|T|$, since each b examined in a list L_a for site $c \in T_a^1$ is 5-nearest neighbor to c; the number of such neighbors is bounded using Lemma 11, averaging over all $c \in T \setminus R$. Construction of the heaps L_a, for $a \in R$, requires $O(r_k^2) = O(n^{1/2^k})$ altogether. Finally, the work in building the data structures $D(T_b^3)$ requires time proportional to the sum, over $b \in T$, of the probability $r_k/|T|$ that $b \in R \subset T$, times the expected work $Y(k+1)$ for T_b^3. Thus the work $Y(k)$ satisfies

$$Y(k) \leq Y(k+1) + Z(k+1)|T| + O(\mathcal{N}_5)|T| + O(n^{1/2^k}) + r_k Y(k+1),$$

or using the bound $E|T| \leq O(\mathcal{N}_3)^k n^{1/2^k}$ of Lemma 30,

$$Y(k) \leq (Z(k+1) + O(\mathcal{N}_5)) O(\mathcal{N}_3)^k n^{1/2^k} + O(n^{1/2^k}) + (n^{1/2^{k+1}} + 1) Y(k+1),$$

with $Y(l_n) \leq O(\mathcal{N}_3)^{l_n}$, giving the result claimed. □

Theorem 33 *Suppose metric space (V, d) has a γ-nearest neighbor bound, and $S \subset V$ is a set of n sites. Then the expected space for $D(S)$ is*

$$O(n)(\lg n)^{O(1)+\lg \mathcal{N}_3}$$

as $n \to \infty$.

PROOF. The total space requires for all lists L_a at each depth k is $O(n)$; the total expected space required for sets at the bottom of the recursion is $nO(\mathcal{N}_3)^{l_n}$, with $l_n \equiv \lceil \lg \lg n \rceil$, and this gives the bound. $\square$

5.1 Remarks

The algorithm recursively builds data structures for sets S_b^3, the sites having b as 3-nearest neighbor in R, rather than for S_b^1, those having b as nearest neighbor. The query procedure would still be correct if the latter procedure were followed, and the space required would be $O(n)$, since the sites would be partitioned at each step. So far, it hasn't been clear how to analyze such an attractive algorithm.

It is a bit artificial to have a fixed schedule of sample sizes, $r_k \equiv n^{1/2^{k+1}}$; a more natural sample size would be simply to use the square root of the set size $|T|$. This seems to be more difficult to analyze, and still seems artificial: why not use a constant sample size, or $n/2$? However, a smaller sample size would have an even bigger blowup in cost, due to the imperfect nature of the divide-and-conquer scheme used. A larger sample size runs into the apparent need for $\Omega(r^2)$ space, to hold the lists L_a that are used to find 3-nearest neighbors. While the choice of r is not tuned to be best possible, it seems close to the only alternative with the ideas available.

In $\Re^k$ space with an L_p norm, the expected query time takes the form

$$(\lg n)^{O(k)+\lg \lg \Upsilon}.$$

5.2 γ-queries and inverse queries

The data structure can be extended in two ways that are sometimes useful. The first extension is to allow finding all γ-nearest neighbors of a query point, not just the nearest neighbors. The other extension is to allow *inverse* queries, for a point p and subset $R \subset S$, that return sites $z \in S \setminus R$ for which p is γ-nearest to z in R.

We will here only sketch the straightforward extension to build a data structure $D_\gamma(S)$ that allows γ-nearest neighbor queries; the change is basically to replace finding 3-neighbors by finding $(1+2\gamma)$-neighbors, to maintain $S_a^{1+2\gamma}$ instead of S_a^3, to search lists only when $d(a,b) \leq (2+2\gamma)d(b,q)$ instead of a multiplier of 4, and to replace $\mathcal{N}_5$ in the analysis by $\mathcal{N}_{3+2\gamma}$, and $\mathcal{N}_3$ by $\mathcal{N}_{1+2\gamma}$.

Theorem 34 *Suppose metric space (V, d) has a γ-nearest neighbor bound, and $S \subset V$ is a set of n sites. The expected time to build $D_\gamma(S)$ is*

$$O(n)A(\lg n)^{O(1)+2\lg B}$$

as $n \to \infty$, and the expected query time is

$$A(\lg n)^{O(1)+2\lg B}.$$

Here

$$A = \mathcal{N}_{3+2\gamma,\Upsilon(S)}$$

and

$$B = \mathcal{N}_{1+2\gamma,\Upsilon(S)},$$

which are $O(\log \Upsilon(S))$ as a function of S when (V, d) has a sphere-packing bound.

Given random $R \subset S$, it is possible to build a data structure for *inverse* neighbor queries: given point p, quickly find all $q \in S \setminus R$ such that p is a γ-nearest neighbor to q in $\{p\} \cup R$.

