Approximate Reverse Nearest Neighbors Search in High Dimensions using Locality-Sensitive Hashing

David Arthur, Steve Y. Oudot

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Approximate Reverse Nearest Neighbors Search in High Dimensions using Locality-Sensitive Hashing

David Arthur — Steve Y. Oudot

N° 7084 — version 3

initial version November 2009 — revised version October 2010

ISSN 0249-6399 ISRN INRIA/RR--7084--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SACLAY - ÎLE-DE-FRANCE

Approximate Reverse Nearest Neighbors Search in High Dimensions using Locality-Sensitive Hashing

David Arthur* , Steve Y. Oudot†

Thème : SYM — Systèmes symboliques
Équipe-Projet Geometrica

Rapport de recherche n° 7084 — version 3 — initial version November 2009 — revised version October 2010 — 27 pages

Abstract: We investigate the problem of finding approximate reverse nearest neighbors efficiently in high dimensions. Given a point cloud P and a parameter ε, our goal is to preprocess P in a way that enables us to quickly return the set of reverse nearest neighbors of any query point q among the points of P, plus possibly a small set of false positives that are ε-close to being true reverse nearest neighbors. Although provable solutions exist for this problem in low or fixed dimensions, to this date the methods proposed in high dimensions are mostly heuristic. We propose a method that is both provably-good and provably-efficient in all dimensions, based on a reduction of the problem to a small number of instances of the now classical ε-nearest neighbor search and r-neighbors reporting problems. Although the former has been extensively studied and elegantly solved in high dimensions using Locality-Sensitive Hashing techniques (LSH), the latter has a complexity that is still not well understood. We propose a new analysis of the LSH scheme for r-neighbors reporting, which brings out a meaningful output-sensitive term in the complexity of the problem, and which down the road enables us to solve the approximate reverse nearest problem, thanks to our reduction. Along the way, we propose a method to perform exact nearest neighbor search, whose analysis sheds new light on the problem by introducing a notion of *condition number* measuring the inherent complexity of a given instance.

Key-words: nearest neighbor search, reverse nearest neighbor search, locality-sensitive hashing.

* Department of Computer Science, Stanford University. Email: darthur@cs.stanford.edu
† Geometrica group, INRIA Saclay – Île-de-France. Email: steve.oudot@inria.fr

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université
4, rue Jacques Monod, 91893 ORSAY Cedex
Téléphone : +33 1 72 92 59 00

Recherche approchée de plus proches voisins inverses en grandes dimensions par hachage géométrique

Résumé : Nous étudions le problème de la recherche efficace de plus proches voisins inverses en grandes dimensions. Étant donné un nuage de points P et un paramètre ε, notre objectif est de pré-traiter le nuage P de telle sorte à pouvoir trouver rapidement l'ensemble des plus proches voisins inverses d'un point de requête q quelconque, plus éventuellement un petit nombre de faux positifs qui sont proches d'être des plus proches voisins inverses de q. Alors que des solutions efficaces et prouvées existent pour ce problème en dimensions petites ou fixées, à ce jour les méthodes proposées en grandes dimensions sont essentiellement heuristiques. Nous proposons une méthode à la fois efficace et prouvée en toutes dimensions, basée sur une réduction du problème à un petit nombre d'instances des problèmes classiques de recherche de plus proche voisin approché et de recherche exhaustive de voisins à distance r fixée. La complexité intrinsèque de ce dernier problème reste peu connue. Nous proposons une nouvelle analyse du comportement du hachage géométrique (LSH) sur ce problème, qui met en évidence une borne dépendant de la taille de la sortie, et qui au final, grâce à notre réduction, nous permet de résoudre le problème de la recherche de plus proches voisins inverses en temps raisonnable. Dans la foulée nous proposons également une méthode pour effectuer des recherches de plus proches voisins exacts, dont l'analyse éclaire le problème d'une nouvelle manière en introduisant une notion de *conditionnement* qui mesure la difficulté intrinsèque d'une instance particulière du problème.

Mots-clés : recherche de plus proche voisin, recherche de plus proches voisins inverses, hachage géométrique

1 Introduction

Proximity queries are ubiquitous in science and engineering, and given their natural importance they have received a lot of attention from the computer science community [10, 12, 18, 27]. *Nearest Neighbor* ($\mathcal{NN}$) search is certainly the most popular among them, both from a theoretical and from a practical points of view. It is defined as follows: given a finite point cloud P lying in some metric space (X, d), preprocess P in such a way that, for any query point $q \in X$, a nearest neighbor of q among the set $P \setminus \{q\}$ can be retrieved quickly. The $\mathcal{NN}$ query can be easily answered in linear time by brute force search, so the algorithmic challenge is to pre-process the data points so as to be able to find the answer in sub-linear time. This emphasis on reducing the query time stems from the fact that typical applications require to answer many $\mathcal{NN}$ queries on the same point set (see Shakhnarovich et al. [27] for a list of sample applications). Numerous methods have been proposed to solve the $\mathcal{NN}$ problem, however their performances degrade as the dimensionality d of the data increases – a phenomenon known as the *curse of dimensionality*. Typically, they suffer from either space or query time that is exponential in d, and in fact their performances become no better than the ones of simple brute-force search when d becomes higher than a few dozens or a few hundreds, both in theory and in practice [32].

In light of the apparent hardness of $\mathcal{NN}$ search, researchers have proposed an approximate version called ε-$\mathcal{NN}$ search, whose answer can be any point of $P \setminus \{q\}$ whose distance to the query point is within a factor $(1+\varepsilon)$ of the distance of q to its true nearest neighbor [4, 9, 19, 21, 24]. This version of the problem proved to be easier to solve efficiently, and, inspired from the random projection techniques developed by Kleinberg [21], two breakthrough results for ε-$\mathcal{NN}$ were obtained independently and on the same year by Kushilevitz et al. [24] and by Indyk and Motwani [19]. Both papers gave data structures that can answer ε-$\mathcal{NN}$ queries with truly sublinear (in n) runtime, using an amount of space that is polynomial in n, d, and $1/\varepsilon$. The currently known fastest algorithms for ε-$\mathcal{NN}$ search in high dimensions (see Andoni and Indyk [3] for a survey treatment) are based on the idea of Locality-Sensitive Hashing (LSH), first introduced by Indyk and Motwani [19]. In their seminal paper, they reduce the ε-$\mathcal{NN}$ problem to a decision version that asks to decide whether the distance from the query point q to the point set P is at most r or greater than $r(1+\varepsilon)$. In other words, the decision problem asks to locate q with respect to the union of balls of radius r about the data points, within some admissible degree of uncertainty prescribed by parameter ε. This query is therefore known as (r, ε)-$\mathcal{PLEB}$ (*Point Location among Equal Balls*). The space decomposition proposed by Indyk and Motwani [19] reduces the ε-$\mathcal{NN}$ problem to a poly-logarithmic number of (r, ε)-$\mathcal{PLEB}$ queries, for some suitable choices of r. Moreover, they show how to use LSH to answer every (r, ε)-$\mathcal{PLEB}$ query with high probability in time $O(dn^{\varrho}\mathrm{polylog}\, n)$ for some constant $\varrho < 1$, thus providing a fully sublinear-time procedure for solving ε-$\mathcal{NN}$. The LSH technique was originally designed for the Hamming cube, but later work by Datar et al. [13] and Andoni and Indyk [2] extended it to affine spaces $\mathbb{R}^d$ equipped with ℓ_p-norms, $p \in (0, 2]$.

In this paper we focus mainly on the reverse problem, known as *Reverse Nearest Neighbor* ($\mathcal{RNN}$) search, whose goal is to compute the *influence set* of a given query point. More precisely, given a finite point cloud P lying in some metric space (X, d), the problem asks to preprocess P in such a way that, for any query point $q \in X$, one can retrieve all those points $p \in P \setminus \{q\}$ that are closer to q than to $P \setminus \{p\}$. This set of *reverse nearest neighbors* is denoted by $\mathcal{RNN}_P(q)$. A relaxed version of $\mathcal{RNN}$ search, called ε-$\mathcal{RNN}$, asks to return any set $S \subseteq P \setminus \{q\}$ containing $\mathcal{RNN}_P(q)$, provided that all the points $p \in S$ satisfy $\mathrm{d}(p, q) \leq (1+\varepsilon)\mathrm{d}(p, p')$ for all $p' \in P \setminus \{p\}$. In other words, all the true reverse nearest neighbors of q must be found, plus possibly a number of false positives that are close to being actual reverse nearest neighbors of q.

$\mathcal{RNN}$ and ε-$\mathcal{RNN}$ queries arise in many different contexts[1], and it is no surprise that these two problems have received a lot of attention since their formal introduction by Korn and Muthukrishnan [22]. A wealth of methods have been proposed [1, 5, 11, 14, 20, 22, 23, 28, 29, 30, 31], which behave well in practice on some classes of inputs. However, these methods are mostly heuristic, and to date very little is known about the theoretical complexity of the $\mathcal{RNN}$ and ε-$\mathcal{RNN}$ problems, except in low [7, 25] or fixed [8] dimensions, where the dimensionality of the data can be considered as a mere constant. The crux of the matter is that, in contrast to ε-$\mathcal{NN}$, the answer to an $\mathcal{RNN}$ or ε-$\mathcal{RNN}$ query is not a single point but a set of points, whose size can be up to exponential in the ambient dimension [26], so there is no way to achieve a systematic sub-linear query time. Ideally, one would like to achieve a query time of the form $\tilde{O}(n^{\varrho} + |\mathcal{RNN}_P(q)|)$, where ϱ is a constant less than 1 and $|\mathcal{RNN}_P(q)|$ is the size of the true reverse nearest neighbors set. The big-$\tilde{O}$ notation may hide some additional factors that are polynomial in d and poly-logarithmic in n/ε. Intuitively, the first term in the bound represents the incompressible time needed to locate the query point q with respect to P, as in a standard $\mathcal{NN}$ query, while the second term represents the size of the sought-for answer.

Our contributions. Our main contribution is a reduction of the ε-$\mathcal{RNN}$ problem to one instance of ε-$\mathcal{NN}$ search plus a poly-logarithmic number of instances of *exhaustive* r-$\mathcal{PLEB}$ (Section 5). The latter query is a variant of $(r, 0)$-$\mathcal{PLEB}$ where not only one ball containing the query point q is sought for, but all such balls. Our reduction is based on a partitioning of the data points into buckets according to their nearest neighbor distances, combined with a pruning strategy that prevents the inspection of too many buckets at query time.

Turning our reduction into an effective algorithm for ε-$\mathcal{RNN}$ search requires to adapt the LSH scheme to exhaustive r-$\mathcal{PLEB}$ queries and to analyze its behavior, which we do in Section 3 using a somewhat refined concept of locality-sensitive hashing (see Definition 3.1). Although the adaptation itself is not novel (it was already proposed e.g. in [27, Chapter 1]), its analysis is, and it reflects the influence of the layout of the point cloud on the average query time (Theorem 3.2).

Down the road, these advances give a randomized algorithm for solving ε-$\mathcal{RNN}$ queries with high probability in expected $\tilde{O}(\frac{1}{\varepsilon^2} n^{\frac{1}{1+\varepsilon}} + \frac{1}{\varepsilon} n^{\frac{\varepsilon}{1+\varepsilon}} |O(\varepsilon)\text{-}\mathcal{RNN}_P(q)|)$ time using fully polynomial space, where n is the number of data points and where $O(\varepsilon)$-$\mathcal{RNN}_P(q)$ is a superset of $\mathcal{RNN}_P(q)$ whose points are close to being true reverse nearest neighbors of q (Theorems 5.4 through 5.7).

Along the way, we also propose a randomized algorithm for solving exact $\mathcal{NN}$ queries with high probability, whose expected running time is $\tilde{O}(\frac{1}{\varepsilon} n^{\frac{1}{1+\varepsilon}} + \frac{1}{\varepsilon} n^{\frac{\varepsilon}{1+\varepsilon}} |O(\varepsilon)\text{-}\mathcal{NN}_P(q)|)$ and whose memory usage is fully polynomial (Section 4, Theorems 4.3 through 4.5). The first term in the running time bound is sublinear in n and corresponds to a standard ε-$\mathcal{NN}$ query, while the second term depends on the size of the approximate nearest neighbors set $O(\varepsilon)$-$\mathcal{NN}_P(q)$, which thereby plays the role of a *condition number* measuring the discrepancy between the inherent difficulties of the exact and approximate $\mathcal{NN}$ queries on a given instance.

Let us emphasize that our complexity bounds hold in the d-dimensional Euclidean space $\mathbb{R}^d$ as well as in the d-dimensional Hamming cube $\mathbb{H}^d$, two spaces that are of practical interest. Perhaps surprisingly, the bounds are obtained by embedding the data and query points into some higher-dimensional spaces, typically of dimensions polynomial in d/ε. This stands in contrast to the general trend of LSH to reduce the dimensionality of the data. In fact, we show that the embeddings can be encoded implicitly in the algorithms, and that they can be re-interpreted as mere modifications of the family of hash functions in the original d-dimensional spaces (Sections 4.1, 4.2 and 5.5).

[1] Among these contexts, the multi-scale reconstruction of geometric structures from high-dimensional point cloud data using so-called *witness complexes* [6, 16] served as our initial motivation for this work.

Finally, for the sake of generality, we also derive complexity bounds in the general setting of a metric space that admits locality-sensitive families of hash functions. This makes our algorithms and their guarantees potentially applicable in other ambient spaces, such as for instance in $\mathbb{R}^d$ endowed with ℓ_p norms, $p \in (0, 2]$.

2 Preliminaries

In Section 2.1 we introduce some useful notation and state the approximate nearest neighbor and reverse nearest neighbor problems formally. In Sections 2.2 and 2.3 we give an overview of LSH and its application to approximate nearest neighbor search. The various data structures and algorithms introduced in this section will be used as black boxes in the rest of the paper.

2.1 Problem statements and notations

Throughout the paper, (X, d) denotes a metric space and P a finite subset of X. Given a point $x \in X$ and a real parameter $\varepsilon \geq 0$, let $\mathrm{d}(x, P)$ denote the distance of x to $P \setminus \{x\}$, that is: $\mathrm{d}(x, P) = \min\{\mathrm{d}(x, p) \mid p \in P \setminus \{x\}\}$. Given some parameter $r \geq 0$, let $\mathcal{B}(x, r)$ denote the metric ball of center x and radius r, and let $\mathcal{B}_P(x, r)$ be the set of points of $P \setminus \{x\}$ that lie within this ball. Then, $\mathcal{B}_P(x, \mathrm{d}(x, P))$ is the set of *nearest neighbors* of x among $P \setminus \{x\}$, noted $\mathcal{NN}_P(x)$. By analogy, ε-$\mathcal{NN}_P(x)$ denotes the set $\mathcal{B}_P(x, (1+\varepsilon)\mathrm{d}(x, P))$ of *ε-nearest neigbors* of x among $P \setminus \{x\}$. The usual convention is that point x itself is excluded from these sets, which is not mentioned explicitly in our notations but will be admitted implicitly throughout the paper.

Problem 1 ($\mathcal{NN}$)**.** *Given a query point $q \in X$, the* nearest neighbor query *asks to retrieve any point of $\mathcal{NN}_P(q)$.*

Problem 2 (ε-$\mathcal{NN}$)**.** *Given a query point $q \in X$, the* ε-nearest neighbor query *asks to retrieve any point of ε-$\mathcal{NN}_P(q)$.*

Given now a point $x \in X$, let $\mathcal{RNN}_P(x)$ denote the set of *reverse nearest neighbors* of x among $P \setminus \{x\}$, which by definition are the points $p \in P \setminus \{x\}$ such that $x \in \mathcal{NN}_{P \cup \{x\}}(p)$. By analogy, let ε-$\mathcal{RNN}_P(x)$ denote the set of *reverse ε-nearest neighbors* of x among $P \setminus \{x\}$, which by definition are the points $p \in P \setminus \{x\}$ such that $x \in \varepsilon$-$\mathcal{NN}_{P \cup \{x\}}(p)$. Here again, point x itself is excluded from the various sets, a fact omitted in our notations for simplicity but admitted implicitly.

Problem 3 ($\mathcal{RNN}$)**.** *Given a query point $q \in X$, the* reverse nearest neighbors query *asks to retrieve the set $\mathcal{RNN}_P(q)$.*

Problem 4 (ε-$\mathcal{RNN}$)**.** *Given a query point $q \in X$, the* reverse ε-nearest neighbors query *asks to retrieve any set S such that $\mathcal{RNN}_P(q) \subseteq S \subseteq \varepsilon$-$\mathcal{RNN}_P(q)$.*

In other words, the goal of an approximate $\mathcal{RNN}$ query is to find all the true reverse nearest neighbors of q, plus possibly some additional false positives that are *close to being* reverse nearest neighbors of q. Parameter ε controls both the number and the quality of the false positives. In particular, when $\varepsilon = 0$, the true reverse nearest neighbors set must be returned.

2.2 Reducing approximate nearest neighbor search to its decision version

Given a parameter r, the decision version of Problem 1 consists in deciding whether $\mathrm{d}(q, P)$ is smaller or larger than r. This problem is also known as *Point Location among Equal r-Balls* (r-$\mathcal{PLEB}$) in the literature, because it is equivalent to deciding whether q lies inside the union of balls of same radius r about the points of P. It is formalized as follows:

Problem 5 (r-$\mathcal{PLEB}$)**.** *Given a query point $q \in X$, the r-$\mathcal{PLEB}$ query asks the following:*

- *if* $\mathrm{d}(q, P) \leq r$*, then return* YES *and any point $p \in P$ such that* $\mathrm{d}(p, q) \leq r$*;*

- *else* ($\mathrm{d}(q,P) > r$)*, return* NO.

By analogy, the decision version of Problem 2 consists in deciding whether $\mathrm{d}(q,P)$ is smaller than r or larger than $r(1+\varepsilon)$. If it lies between these two bounds, then any answer is acceptable. The formal statement is the following:

Problem 6 ((r,ε)-$\mathcal{PLEB}$)**.** *Given a query point $q \in X$, the (r,ε)-$\mathcal{PLEB}$ query asks the following:*

- *if* $\mathrm{d}(q,P) \leq r$*, then return* YES *and any point $p \in P$ such that* $\mathrm{d}(p,q) \leq r(1+\varepsilon)$*;*
- *if* $\mathrm{d}(q,P) > r(1+\varepsilon)$*, then return* NO*;*
- *else* ($r < \mathrm{d}(q,P) \leq r(1+\varepsilon)$)*, return any of the above answers.*

In the Hamming cube $\mathbb{H}^d$ endowed with the standard Hamming distance, it is straightforward to reduce ε-$\mathcal{RNN}$ to a small number of instances of (r,ε)-$\mathcal{PLEB}$: simply perform a binary search among the set of powers of $1+\varepsilon$ within the range $[1,d]$. For every value r considered, the (r,ε)-$\mathcal{PLEB}$ determines whether to eliminate the values below r or the ones above r from the set, until there is only one value left in the set. The number of iterations is $O(\log_2 \log_{1+\varepsilon} d) = O(\log_2 \frac{1}{\varepsilon} + \log_2 \log_2 d)$.

In the Euclidean space $\mathbb{R}^d$, the original LSH paper [19] showed a construction that reduces the ε-$\mathcal{NN}$ problem to a logarithmic number of (r,ε)-$\mathcal{NN}$ queries. More general constructions have since been proposed, and in this paper we will make use of the following one, proposed by Har-Peled [17], which is simple and works in any metric space. It is based on a divide-and-conquer strategy, building a tree $\mathcal{T}(P,\varepsilon)$ of height $O(\log_2 n)$, such that each node v is assigned a subset $P_v \subseteq P$ and an interval $[r_v, R_v]$ of possible values for parameter r. Each ε-$\mathcal{NN}$ query is performed by traversing down the search tree $\mathcal{T}(P,\varepsilon)$, and by answering two (r,ε)-$\mathcal{PLEB}$ queries at each node v to decide (approximately) whether $\mathrm{d}(q,P)$ belongs to the interval $[r_v,R_v]$ or not: in the former case, $O(\log_2 \log_{1+\varepsilon} \frac{R_v}{r_v})$ queries of type (r,ε)-$\mathcal{PLEB}$ are sufficient for determining precisely where $\mathrm{d}(q,P)$ lies in the interval and for returning a point of ε-$\mathcal{NN}_P(q)$; in the latter case, the choice of the child of v in which to continue the search is determined from the output of the two (r,ε)-$\mathcal{PLEB}$ queries. In this construction, the ratio $\frac{R_v}{r_v}$ is guaranteed to be at most a polynomial in $\frac{n}{\varepsilon}$, with bounded degree, so we have $O(\log_{1+\varepsilon} \frac{R_v}{r_v}) = O(\log_{1+\varepsilon} \frac{n}{\varepsilon})$. Thus,

Theorem 2.1 (see Chapter 14 of [17])**.** *Given a finite set $P \subseteq X$ with n points, the tree $\mathcal{T}(P,\varepsilon)$ stores $O(\frac{1}{\varepsilon}\log_2 \frac{n}{\varepsilon})$ data structures for (r,ε)-$\mathcal{PLEB}$ queries per node, and it reduces every ε-$\mathcal{NN}$ query to a set of $O(\log_2 n + \log_2 \frac{1}{\varepsilon} + \log_2 \log_2 \frac{n}{\varepsilon})$ queries of type (r,ε)-$\mathcal{PLEB}$.*

2.3 Solving (r,ε)-$\mathcal{PLEB}$ queries using Locality-Sensitive Hashing

Definition 2.2. *Given a metric space (X,d) and two radii $r_1 < r_2$, a family $\mathcal{F} = \{f : X \to \mathbb{Z}\}$ of hash functions is called* (r_1,r_2,p_1,p_2)-sensitive *if there exist quantities $1 > p_1 > p_2 > 0$ such that $\forall x,y \in X$,*

- $\mathrm{d}(x,y) \leq r_1 \Rightarrow \mathrm{Prob}\,[f(x) = f(y)] \geq p_1$*,*
- $\mathrm{d}(x,y) \geq r_2 \Rightarrow \mathrm{Prob}\,[f(x) = f(y)] \leq p_2$*,*

where probabilities are given for a random choice of hash function $f \in \mathcal{F}$ according to some probability distribution associated with the family.

Intuitively, an (r_1,r_2,p_1,p_2)-sensitive family of hash functions can distinguish points that are close together from points that are far away from each other. Such families exist at least when X is the Hamming cube $\mathbb{H}^d$ endowed with the standard Hamming distance [19], or $\mathbb{R}^d$ endowed with an ℓ_p norm such that $p \in [1,2]$ [13].

Provided that a $(r, r(1+\varepsilon), p_1, p_2)$-sensitive family $\mathcal{F}$ of hash functions is given, it is possible to answer (r,ε)-$\mathcal{PLEB}$ queries in sub-linear time [15, 19]. The algorithm proceeds as follows:

- In the pre-processing phase, it *boosts* the sensitivity of the family $\mathcal{F}$ by building k-dimensional vectors $g = (f_1, \cdots, f_k) : X \to \mathbb{R}^k$ whose coordinate functions f_i are drawn independently at random from $\mathcal{F}$. The hash key of a point $x \in X$ is now a k-dimensional vector $g(x) = (f_1(x), \cdots, f_k(x))$, and two keys $g(x)$ and $g(y)$ are considered equal if and only if $f_i(x) = f_i(y)$ for all $i = 1, \cdots, k$. Call $\mathcal{G}$ the family of such random hash vectors. The algorithm draws L elements $g_1, \cdots, g_L$ independently from $\mathcal{G}$, and it builds the L corresponding hash tables $H_1, \cdots, H_L$. It then hashes every data point $p \in P$ into every hash table H_i using vector $g_i(p)$.
- In the online query phase, the algorithm hashes the query point q into each of the L hash tables, and it retrieves all the points colliding with q therein, until either some point $p \in \mathcal{B}_P(q, r(1+\varepsilon))$ has been found or more than $3L$ points (including duplicates) have been retrieved in total. In the former case the algorithm answers YES and returns p, while in the latter case it answers NO. It also answers NO if no point of $\mathcal{B}_P(q, r(1+\varepsilon))$ has been found after visiting all the hash tables.