One approach is the following: build the data structure $D_{1+\gamma}(R)$, and use it to find the nearest neighbor, and all $(1+\gamma)$-nearest neighbors, in R of all $q \in S \setminus R$. For each $c \in R$, find the sets $S_c^{1+\gamma}$ of q that have c as $(1+\gamma)$-nearest neighbor in R, and the analogous set S_c^1. Store these sets in heaps, so that all q in them with $d(q, c)$ greater than some value can be found in constant time per Q.

With this preprocessing, for given p, find its $(1+\gamma)$-nearest neighbors in R, including its nearest neighbor a. Examine all $q \in S_a^{1+\gamma}$ that have $d(a, q) \geq d(a, p)/(1+\gamma)$, and check if they have p as γ-nearest. For each $b \in R$ that is $(1+\gamma)$-nearest to p in R, examine the set S_b^1 of q that have b as nearest in R, and check all such q that have $d(b, q) \geq d(a, p)$ and $d(b, q) \geq d(b, p)/(1+\gamma)$, to see if p is a $(1+\gamma)$-nearest neighbor.

This completes the query procedure.

Theorem 35 *The inverse neighbor procedure finds all q with p γ-nearest in $\{p\} \cup R$.*

PROOF. In the procedure, suppose a is nearest to p, b is nearest to q, and p is a γ-nearest neighbor of q. If $d(a, p) \geq d(b, q)$, then p has b as $(1+\gamma)$-nearest, using the triangle inequality, while if $d(a, p) \leq d(b, q)$, then q has a as $(1+\gamma)$-nearest. Moreover, if $d(p, q) \leq \gamma d(b, q)$, then $d(p, b) \leq d(p, q) + d(q, b) \leq (1+\gamma)d(b, q)$, and similar reasoning applies to $q \in T_a^{1+\gamma}$ with $d(p, q) \leq \gamma d(a, q)$. $\square$

Theorem 36 *Suppose metric space (V, d) has a γ-nearest neighbor bound, and $S \subset V$ is a set of n sites. Then the preprocessing time for the above data structure for inverse γ-nearest neighbor queries is the same as for $(1+\gamma)$-neighbor queries, and if p is random in $\{p\} \cup S$, then the expected query time is the time for a $(1+\gamma)$-neighbor query, plus $O(\mathcal{N}_{1+\gamma,\Upsilon(S)}) + O(\mathcal{N}_{2+\gamma,\Upsilon(S)})|S|/|R|$.*

PROOF. Every b examined is a $(1+\gamma)$-nearest neighbor of p, and the expected number of such b is bounded by Lemma 11. Every q examined has p as a $(2+\gamma)$-nearest neighbor, and is examined at most twice, once as a member of T_b^1 and once as a member of $S_a^{1+\gamma}$. The result follows from Lemma 10. $\square$

6 Conclusions

This paper has shown that nearest neighbor problems can be solved efficiently, even when no information about the distance measure is given explicitly. Versions of some of the algorithms given here have been implemented, with promising results which will be reported elsewhere.

It is worth noting that the failure probability analysis and query time analysis do not depend on the triangle inequality, but only the sphere-packing properties, and there could be (V, d) which is not even a metric space, but for which these results hold, or for which the algorithms are effective in practice.

While it is satisfying that the preprocessing for $M(S, Q)$ is nearly linear, the time bound is higher than one would like, and worse, the space for the construction degrades to $\Omega(n^2)$ in high dimension. Is there a different construction without this flaw?

Acknowledgments. I'd like to thank the anonymous referees for their thoughtful comments.

References

[Bri95] S. Brin. Near neighbor search in large metric spaces. In *Proc. 21st Int. Conf. on Very Large Data Bases*, pages 574–584, 1995.

[Cla83] K. L. Clarkson. Fast algorithms for the all nearest neighbors problem. In *Proc. 24th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 226–232, 1983.

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

[Cla92] K. L. Clarkson. Randomized geometric algorithms. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, volume 1 of *Lecture Notes Series on Computing*, pages 117–162. World Scientific, Singapore, 1992.

[FC96] C. Faloutsos and V. Caede. Analysis of n-dimensional quadtrees using the Hausdorff fractal dimension. In *Proc. 22nd Int. Conf. on Very Large Data Bases*, 1996.

[FS82] C.D. Feustel and L. G. Shapiro. The nearest neighbor problem in an abstract metric space. *Pattern Recognition Letters*, 1:125–128, December 1982.

[MR86] R. Motwani and P. Raghavan. Deferred data structuring: query-driven preprocessing for geometric search problems. In *Proc. 2nd Annu. ACM Sympos. Comput. Geom.*, pages 303–312, 1986.

[Mul93] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms.* Prentice Hall, Englewood Cliffs, NJ, 1993.

[Uhl91] J. K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Inform. Proc. Letters*, 40:175–179, 1991.

[Vai89] P. M. Vaidya. An $O(n \log n)$ algorithm for the all-nearest-neighbors problem. *Discrete Comput. Geom.*, 4:101–115, 1989.

[Yia93] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, pages 311–321, 1993.