Letting $k = \lfloor \frac{\ln n}{\ln 1/p_2} \rfloor$ and $L = n^\varrho$, where $\varrho = \frac{\ln p_1}{\ln p_2}$, one can prove that this procedure gives the correct answer with constant probability [15, 19]. By repeating it a logarithmic number of times, one can increase the probability of success to $1 - \frac{1}{n^s}$, for any fixed degree s. Thus,

Theorem 2.3 (see [15, 19])**.** *Given a finite set $P \subseteq X$ with n points, two parameters $r, \varepsilon \geq 0$, and a $(r, r(1+\varepsilon), p_1, p_2)$-sensitive family $\mathcal{F}$ of hash functions for some constants $p_1 > p_2$, it is possible to construct a data structure of size $O(n^{1+\varrho} \log_2 n)$ that answers (r, ε)-$\mathcal{PLEB}$ queries correctly with high probability in $O(n^\varrho \log_{1/p_2} n \log_2 n)$ time, where $\varrho = \frac{\ln p_1}{\ln p_2}$.*

Note that the running time bound ignores the time needed to compute distances and to evaluate hash functions, because these depend on the ambient space X and on the hash family $\mathcal{F}$. In the practical scenarii considered in the paper, the metric space X will be a d-dimensional Hamming cube or Euclidean space, where each distance computation or hash function evaluation will take $O(d)$. The induced polynomial factors in d will be omitted from our bounds for simplicity, as in previous literature [18]. Note also that the quantity $1/p_2$ will be considered as a constant in the sequel, so $\log_{1/p_2} n$ will be replaced by $\log_2 n$ in the running time bound.

Focusing back on Har-Peled's construction, let us recall from Theorem 2.1 that each node v of the tree $\mathcal{T}(P, \varepsilon)$ stores $O(\frac{1}{\varepsilon} \log_2 \frac{n}{\varepsilon})$ data structures for answering (r, ε)-$\mathcal{PLEB}$ queries, each of size $O(|P_v|^{1+\varrho} \log_2 |P_v|)$. Let us also point out that by construction the subsets of P assigned to the sons of v form a partition of P_v. Then, an easy recursion gives the following bounds on the size of $T(P, \varepsilon)$ and on the query time:

Corollary 2.4. *Given a finite set $P \subseteq X$ with n points, if for some $\varepsilon > 0$ the ambient space X admits $(r, r(1+\varepsilon), p_1, p_2)$-sensitive families of hash functions for all values $r > 0$, then, combined with the appropriate (r, ε)-$\mathcal{PLEB}$ data structures, $\mathcal{T}(P, \varepsilon)$ answers ε-$\mathcal{NN}$ queries correctly with high probability in $O\left(n^\varrho \log_2^2 n (\log_2 n + \log_2 \frac{1}{\varepsilon} + \log_2 \log_2 \frac{n}{\varepsilon})\right)$ time using $O\left(\frac{1}{\varepsilon} n^{1+\varrho} \log_2 n \log_2 \frac{n}{\varepsilon}\right)$ space, where $\varrho = \frac{\ln p_1}{\ln p_2}$.*

From now on we will hide poly-logarithmic factors in $\frac{n}{\varepsilon}$ within big-$\tilde{O}$ notations, for the sake of simplicity. Thus, the time and space complexities given in Theorem 2.3 become respectively $\tilde{O}(n^\varrho)$ and $\tilde{O}(n^{1+\varrho})$, while those given in Corollary 2.4 become respectively $\tilde{O}(n^\varrho)$ and $\tilde{O}(\frac{1}{\varepsilon} n^{1+\varrho})$.

3 Exhaustive r-$\mathcal{PLEB}$

In the rest of the paper, we will use a variant of r-$\mathcal{PLEB}$ called *exhaustive* r-$\mathcal{PLEB}$, where not only one arbitrary ball containing the query point is sought for, but all such balls:

Problem 7 (Exhaustive r-$\mathcal{PLEB}$)**.** *Given a finite set $P \subseteq X$ and a query point $q \in X$, the* exhaustive r-$\mathcal{PLEB}$ *query asks to return the set $\mathcal{B}_P(q,r)$.*

This problem was already introduced in [27, Chapter 1] under the name *near-neighbor reporting*, and a variant of the LSH scheme of Section 2.3 was proposed for solving it. The difference with the original LSH scheme is that the query procedure does not stop when $3n^\varrho$ collisions with the query point q have been found, but rather it continues until all the points colliding with q in the n^ϱ hash tables have been retrieved. The output is then the subset of these points that lie within $\mathcal{B}_P(q,r)$. The details of the pre-processing and query phases are given in Algorithms 1 and 2 respectively. Note that parameter ε no longer controls the quality of the output, which, according to the analysis, coincides with the set $\mathcal{B}_P(q,r)$ with high probability. However, the choice of ε has an impact on the complexity of the procedure, as we will see later on.

Input : metric space (X, d), finite point set $P \subseteq X$, parameters $r, \varepsilon \geq 0$
Output: $\mathcal{A}(P, r, \varepsilon)$ data structure

Take an $(r, r(1+\varepsilon), p_1, p_2)$-sensitive LSH family $\mathcal{F}$
Let $\varrho = \frac{\ln p_1}{\ln p_2}$ and $k = \lfloor \frac{\ln n}{\ln 1/p_2} \rfloor$
Create the k-dimensional hash family $\mathcal{G}$ as described in Section 2.3
for $i = 1$ *to* $\lceil c \ln n \rceil$ `// c is a constant to be explicited later`
do
  pick n^ϱ functions $\{g_1, \ldots, g_{n^\varrho}\}$ independently at random from $\mathcal{G}$
  Create the corresponding hash tables $\{H_1, \ldots, H_{n^\varrho}\}$
  forall $p \in P$ **do**
    for $j = 1$ *to* n^ϱ **do**
      Insert p into H_j using the key $g_j(p)$
    end
  end
  Construct the data structure $\mathcal{A}_i(P, r, \varepsilon) := \{g_1, \cdots, g_{n^\varrho}\} \sqcup \{H_1, \cdots, H_{n^\varrho}\}$
end
Output $\mathcal{A}(P, r, \varepsilon) := \bigcup_i \mathcal{A}_i(P, r, \varepsilon)$

Algorithm 1: *Pre-processing phase for exhaustive r-$\mathcal{PLEB}$*

In the remainder of the section we propose a new analysis of this modified LSH scheme, which refines the analysis given in [27, Chapters 1 and 3] and better reflects the influence of the layout of the point cloud on the average query time. We also answer a question raised at the end of [27, §3.3] by showing that the same choice of parameters k, L as in Section 2.3 is sufficient to guarantee our bound on the average query time for any query point. Our analysis relies on the following refined concept of locality-sensitive family of hash functions[2]:

Definition 3.1. *Given a metric space (X, d) and positive radii $r_0 \leq r_1 < r_2$, a family $\mathcal{F} = \{f : X \to \mathbb{Z}\}$ of hash functions is called* $(r_0, r_1, r_2, p_0, p_1, p_2)$-sensitive *if there exist quantities $1 > p_0 \geq p_1 > p_2 \geq 0$ such that $\forall x, y \in X$,*

(i) $\mathrm{d}(x,y) \leq r_1 \Rightarrow \mathrm{Prob}\,[f(x) = f(y)] \geq p_1$,
(ii) $\mathrm{d}(x,y) \geq r_2 \Rightarrow \mathrm{Prob}\,[f(x) = f(y)] \leq p_2$,
(iii) $\mathrm{d}(x,y) \geq r_0 \Rightarrow \mathrm{Prob}\,[f(x) = f(y)] \leq p_0$,

where probabilities are given for a random choice of hash function $f \in \mathcal{F}$ according to some probability distribution associated with the family.

[2] An even finer concept, proposed in [27, § 3.3], makes the probability of having $f(x) = f(y)$ a function of the distance between x and y. However, for our purpose it is not necessary to go to this level of refinement.

Input : metric space (X, d), $\mathcal{A}(P, r, \varepsilon)$ data structure, query point $q \in X$

Let ϱ, k and c be defined as in Algorithm 1
Initialize the output set: $S := \emptyset$
for $i = 1$ *to* $\lceil c \ln n \rceil$ **do**
  Let $\{g_1, \ldots, g_{n^\varrho}\}$ be the functions and $\{H_1, \ldots, H_{n^\varrho}\}$ the tables contained in $\mathcal{A}_i(P, r, \varepsilon)$
  for $j = 1$ *to* n^ϱ **do**
    Compute $g_j(q)$ and retrieve the set C_j of the points colliding with q in H_j
    forall $p \in C_j$ **do**
      if $\mathrm{d}(q, p) \leq r$ **then**
        Update the output set: $S := S \cup \{p\}$
      end
    end
  end
end
Return S

Algorithm 2: *Online query phase for exhaustive r-$\mathcal{PLEB}$*

Axioms (i) and (ii) correspond to the classical notion of locality-sensitive family of hash function (Definition 2.2). They do not make it possible to limit the number of collisions between the query point q and the points of $\mathcal{B}_P(q, r_1)$ in the analysis of exhaustive r_1-$\mathcal{PLEB}$ queries. Specifically, every point of $\mathcal{B}_P(q, r_1)$ might collide with q in every hash table in theory, thus raising the cost of an exhaustive r_1-$\mathcal{PLEB}$ query to $\Omega(n^\varrho)$ per point of $\mathcal{B}_P(q, r_1)$. This is in fact all theoretical, since in practice the hash functions should make a difference between those points of $\mathcal{B}_P(q, r_1)$ that are really close to q and those that are farther away. This is the reason for introducing the third axiom (iii), which will show its usefulness in the subsequent sections of the paper, where most of the points of P will be guaranteed to lie farther than r_0 from q. For now, we will work out the following guarantees for r-$\mathcal{PLEB}$ queries:

Theorem 3.2. *Given a finite set $P \subseteq X$ with n points and two parameters $r, \varepsilon \geq 0$, if (X, d) admits a (r_1, r_2, p_1, p_2)-sensitive family $\mathcal{F}$ of hash functions with $r_1 = r$ and $r_2 \leq r(1+\varepsilon)$, then Algorithm 2 answers exhaustive r-$\mathcal{PLEB}$ queries correctly with high probability in expected $\tilde{O}(n^\varrho(1 + |\mathcal{B}_P(q, r(1+\varepsilon))|))$ time, involving $\tilde{O}(n^\varrho + |\mathcal{B}_P(q, r(1+\varepsilon))|)$ distance computations and hash function evaluations only, and using $\tilde{O}(n^{1+\varrho})$ space, where $\varrho = \frac{\ln p_1}{\ln p_2}$. If moreover the family $\mathcal{F}$ is $(r_0, r_1, r_2, p_0, p_1, p_2)$-sensitive for some $r_0 \leq r_1$, then on any input point $q \in X$ the algorithm answers exhaustive r-$\mathcal{PLEB}$ queries in expected $\tilde{O}(n^\varrho(1 + |\mathcal{B}_P(q, r_0)|) + n^\alpha |\mathcal{B}_P(q, r(1+\varepsilon)) \setminus \mathcal{B}(q, r_0)|)$ time, where $\alpha = \varrho(1 - \frac{\ln p_0}{\ln p_1}) \leq \varrho$.*

The first term (n^ϱ) in the running time bound corresponds to the complexity of a standard (r, ε)-$\mathcal{PLEB}$ query and can be viewed as the incompressible time needed to locate the query point q in the data structure. The second term $(n^\varrho |\mathcal{B}_P(q, r_0)|)$ arises from the fact that points closer to q than r_0 may collide up to n^ϱ times with q each. Finally, the third term $(n^\alpha |\mathcal{B}_P(q, r(1+\varepsilon)) \setminus \mathcal{B}(q, r_0)|)$ arises from the fact that points lying farther away from q only collide up to n^α times with q each, for some $\alpha \leq \varrho$. Note that the less sensitive the family $\mathcal{F}$ between radii r_0 and r_1, the closer to 1 the ratio $\frac{\ln p_0}{\ln p_1}$, and therefore the smaller α compared to ϱ.

The rest of the section is devoted to the proof of Theorem 3.2, which is divided into three parts: (1) proving the correctness of the output of Algorithm 2 with high probability, (2) bounding the expected query time, and (3) bounding the size of the data structure.

3.1 Correctness of the output

Note that the test on line 8 of Algorithm 2 ensures that the output set S is always a subset of $\mathcal{B}_P(q,r)$. Thus, we only need to show that S contains all the points of $\mathcal{B}_P(q,r)$ with high probability at the end of the query.

Lemma 3.3. *$\mathcal{B}_P(q,r) \subseteq S$ with probability at least $1 - n^{1-c\ln\frac{5}{2}}$.*

This result means that the probability of success of the query is high, even for small values of c. For instance, it is at least $1 - \frac{1}{n}$ for $c \geq \frac{2}{\ln\frac{5}{2}}$, and more generally it is at least $1 - \frac{1}{n^i}$ for $c \geq \frac{1+i}{\ln\frac{5}{2}}$.

Proof of the lemma. Let p be a point of $\mathcal{B}_P(q,r)$. Consider a single loop i of Algorithm 2, and let us show that p is inserted in the output set S during this loop with constant probability. This is equivalent to showing that, with constant probability, there exists some function $g_j(\cdot)$ that hashes q and p to the same location ($g_j(q) = g_j(p)$) during the loop. Since $d(q,p) \leq r$, the probability of a collision for a fixed j is at least $p_1^k = p_1^{\lfloor\frac{\ln n}{\ln 1/p_2}\rfloor} = e^{-\ln 1/p_1 \lfloor\frac{\ln n}{\ln 1/p_2}\rfloor} \geq e^{-\ln 1/p_1 \frac{\ln n}{\ln 1/p_2}} = n^{-\frac{\ln 1/p_1}{\ln 1/p_2}} = n^{-\varrho}$. Therefore, the probability that no hash function g_j generates a collision is at most $(1-n^{-\varrho})^{n^\varrho}$ since n^ϱ functions are picked from $\mathcal{G}$ at each loop of Algorithm 2. Thus, the probability that loop i inserts p into the output set S is at least $1 - (1-n^{-\varrho})^{n^\varrho} \geq 1 - \frac{1}{e} > \frac{3}{5}$.

Now, there are $\lceil c\ln n\rceil$ iterations in total, with independent hash functions, so the probability that $p \notin S$ at the end of the query is at most $\left(\frac{2}{5}\right)^{\lceil c\ln n\rceil} = e^{\ln\frac{2}{5}\lceil c\ln n\rceil} \leq n^{c\ln\frac{2}{5}}$. Applying the union bound on the set $\mathcal{B}_P(q,r)$, we obtain that the probability that all points of $\mathcal{B}_P(q,r)$ belong to S at the end of the query is at least $1 - |\mathcal{B}_P(q,r)|n^{c\ln\frac{2}{5}} \geq 1 - n^{1+c\ln\frac{2}{5}} = 1 - n^{1-c\ln\frac{5}{2}}$. □

Remark 3.4. *It is easily seen from the final paragraph of the proof of Lemma 3.3 that the correctness of the output can be guaranteed with probability $1 - m^{1-c\ln\frac{5}{2}}$ for any given $m \geq n$. Indeed, by running $\lceil c\ln m\rceil$ iterations of the main loops of Algorithms 1 and 2 instead of $\lceil c\ln n\rceil$ iterations, we obtain that each point of $\mathcal{B}_P(q,r)$ belongs to S at the end of the query with probability at least $1 - m^{-c\ln\frac{5}{2}}$, and thus that $\mathcal{B}_q(r,\subseteq)S$ with probability at least $1 - m^{1-c\ln\frac{5}{2}}$. This remark will be useful when dealing with $\mathcal{RNN}$ queries in Section 5.*

3.2 Expected query time

First of all, the query point q is hashed into $n^\varrho\lceil c\ln n\rceil = \tilde{O}(n^\varrho)$ hash tables in total, and each hashing operation involves $k = \lfloor\frac{\ln n}{\ln 1/p_2}\rfloor = O(\ln n)$ hash function evaluations, c and p_2 being constants here. Thus, the total time spent hashing q is $\tilde{O}(n^\varrho)$. There remains to bound the expected number of colllisions of q with points of P in the hash tables.

Lemma 3.5. *The expected total number of collisions of q with points of $P \setminus \mathcal{B}(q, r(1+\varepsilon))$ is $\tilde{O}(n^\varrho)$.*

Proof. Take an arbitrary loop i of Algorithm 2 and an arbitrary hash table H_j considered in that loop. Recall that the hash family $\mathcal{G}$ is constructed in Algorithm 1 by concatenating $k = \lfloor\frac{\ln n}{\ln 1/p_2}\rfloor$ functions drawn from a $(r, (1+\varepsilon)r, p_1, p_2)$-sensitive family $\mathcal{F}$. Therefore, the probability that a given point $p \in P\setminus\mathcal{B}(q, r(1+\varepsilon))$ collides with q in H_j is at most $p_2^k = p_2^{\lfloor\frac{\ln n}{\ln 1/p_2}\rfloor} = e^{\ln(p_2)\lfloor\frac{\ln n}{\ln 1/p_2}\rfloor} \leq e^{\ln(p_2)(\frac{\ln n}{\ln 1/p_2}-1)} = \frac{1}{p_2\, n}$. It follows that the expected number of points $p \in P \setminus \mathcal{B}(q, r(1+\varepsilon))$ that collide with q in H_j is at most $\frac{1}{p_2}$, from which we conclude that the expected total number of such collisions in all the hash tables at all iterations is at most $\frac{1}{p_2}n^\varrho\lceil c\ln n\rceil = \tilde{O}(n^\varrho)$. □

Without any further assumptions on the family $\mathcal{F}$ of hash functions, each point of $\mathcal{B}_P(q, r(1+\varepsilon))$ may collide with q in every hash table. The number of collisions of q with points of $\mathcal{B}_P(q, r(1+\varepsilon))$ is therefore $O(n^\varrho c \ln n |\mathcal{B}_P(q, r(1+\varepsilon))|) = \tilde{O}(n^\varrho |\mathcal{B}_P(q, r(1+\varepsilon))|)$, which, combined with Lemma 3.5, implies that the expected running time of the algorithm is $\tilde{O}(n^\varrho(1+|\mathcal{B}_P(q, r(1+\varepsilon))|))$, as claimed in the theorem. For every collision considered, a test is made on the distance between q and the colliding point of P (see line 8 of Algorithm 2). With a simple book-keeping, e.g. by marking the points of P that have already been considered during the query, we can afford to do the test at most once per point of P, thus yielding a total number of distance computations of the order of $\tilde{O}(n^\varrho + |\mathcal{B}_P(q, r(1+\varepsilon))|)$.

Consider now the stronger hypothesis that the family $\mathcal{F}$ of hash functions is $(r_0, r, r(1+\varepsilon), p_0, p_1, p_2)$-sensitive for some $r_0 \leq r_1$:

Lemma 3.6. *Assuming that $\mathcal{F}$ is $(r_0, r, r(1+\varepsilon), p_0, p_1, p_2)$-sensitive, the expected total number of collisions of q with points of $\mathcal{B}_P(q, r(1+\varepsilon)) \setminus \mathcal{B}(q, r_0)$ is $\tilde{O}(n^\alpha |\mathcal{B}_P(q, r(1+\varepsilon)) \setminus \mathcal{B}(q, r_0)|)$, where $\alpha = \varrho(1 - \frac{\ln p_0}{\ln p_1}) \leq \varrho$.*

Proof. Take an arbitrary loop i of Algorithm 2 and an arbitrary hash table H_j considered in that loop. The probability that a given point $p \in \mathcal{B}_P(q, r(1+\varepsilon)) \setminus \mathcal{B}(q, r_0)$ collides with q in H_j is at most $p_0^k = p_0^{\lfloor \frac{\ln n}{\ln 1/p_2} \rfloor} = e^{\ln p_0 \lfloor \frac{\ln n}{\ln 1/p_2} \rfloor} \leq e^{\ln p_0 (\frac{\ln n}{\ln 1/p_2} - 1)} = \frac{1}{p_0} n^{-\frac{\ln p_0}{\ln p_2}}$. It follows by additivity of expectation that the expected total number of collisions between p and q during the execution of the algorithm is at most $\frac{1}{p_0} n^\varrho n^{-\frac{\ln p_0}{\ln p_2}} \lceil c \ln n \rceil = \frac{1}{p_0} n^{\frac{\ln p_1}{\ln p_2} - \frac{\ln p_0}{\ln p_2}} \lceil c \ln n \rceil = \frac{1}{p_0} n^\alpha \lceil c \ln n \rceil = \tilde{O}(n^\alpha)$, where $\alpha = \frac{\ln p_1}{\ln p_2}(1 - \frac{\ln p_0}{\ln p_1})$. We conclude that the expected total number of collisions of q with points of $\mathcal{B}_P(q, r(1+\varepsilon)) \setminus \mathcal{B}(q, r_0)$ during the course of the algorithm is $\tilde{O}(n^\alpha |\mathcal{B}_P(q, r(1+\varepsilon)) \setminus \mathcal{B}(q, r_0)|)$. □

It follows from Lemma 3.6 that the expected query time becomes $\tilde{O}(n^\varrho(1 + |\mathcal{B}_P(q, r_0)|) + n^\alpha |\mathcal{B}_P(q, r(1+\varepsilon)) \setminus \mathcal{B}(q, r_0)|)$ when the family $\mathcal{F}$ of hash functions is $(r_0, r, r(1+\varepsilon), p_0, p_1, p_2)$-sensitive, as claimed in the theorem.

3.3 Size of the data structure

Each hash table contains one pointer per point of P, and there are $n^\varrho \lceil c \ln n \rceil$ such hash tables in total, so we need to store $\tilde{O}(n^{1+\varrho})$ pointers in total. In addition, we need to store the $n^\varrho \lceil c \ln n \rceil$ vectors of hash functions corresponding to the hash tables: each vector requires to store $\lfloor \frac{\ln n}{\ln 1/p_2} \rfloor$ hash functions, or rather their indices in the family $\mathcal{F}$. So, in total our data structure has a space complexity of $\tilde{O}(n^{1+\varrho})$. This bound ignores the cost of storing the input point cloud, which depends on the type of data representation. Typically, this cost will be $O(dn)$ in the d-dimensional Hamming cube or Euclidean space.

4 Interlude: from exhaustive r-$\mathcal{PLEB}$ to exact $\mathcal{NN}$

Before dealing with ε-$\mathcal{RNN}$ queries (the main topic of the paper), let us show a simple but pedagogical application of exhaustive r-$\mathcal{PLEB}$ queries to exact $\mathcal{NN}$ search. Given a set P of n points in the Euclidean space $\mathbb{R}^d$ or in the Hamming cube $\mathbb{H}^d$, we will show that $\mathcal{NN}$ queries can be solved exactly with high probability on any query point q in expected $\tilde{O}(\frac{1}{\varepsilon} n^{\frac{1}{1+\varepsilon}} + \frac{1}{\varepsilon} n^{\frac{\varepsilon}{1+\varepsilon}} |\varepsilon(1+2\varepsilon)\text{-}\mathcal{NN}_P(q)|)$ time using $\tilde{O}(\frac{1}{\varepsilon^2} n^{1+\frac{1}{1+\varepsilon}})$ space (Theorems 4.4 and 4.5). Our running time bound is composed of two terms: the first one is clearly sublinear in n, while the second one depends on the size of the approximate nearest neighbors set $\varepsilon(1+2\varepsilon)$-$\mathcal{NN}_P(q)$. Whether the bound will be sublinear in n or not in practice depends on the size of this set compared to $n^{\frac{1}{1+\varepsilon}}$. This follows the intuition that finding the exact nearest neighbor of q is easy when q does not have too many approximate nearest neighbors, and in this respect the quantity $|\varepsilon(2+\varepsilon)\text{-}\mathcal{NN}_P(q)|$

plays the role of a *condition number* measuring the inherent difficulty of a given instance of the exact $\mathcal{NN}$ problem. The interesting point to raise here is that the limit on this number for our algorithm to be sublinear is pretty high, of the order of $n^{\frac{1}{1+\varepsilon}}$. For comparison, the original concept of locality sensitive hashing (Definition 2.2) only yields a limit of the order of $n^{\frac{\varepsilon}{1+\varepsilon}}$ on the condition number.

The algorithm. The preprocessing phase consists of the following steps:

i. Choose an arbitrary parameter $\varepsilon > 0$.

ii. Build the tree structure $\mathcal{T}(P, \varepsilon)$ of Section 2.2 and its associated (r, ε)-$\mathcal{PLEB}$ data structures.

iii. For every (r, ε)-$\mathcal{PLEB}$ data structure built on a subset of P at step ii, build an $\mathcal{A}(P, r, \varepsilon)$ data structure using Algorithm 1.

Then, given a query point q, we proceed as follows:

1. Answer an ε-$\mathcal{NN}$ query using the tree structure $\mathcal{T}(P, \varepsilon)$, to obtain a value r such that $\mathrm{d}(q, P) \leq r \leq \mathrm{d}(q, P)(1 + \varepsilon)$ with high probability.

2. Answer an exhaustive r-$\mathcal{PLEB}$ query using Algorithm 2, to obtain a set S such that $S = \mathcal{B}_P(q, r)$ with high probability.

3. Iterate over the points of S and return the one that is closest to q. If S is empty, then return any arbitrary point of P.

Note that the execution of Algorithm 2 at step 2 is made possible by the fact that the algorithm solving the ε-$\mathcal{NN}$ query at step 1 returns a radius r that is stored in one of the $\mathcal{A}(P, r, \varepsilon)$ data structures built during the preprocessing phase. For any other value r we would not be able to perform step 2 because we would not have the corresponding $\mathcal{A}(P, r, \varepsilon)$ data structure at hand.

Analysis. We begin by showing the correctness of the query procedure:

Lemma 4.1. *The query procedure returns a point of $\mathcal{NN}_P(q)$ with high probability.*

Proof. Corollary 2.4 guarantees that the radius r computed at step 1 satisfies $\mathrm{d}(q, P) \leq r \leq \mathrm{d}(q, P)(1 + \varepsilon)$ with high probability. Under this condition, we have $\mathcal{NN}_P(q) \subseteq \mathcal{B}_P(q, r)$, and so Theorem 3.2 guarantees that the set S computed at step 2 contains $\mathcal{NN}_P(q)$ with high probability. It follows that the point returned at step 3 belongs to $\mathcal{NN}_P(q)$. □

Assume from now on that the ambient space admits $({}^{r}/_{(1+\varepsilon)}, r, r(1 + \varepsilon), p_0, p_1, p_2)$-sensitive families of hash functions for all values $r > 0$. The running time of step 1 is then $\tilde{O}(n^{\varrho})$, where $\varrho = \frac{\ln p_1}{\ln p_2}$, by Corollary 2.4. The running time of step 3 is $O(|S|)$, so it is dominated by the running time of step 2.

Lemma 4.2. *The expected running time of step 2 is $\tilde{O}(n^{\varrho} + n^{\alpha}|\varepsilon(2 + \varepsilon)\text{-}\mathcal{NN}_P(q)|)$, where $\varrho = \frac{\ln p_1}{\ln p_2}$ and $\alpha = \varrho(1 - \frac{\ln p_0}{\ln p_1})$.*

Proof. Let r be the radius computed at step 1. If $r \leq \mathrm{d}(q, P)(1+\varepsilon)$, then we have $\mathrm{d}(q, P) \geq \frac{r}{1+\varepsilon}$, and so the points of P lie at least $\frac{r}{1+\varepsilon}$ away from q. As a consequence, the expectation of the running time of step 2, conditional on the event that $r \leq \mathrm{d}(q, P)(1+\varepsilon)$, is $\tilde{O}(n^{\varrho}+n^{\alpha}|\mathcal{B}_P(q, r(1+\varepsilon))|) = \tilde{O}(n^{\varrho} + n^{\alpha}|\mathcal{B}_P(q, \mathrm{d}(q, P)(1 + \varepsilon)^2)|)$, by Theorem 3.2. By contrast, if $r > \mathrm{d}(q, P)(1 + \varepsilon)$, then we have no bound on the size of $\mathcal{B}_P(q, r(1 + \varepsilon))$ other than n, and Theorem 3.2 states that the number of collisions per point of $\mathcal{B}_P(q, r(1 + \varepsilon))$ can be up to $\tilde{O}(n^{\varrho})$. Hence, the

expectation of the running time of step 2, conditional on the event that $r > \mathrm{d}(q,P)(1+\varepsilon)$, is $\tilde{O}(n^\varrho(1+|\mathcal{B}_P(q,r(1+\varepsilon))|)) = \tilde{O}(n^\varrho(1+n))$. Now, recall from Section 2.3 that the event that $r > \mathrm{d}(q,P)(1+\varepsilon)$ only occurs with very low probability, more precisely with probability at most $\frac{1}{n^2}$ under suitable choices of parameters in the (r,ε)-$\mathcal{PLEB}$ data structures attached to the tree $\mathcal{T}(P,\varepsilon)$. Therefore, the expected running time of step 2 is bounded by $\tilde{O}(n^\varrho+n^\alpha|\mathcal{B}_P(q,\mathrm{d}(q,P)(1+\varepsilon)^2)|+\frac{1}{n^2}n^\varrho(1+n)) = \tilde{O}(n^\varrho+n^\alpha|\mathcal{B}_P(q,\mathrm{d}(q,P)(1+\varepsilon)^2)|) = \tilde{O}(n^\varrho+n^\alpha|\varepsilon(2+\varepsilon)\text{-}\mathcal{NN}_P(q)|)$. □

The total size of the tree $\mathcal{T}(P,\varepsilon)$ and associated (r,ε)-$\mathcal{PLEB}$ data structures is $\tilde{O}(\frac{1}{\varepsilon}n^{1+\varrho})$, by Corollary 2.4. In addition, since $\mathcal{T}(P,\varepsilon)$ has $\tilde{O}(n)$ nodes in total, each one storing $\tilde{O}(\frac{1}{\varepsilon})$ data structures for (r,ε)-$\mathcal{PLEB}$, the total number of $\mathcal{A}(P,r,\varepsilon)$ data structures built at step iii of the preprocessing phase is $\tilde{O}(\frac{n}{\varepsilon})$. Therefore, by Theorem 3.2, the total memory usage of the algorithm is $\tilde{O}(\frac{1}{\varepsilon}n^{2+\varrho})$. Thus,

Theorem 4.3. *Given a finite set $P \subseteq X$ with n points, if for some $\varepsilon > 0$ the ambient space X admits $(r/(1+\varepsilon), r, r(1+\varepsilon), p_0, p_1, p_2)$-sensitive families of hash functions for all values $r > 0$, then our algorithm answers exact $\mathcal{NN}$ queries with high probability in expected $\tilde{O}(n^\varrho + n^\alpha|\varepsilon(2+\varepsilon)\text{-}\mathcal{NN}_P(q)|)$ time using $\tilde{O}(\frac{1}{\varepsilon}n^{2+\varrho})$ space, where $\varrho = \frac{\ln p_1}{\ln p_2}$ and $\alpha = \varrho(1-\frac{\ln p_0}{\ln p_1})$.*

An important question to answer at this point is how small α can be made compared to ϱ for a given (fixed) parameter $\varepsilon > 0$. We will investigate two specific scenarios that are of practical interest: when X is the d-dimensional Euclidean space $\mathbb{R}^d$, and when X is the d-dimensional Hamming cube $\mathbb{H}^d$ endowed with the Hamming distance. As we will see, in both scenarios we can make α of the order of ε. Note that the user is free to choose any positive value for ε in practice. However, a trade-off needs to be made, since the smaller ε, the smaller the set $\varepsilon(2+\varepsilon)$-$\mathcal{NN}_P(q)$ and the smaller α compared to ϱ, but on the other hand the higher ϱ itself.

4.1 Euclidean case

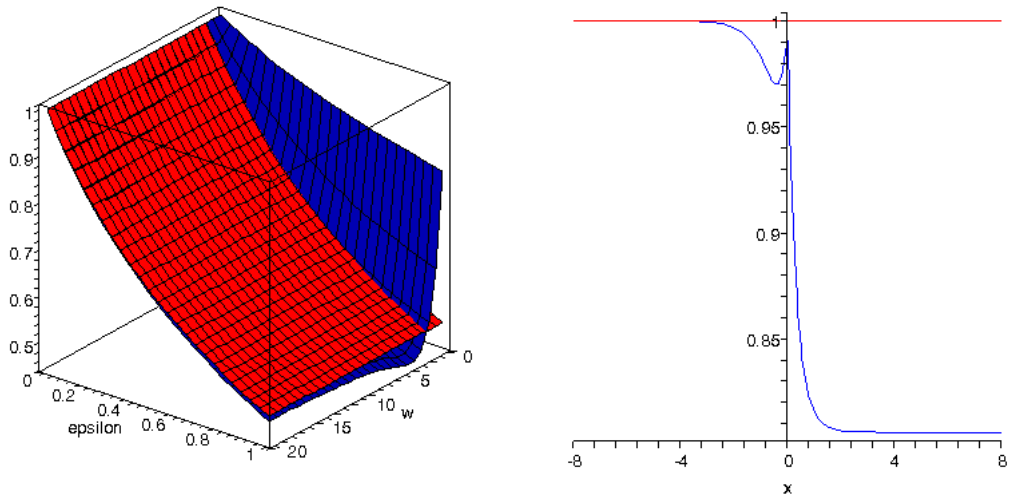

Figure 1: *Comparing $\varrho = \frac{\ln p_1}{\ln p_2}$ against $\frac{1}{1+\varepsilon}$ in Euclidean space. Left: plots of ϱ (blue) and $\frac{1}{1+\varepsilon}$ (red) versus ε and w. Right: plot of $\varrho(1+\varepsilon)$ (blue) versus ε on a logarithmic scale ($x = \log_{10}\varepsilon$), with $w = 2\max\{1,\varepsilon\}$.*

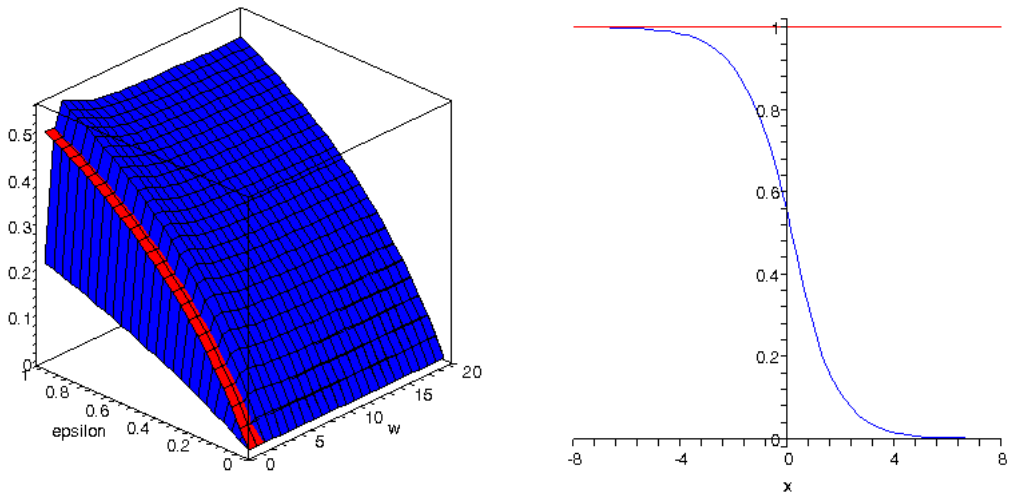

Figure 2: *Left: plots of* $\frac{\alpha}{\varrho} = 1 - \frac{\ln p_0}{\ln p_1}$ *(blue) and* $\frac{\varepsilon}{1+\varepsilon}$ *(red) versus* ε *and* w*. Right: plot of* $\frac{\alpha}{\varepsilon\varrho}$ *(blue) versus* ε *on a logarithmic scale* ($x = \log_{10}\varepsilon$)*, with* $w = \frac{2}{\varepsilon}$*.*

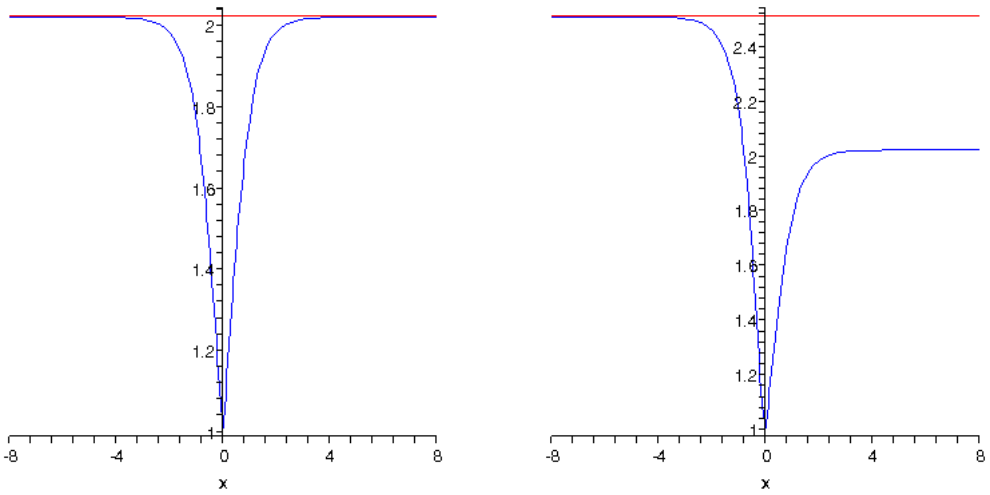

Figure 3: *Left: plot of* $\frac{1}{\ln 1/p_2}$ *(blue) versus* ε *on a logarithmic scale* ($x = \log_{10}\varepsilon$)*, with* $w = 2\max\{1, \varepsilon\}$*. Right: plot of* $\frac{1}{\ln 1/p_2}\min\{1, \frac{1}{\varepsilon}\}$ *(blue) versus* ε *on a logarithmic scale* ($x = \log_{10}\varepsilon$)*, with* $w = 2\max\{\frac{1}{\varepsilon}, \varepsilon\}$*.*

In the Euclidean space $\mathbb{R}^d$ we use the families of hash functions introduced by Datar et al. [13][3], which derive from so-called stable distributions, and more precisely from the normal distribution. Given a radius r and a choice of parameter $\varepsilon > 0$, we rescale the data points and the

[3] A possible improvement would be to use the hash functions defined by Andoni and Indyk [2], which give $\varrho \leq \frac{1}{(1+\varepsilon)^2}$ and could perhaps also improve our bound on α.

query point so that $r = 1$, we choose a real value $w > 0$, and we define a two-parameters family of hash functions $\mathcal{F} = \{f_{a,b} : \mathbb{R}^d \to \mathbb{Z}\}_{a\in\mathbb{R}^d, b\in[0,w)}$ by $f_{a,b}(x) = \lfloor \frac{x\cdot a+b}{w} \rfloor$, where $\cdot$ stands for the inner product in $\mathbb{R}^d$. The probability distribution put on the parameter space is not uniform: the coordinates of vector a are chosen independently according to the normal distribution $\mathcal{N}(0,1)$, while b is chosen uniformly at random from $[0, w)$.

The local sensitivity of this family depends on the choice of parameter w. More precisely, according to Datar et al. [13], the probability that two points at distance l of each other collide is $P(l) = \int_0^w \frac{1}{l} \mathrm{f}_{\mathcal{N}}(\frac{t}{l})(1 - \frac{t}{w})dt = 1 - 2F_{\mathcal{N}}(-w/l) - \frac{2}{\sqrt{2\pi}w/l}(1 - e^{-w^2/2l^2})$, where $\mathrm{f}_{\mathcal{N}}$ and $F_{\mathcal{N}}$ denote respectively the probability density function and the cumulative distribution function of the normal distribution $\mathcal{N}(0,1)$. The probability $P(l)$ decreases with l, so for $r_1 = r = 1$, $r_2 = 1+\varepsilon$, and $r_0 = \frac{1}{1+\varepsilon}$, we have $p_1 = P(1) = 1-2F_{\mathcal{N}}(-w) - \frac{2}{\sqrt{2\pi}w}(1-e^{-w^2/2})$, $p_2 = P(1+\varepsilon) = 1 - 2F_{\mathcal{N}}(-w/(1+\varepsilon)) - \frac{2}{\sqrt{2\pi}w/(1+\varepsilon)}(1 - e^{-w^2/2(1+\varepsilon)^2})$, and $p_0 = P(\frac{1}{1+\varepsilon}) = 1 - 2F_{\mathcal{N}}(-w(1+\varepsilon)) - \frac{2}{\sqrt{2\pi}w(1+\varepsilon)}(1 - e^{-w^2(1+\varepsilon)^2/2})$.

As claimed in [13] and illustrated in Figure 1 (left), the ratio $\varrho = \frac{\ln p_1}{\ln p_2}$ tends to $\frac{1}{1+\varepsilon}$ as parameter w tends to infinity, and for large enough values of w we have $\varrho \leq \frac{1}{1+\varepsilon}$. To be more precise, choosing $w \geq 2\max\{1, \varepsilon\}$ suffices to get $\varrho \leq \frac{1}{1+\varepsilon}$ for any $\varepsilon > 0$, as shown in Figure 1 (right). In the meantime, choosing $w \leq 2\max\{1, \varepsilon\}$ makes the dimensionality $k = \lfloor \frac{\ln n}{\ln 1/p_2} \rfloor$ of the vectors of hash functions smaller than $3 \ln n$, as illustrated in Figure 3 (left). Differently, the ratio $\frac{\alpha}{\varrho} = 1 - \frac{\ln p_0}{\ln p_1}$ exceeds its limit $\frac{\varepsilon}{1+\varepsilon}$ for arbitrarily large finite values of w — see Figure 2 (left). Choosing $w \geq \frac{2}{\varepsilon}$ enables us to bound $\frac{\alpha}{\varrho}$ from above by ε — see Figure 2 (right), at the price of an increased dimensionality k of the vectors of hash functions, which can exceed $2\max\{1, \frac{1}{\varepsilon}\}\ln n$ but stays below $3\max\{1, \frac{1}{\varepsilon}\}\ln n$ as long as $w \leq 2\max\{\frac{1}{\varepsilon}, \varepsilon\}$ — see Figure 3 (right). Thus, choosing $w = 2\max\{\frac{1}{\varepsilon}, \varepsilon\}$ guarantees that $\varrho \leq \frac{1}{1+\varepsilon}$ and $\alpha \leq \varepsilon\varrho \leq \frac{\varepsilon}{1+\varepsilon}$, while $k \leq \frac{3}{\min\{1,\varepsilon\}} \ln n = \tilde{O}(\frac{1}{\varepsilon})$. We deduce the following complexity bounds for our algorithm:

Theorem 4.4. *Given a finite set P with n points in the Euclidean space $\mathbb{R}^d$, for any choice of $\varepsilon > 0$ the algorithm answers exact $\mathcal{NN}$ queries correctly with high probability in expected $\tilde{O}(\frac{1}{\varepsilon}n^{\varrho} + \frac{1}{\varepsilon}n^{\alpha}|\varepsilon(2+\varepsilon)\text{-}\mathcal{NN}_P(q)|)$ time using $\tilde{O}(\frac{1}{\varepsilon^2}n^{2+\varrho})$ space, where $\varrho \leq \frac{1}{1+\varepsilon}$ and $\alpha \leq \varepsilon\varrho \leq \frac{\varepsilon}{1+\varepsilon}$.*

4.2 Hamming case

In the Hamming cube $\mathbb{H}^d$ we use projections along random coordinates as our hash functions, like in the original LSH paper [19]. Specifically, each hash function f in our family is the projection along one coordinate of the Hamming cube, chosen uniformly at random from the set of d coordinates. Given two points $x, y \in \mathbb{H}^d$, the probability that $f(x) = f(y)$ is then precisely $1 - \frac{\mathrm{d_H}(x,y)}{d}$. Given a radius r and a choice of parameter $\varepsilon > 0$, let us replace p_0, p_1 and p_2 by their respective values in the expressions of ϱ and α given in Theorem 3.2:

$$\begin{aligned} \varrho &= \tfrac{\ln(1-r/d)}{\ln(1-r(1+\varepsilon)/d)}, \\ \alpha &= \varrho\left(1 - \tfrac{\ln(1-r/d(1+\varepsilon))}{\ln(1-r/d)}\right) = \tfrac{\ln(1-r/d)}{\ln(1-r(1+\varepsilon)/d)}\left(1 - \tfrac{\ln(1-r/d(1+\varepsilon))}{\ln(1-r/d)}\right). \end{aligned} \tag{1}$$

These values depend on ε and on the ratio $\frac{r}{d}$. As illustrated in Figure 4, ϱ always remains below $\frac{1}{1+\varepsilon}$ as desired, but the ratio $\frac{\alpha}{\varrho}$ increases with $\frac{r}{d}$ until it reaches 1 as $\frac{r}{d} \to 1$. So, without any further restrictions on the ratio $\frac{r}{d}$, we cannot hope to reduce the output-sensitive term in the running time of the exact $\mathcal{NN}$ algorithm below $\tilde{O}(n^{\varrho})$ per point.

Since $\frac{\alpha}{\varrho}$ increases monotonically with $\frac{r}{d}$, with a limit equal to $\frac{\varepsilon}{1+\varepsilon}$ when $\frac{r}{d} \to 0$, we can make α smaller than $\varepsilon\varrho$ by forcing the ratio $\frac{r}{d}$ to be small. Specifically, a calculation with Maple shows

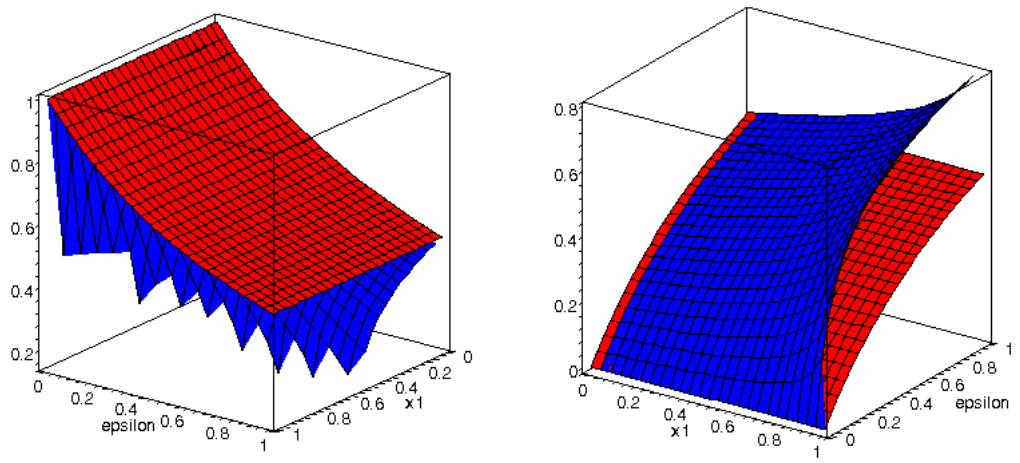

Figure 4: *Comparing* ϱ *and* $\frac{\alpha}{\varrho}$ *against* $\frac{1}{1+\varepsilon}$ *and* $\frac{\varepsilon}{1+\varepsilon}$ *respectively in the Hamming cube. Left: plots of* ϱ *(blue) and* $\frac{1}{1+\varepsilon}$ *(red) versus* ε *and* $x_1 = \frac{r}{d}$*. Right: plots of* $\frac{\alpha}{\varrho}$ *(blue) and* $\frac{\varepsilon}{1+\varepsilon}$ *(red) versus* ε *and* $x_1 = \frac{r}{d}$*.*

that $\alpha \leq \varepsilon\varrho$ as long as $\frac{r}{d}$ remains below ε. In order to make $\frac{r}{d}$ smaller than ε, we artificially embed the data and query points into a higher-dimensional Hamming space, of dimension $d' = \frac{d}{\varepsilon}$, by adding $(d' - d)$ coordinates equal to 0 to every point. This embedding is clearly isometric. Our new family of hash functions is made of the projections along random coordinates in $\mathbb{H}^{d'}$, with a uniform distribution over the coordinates. So, given two points $x, y \in \mathbb{H}^d$ and their images $x', y' \in \mathbb{H}^{d'}$ through the embedding, the probability that x, y get the same hash value is now $1 - \frac{\mathrm{d_H}(x',y')}{d'} = 1 - \frac{\mathrm{d_H}(x,y)}{d'} \geq 1 - \frac{d}{d'} = 1 - \varepsilon$.

Since the embedding is isometric, the set $\mathcal{B}_P(q, r)$ is the same in $\mathbb{H}^{d'}$ as in $\mathbb{H}^d$. This means that the solution set of the exhaustive r-$\mathcal{PLEB}$ query is the same, and that it is still equal to the output of Algorithm 2 (now applied in $\mathbb{H}^{d'}$) with high probability.

The effect of the embedding on ϱ and α is to replace $\frac{r}{d}$ by $\frac{r}{d'} = \varepsilon\frac{r}{d}$ in Eq. 1. As a result, ϱ and α remain below $\frac{1}{1+\varepsilon}$ and $\varepsilon\varrho$ respectively, as shown in Figure 5. In the meantime, the embedding influences the complexity of the algorithm in a more subtle way: the probabilities p_0, p_1 and p_2 are now changed, and in particular p_2 becomes $1 - \frac{r\varepsilon(1+\varepsilon)}{d}$, so at fixed r the algorithm takes vectors of hash functions of dimension $\lfloor \frac{\ln n}{-\ln p_2} \rfloor = \lfloor \frac{\ln n}{-\ln(1-r\varepsilon(1+\varepsilon)/d)} \rfloor \leq \lfloor \frac{1}{\varepsilon}\left(1 + \frac{\ln n}{-\ln(1-r(1+\varepsilon)/d)}\right) \rfloor \leq \frac{1}{\varepsilon}\left(2 + \lfloor \frac{\ln n}{-\ln(1-r(1+\varepsilon)/d)} \rfloor\right)$. As a consequence, the time and space complexities of step 2 of the query algorithm (the only step that requires the embedding) are multiplied by $O(\frac{1}{\varepsilon})$.

Finally, there is the matter of storing the data and query points after the embedding into $\mathbb{H}^{d'}$. First, note that the embedding is independent of the value of parameter r, so it can be computed once and for all before running the whole algorithm. Furthermore, the embedding can be made implicit by keeping the points in $\mathbb{H}^d$ and by adding the constant map $\mathbb{H}^d \to \{0\}$ to the family of hash functions. This new hash function stands for the projections onto coordinates higher than d, and as such it is assigned a probability equal to $\frac{d'-d}{d'} = 1 - \varepsilon$, while each of the d original coordinates is assigned a probability equal to $\frac{1}{d'} = \frac{\varepsilon}{d}$. This avoids incurring an extra $\frac{1}{\varepsilon}$ factor in the memory space used to store the data points.

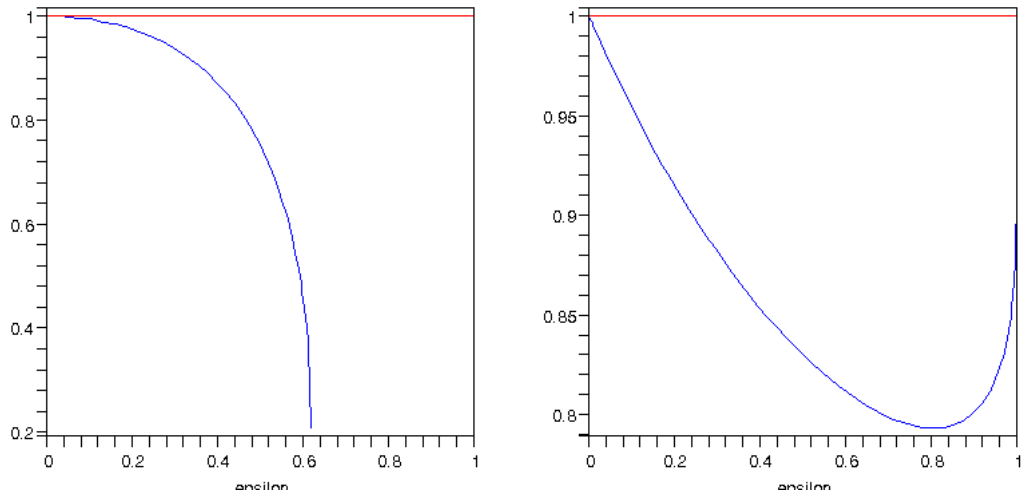

Figure 5: *Comparing ϱ and α against $\frac{1}{1+\varepsilon}$ and $\varepsilon\varrho$ respectively after the embedding into $\mathbb{H}^{d/\varepsilon}$. Left: plot of $\varrho(1+\varepsilon)$ (blue) versus ε, with $r = d$ (the highest possible value for r). The limits in $\varepsilon = 0$ and $\varepsilon = \frac{\sqrt{5}-1}{2}$ are equal to 1 and 0 respectively. Right: plot of $\frac{\alpha}{\varepsilon\varrho}$ (blue) versus ε, with $r = d$. The limits in $\varepsilon = 0$ and $\varepsilon = 1$ are both equal to 1.*

All in all, we obtain the following bounds on the complexity of the algorithm:

Theorem 4.5. *Given a finite set P with n points in the Hamming cube $\mathbb{H}^d$, for any choice of $\varepsilon > 0$ the algorithm answers exact $\mathcal{NN}$ queries correctly with high probability in expected $\tilde{O}(\frac{1}{\varepsilon}n^{\varrho} + \frac{1}{\varepsilon}n^{\alpha}|\varepsilon(2+\varepsilon)\text{-}\mathcal{NN}_P(q)|)$ time using $\tilde{O}(\frac{1}{\varepsilon^2}n^{2+\varrho})$ space, where $\varrho \leq \frac{1}{1+\varepsilon}$ and $\alpha \leq \varepsilon\varrho \leq \frac{\varepsilon}{1+\varepsilon}$.*

In practice a choice must be made between re-embedding the data in order to reduce α down to $\frac{\varepsilon}{1+\varepsilon}$, and not re-embedding the data in order to avoid incurring an extra global factor of $\frac{1}{\varepsilon}$ in the complexity of the algorithm.

5 From exhaustive r-$\mathcal{PLEB}$ to ε-$\mathcal{RNN}$

In this section we focus on our main problem (ε-$\mathcal{RNN}$) and show how it can be reduced to a single instance of ε-$\mathcal{NN}$ search plus a controlled number of instances of exhaustive r-$\mathcal{PLEB}$. The details of the reduction are given in Section 5.2, its output proven correct in Section 5.3, and its complexity analyzed in full generality in Section 5.4. In Section 5.5 we derive precise complexity bounds in the Euclidean space $\mathbb{R}^d$ and in the Hamming cube $\mathbb{H}^d$. For now we begin with an overview of the reduction and of its key ingredients in Section 5.1.

5.1 Overview of the reduction

We are given as input a cloud P of n points in some metric space (X, d). Suppose the distance of every point $p \in P$ to its nearest neighbor in P has been pre-computed. Then, given a query point q, computing a solution to the exact $\mathcal{RNN}$ query amounts to checking, for every point $p \in P$, whether $\mathrm{d}(q,p) \leq \mathrm{d}(p,P)$ or $\mathrm{d}(q,p) > \mathrm{d}(p,P)$: in the first case, p must be included in the solution, whereas in the second case it must not. This check for point p can be done by computing the solution S of the exaustive r-$\mathcal{PLEB}$ query on input (P,q), with $r = \mathrm{d}(p,P)$, and by including p in the answer if and only if it belongs to S. Indeed, we have

$$p \in \mathcal{RNN}_P(q) \Leftrightarrow \mathrm{d}(p,q) \leq \mathrm{d}(p,P) = r \Leftrightarrow p \in \mathcal{B}_P(q,r) \Leftrightarrow p \in S.$$

This observation was exploited in previous work [22] and lies at the core of approach as well. The main problem is that parameter r changes with every data point $p \in P$ considered, so the total number of exhaustive r-$\mathcal{PLEB}$ queries to be solved can be up to linear in n. To reduce this number, we turn our focus to the ε-approximate $\mathcal{RNN}$ problem and use a bucketing strategy. In a pre-processing phase, we compute $\mathrm{d}(p,P)$ for every point $p \in P$ and then we hash the data points into buckets according to their nearest neighbor distances, so that bucket P_i contains the points $p \in P$ such that $\mathrm{d}(p,P) \in \big[(1+\varepsilon)^{i-1}, (1+\varepsilon)^i\big)$. At query time, we solve an exhaustive r-$\mathcal{PLEB}$ query with $r = (1+\varepsilon)^i$ on each bucket P_i separately, and then we take the union of the solutions as our output. Since the points $p \in P_i$ satisfy $(1+\varepsilon)^{i-1} \leq \mathrm{d}(p,P) < (1+\varepsilon)^i$, it is easily seen that our output is an admissible solution to the ε-$\mathcal{RNN}$ query on q.

A remaining issue is that we do not impose any constraints on parameter i, so at query time we need to inspect every non-empty bucket P_i. As a result, in pathological cases such as when all non-empty buckets are singletons, we will end up considering a linear number of buckets, even though the set ε-$\mathcal{RNN}_P(q)$ itself might be empty. To avoid this pitfall, we limit the range of values of i to be considered thanks to the following observations, where y is an arbitrary point of ε-$\mathcal{NN}_P(q)$:

Observation 1. *Every point $p \in \mathcal{RNN}_P(q)$ satisfies $\mathrm{d}(p,P) \geq \frac{\mathrm{d}(q,y)}{2(1+\varepsilon)}$.*

Proof. Let $p \in P$ be such that $\mathrm{d}(p,P) < \frac{\mathrm{d}(q,y)}{2(1+\varepsilon)}$, and let $p' \in P \setminus \{p\}$ be closest to p. Then, $\mathrm{d}(p,p') = \mathrm{d}(p,P) < \frac{\mathrm{d}(q,y)}{2(1+\varepsilon)}$, which is at most $\frac{1}{2}\mathrm{d}(q,p')$ since $y \in \varepsilon$-$\mathcal{NN}_P(q)$. It follows that $\mathrm{d}(p,p') < \mathrm{d}(p,q)$, which means that $p \notin \mathcal{RNN}_P(q)$. □

Observation 2. *Every point $p \in \mathcal{RNN}_P(q)$ such that $\mathrm{d}(p,P) \geq \frac{2\mathrm{d}(q,y)}{\varepsilon}$ belongs to $\frac{\varepsilon}{2}$-$\mathcal{RNN}_P(y)$.*

Proof. Since $p \in \mathcal{RNN}_P(q)$, we have $\mathrm{d}(p,q) \leq \mathrm{d}(p,P)$. In addition, we have $\mathrm{d}(q,y) \leq \frac{\varepsilon}{2}\mathrm{d}(p,P)$ by hypothesis. It follows that $\mathrm{d}(p,y) \leq \mathrm{d}(p,q) + \mathrm{d}(q,y) \leq (1+\frac{\varepsilon}{2})\mathrm{d}(p,P)$, which means that p belongs to $\frac{\varepsilon}{2}$-$\mathcal{RNN}_P(y)$. □

Assuming that we have precomputed a data structure that enables us to find some $y \in \varepsilon$-$\mathcal{NN}_P(q)$, Observation 1 guarantees that we can safely ignore those buckets P_i with $i \leq \log_{1+\varepsilon} \frac{\mathrm{d}(q,y)}{2(1+\varepsilon)}$. Furthermore, assuming that the set $\frac{\varepsilon}{2}$-$\mathcal{RNN}_P(y)$ has been precomputed, Observation 2 guarantees that the reverse nearest neighbors of q that belong to those buckets P_i with $i \geq 1 + \log_{1+\varepsilon} \frac{2\mathrm{d}(q,y)}{\varepsilon}$ can simply be looked for among the points of $\frac{\varepsilon}{2}$-$\mathcal{RNN}_P(y)$. Thus, the total number of buckets to be inspected is reduced to $\lceil \log_{1+\varepsilon} \frac{2\mathrm{d}(q,y)}{\varepsilon} \rceil - \lfloor \log_{1+\varepsilon} \frac{\mathrm{d}(q,y)}{2(1+\varepsilon)} \rfloor \leq 2 + \log_{1+\varepsilon} \frac{4(1+\varepsilon)}{\varepsilon} = O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$.

5.2 Details of the reduction

Assuming that the ambient metric space (X,d) admits $(r, r(1+\varepsilon), p_1, p_2)$-sensitive families of hash functions for all values $r > 0$, our pre-computation phase builds a data structure $\mathcal{RNNDS}(P,\varepsilon)$ that stores the following pieces of information (see Algorithm 3):

- A collection of buckets $\{P_i\}_{i\in\mathbb{Z}}$ that partition P. Specifically, each bucket P_i contains those points $p \in P$ such that $(1+\varepsilon)^{i-1} \leq \mathrm{d}(p,P) < (1+\varepsilon)^i$. To compute the buckets, we iterate over the points $p \in P$, compute $\mathrm{d}(p,P)$ exactly either by brute-force or by using the algorithms of Section 4, and assign p to its corresponding bucket (lines 2-5). Once this is done, the empty buckets are discarded and the non-empty buckets are stored in a hash table to ensure constant look-up time. On each non-empty bucket P_i we build an $\mathcal{A}(P_i, (1+\varepsilon)^i, \varepsilon)$ data structure using Algorithm 1 (lines 6-8). Note that when applying Algorithm 1 we follow Remark 3.4 and increase the number of iterations of the main loop

from $\lceil c \ln |P_i| \rceil$ to $\lceil c \ln n \rceil$, where $c = \frac{3}{\ln \frac{5}{2}}$. This will guarantee the success of future queries with probability at least $1 - \frac{1}{n^2}$.

- For each point $y \in P$, the set $\frac{\varepsilon}{2}$-$\mathcal{RNN}_P(y)$ is stored in a variable P_y. These sets are easily computed in time $O(n^2)$ once $\mathrm{d}(p, P)$ has been computed for every point $p \in P$ at the first step of the construction (lines 9-12).
- The tree structure $\mathcal{T}(P, \varepsilon)$ of Section 2.2 and its associated (r, ε)-$\mathcal{PLEB}$ data structures (line 13).

Now, given the $\mathcal{RNNDS}(P, \varepsilon)$ data structure and a query point $q \in X$, we answer the ε-$\mathcal{RNN}$ query as follows (see Algorithm 4):

- We find an ε-nearest neighbor y of q among the points of P with high probability, using the procedure of Section 2.2 (line 1).
- We apply Algorithm 2 to answer an exhaustive $(1+\varepsilon)^i$-$\mathcal{PLEB}$ query on each bucket P_i separately, for i lying in the range prescribed by Observations 1 and 2 (lines 2-6), and then we gather the solutions into a set S (line 7). Note that we apply Algorithm 2 on P_i with an increased number of iterations of the main loop, specifically from $\lceil c \ln |P_i| \rceil$ to $\lceil c \ln n \rceil$, where $c = \frac{3}{\ln \frac{5}{2}}$, which raises the probability of success of the query from $1 - \frac{1}{|P_i|^2}$ (which could be as small as 0 if P_i turned out to be a singleton) to $1 - \frac{1}{n^2}$, as per Remark 3.4.
- We iterate over the points $p \in P_y$ that satisfy $\mathrm{d}(p, P) \geq \frac{2\mathrm{d}(q,y)}{\varepsilon}$, and we add to S the ones that are closer to q than to y (lines 8-12).

Upon termination, we return the set S (line 13).

Input : metric space (X, d), point cloud $P \subseteq X$, parameter $\varepsilon \geq 0$
Output: $\mathcal{RNNDS}(P, \varepsilon)$ data structure

1 Initialize $P_i := \emptyset$ for $i \in \mathbb{Z}$
2 **foreach** $p \in P$ **do**
3 | Compute $\mathrm{d}(p, P)$ exactly
4 | Find i s.t. $(1+\varepsilon)^{i-1} \leq \mathrm{d}(p, P) < (1+\varepsilon)^i$ and update $P_i := P_i \cup \{p\}$
5 **end**
6 **foreach** $P_i \neq \emptyset$ **do**
7 | Build an $\mathcal{A}(P_i, (1+\varepsilon)^i, \varepsilon)$ data structure
8 **end**
9 **foreach** $y \in P$ **do**
10 | Build the set $P_y := \frac{\varepsilon}{2}$-$\mathcal{RNN}_P(y)$
11 | Sort the points $p \in P_y$ by increasing distances $\mathrm{d}(p, P)$
12 **end**
13 Build the tree structure $\mathcal{T}(P, \varepsilon)$ of Section 2.2 and its associated (r, ε)-$\mathcal{PLEB}$ data structures

Algorithm 3: Pre-processing phase for ε-$\mathcal{RNN}$.

5.3 Correctness of the output

Corollary 2.4 guarantees that Line 1 of Algorithm 4 will retrieve a point $y \in \varepsilon$-$\mathcal{NN}_P(q)$ with high probability. Let us show that, given that $y \in \varepsilon$-$\mathcal{NN}_P(q)$, the set S output by the algorithm satisfies $\mathcal{RNN}_P(q) \subseteq S \subseteq \varepsilon$-$\mathcal{RNN}_P(q)$ with high probability.

For clarity, we let S_1 be the set of points inserted in S on lines 2-7 of Algorithm 4, and S_2 be the set of points inserted in S on lines 8-12. The output of the algorithm is $S_1 \cup S_2$. Our plan is to show that S_1 is correct with high probability (Lemma 5.1), then that S_2 is also correct

Input : metric space (X, d), $\mathcal{RNNDS}(P, \varepsilon)$ data structure, query point $q \in X$

1 Answer an ε-$\mathcal{NN}$ query on input (P, q), and let y be the output
2 **for** $i = \left\lfloor \log_{1+\varepsilon} \frac{\mathrm{d}(q,y)}{2(1+\varepsilon)} \right\rfloor + 1$ to $\left\lceil \log_{1+\varepsilon} \frac{2\mathrm{d}(q,y)}{\varepsilon} \right\rceil$ **do**
3 **if** $P_i \neq \emptyset$ **then**
4 Answer an exhaustive $(1+\varepsilon)^i$-$\mathcal{PLEB}$ query on input (P_i, q), and let S_i be the output
5 **end**
6 **end**
7 Let $S := \bigcup_i S_i$
8 **foreach** $p \in P_y$ such that $\mathrm{d}(p, P) \geq \frac{2\mathrm{d}(q,y)}{\varepsilon}$ **do**
9 **if** $\mathrm{d}(p, q) \leq \mathrm{d}(p, y)$ **then**
10 $S := S \cup \{p\}$
11 **end**
12 **end**
13 Return S

Algorithm 4: Online query phase for ε-$\mathcal{RNN}$.

with high probability (Lemma 5.2), from which we will deduce that the output $S_1 \cup S_2$ itself is correct with high probability (Theorem 5.3). Let $P_{S_1} = \bigcup_i P_i$ for $i = \lfloor \log_{1+\varepsilon} \frac{\mathrm{d}(q,y)}{2(1+\varepsilon)} \rfloor + 1$ to $\lceil \log_{1+\varepsilon} \frac{2\mathrm{d}(q,y)}{\varepsilon} \rceil$.

Lemma 5.1. *Given that $y \in \varepsilon$-$\mathcal{NN}_P(q)$, we have $\mathcal{RNN}_P(q) \cap P_{S_1} \subseteq S_1 \subseteq \varepsilon$-$\mathcal{RNN}_P(q)$ with high probability.*

Proof. Line 7 of Algorithm 4 builds S_1 by taking the union of the sets S_i generated by answering exhaustive $(1+\varepsilon)^i$-$\mathcal{PLEB}$ queries on the point sets P_i with query point q. Recall from line 4 of Algorithm 3 that every point $p \in P_i$ satisfies $(1+\varepsilon)^{i-1} \leq \mathrm{d}(p, P) < (1+\varepsilon)^i$. Recall also that the test on line 8 of Algorithm 2 ensures that every point $p \in S_i$ belongs to $\mathcal{B}_{P_i}(q, (1+\varepsilon)^i)$, which implies that $\mathrm{d}(p, q) \leq (1+\varepsilon)\mathrm{d}(p, P)$. Thus, $S_i \subseteq \varepsilon$-$\mathcal{RNN}_P(q)$. In addition, the points $p \in \mathcal{RNN}_P(q) \cap P_i$ satisfy $\mathrm{d}(p, q) \leq \mathrm{d}(p, P) \leq (1+\varepsilon)^i$ and therefore belong to $\mathcal{B}_{P_i}(q, (1+\varepsilon)^i)$. Since $\lceil c \ln n \rceil$ iterations of the main loops of Algorithms 1 and 2 are run, with $c = \frac{3}{\ln \frac{5}{2}}$, Lemma 3.3 and Remark 3.4 guarantee that $\mathcal{B}_{P_i}(q, (1+\varepsilon)^i)$ is included in S_i with probability at least $1 - \frac{1}{n^2}$. Hence, with the same probability we have $\mathcal{RNN}_P(q) \cap P_i \subseteq S_i$.

Since the above inclusions hold with probability $\geq 1 - \frac{1}{n^2}$ for each non-empty set P_i taken separately, and since the total number of non-empty sets P_i is at most n, we conclude by the union bound that $\mathcal{RNN}_P(q) \cap P_{S_1} \subseteq S_1 \subseteq \varepsilon$-$\mathcal{RNN}_P(q)$ with probability at least $1 - \frac{1}{n}$, where P_{S_1} is the union of the P_i and S_1 is the union of the S_i for i ranging from $\lfloor \log_{1+\varepsilon} \frac{\mathrm{d}(q,y)}{2(1+\varepsilon)} \rfloor + 1$ to $\lceil \log_{1+\varepsilon} \frac{2\mathrm{d}(q,y)}{\varepsilon} \rceil$. □

Lemma 5.2. *Given that $y \in \varepsilon$-$\mathcal{NN}_P(q)$, we have $\mathcal{RNN}_P(q) \setminus P_{S_1} \subseteq S_2 \subseteq \frac{\varepsilon}{2}$-$\mathcal{RNN}_P(q)$ with high probability.*

Proof. The first inclusion follows from Observations 1 and 2. Indeed, recall from line 4 of Algorithm 3 that every point $p \in P_i$ with $i < \lfloor \log_{1+\varepsilon} \frac{\mathrm{d}(q,y)}{2(1+\varepsilon)} \rfloor + 1$ satisfies $\mathrm{d}(p, P) < (1+\varepsilon)^i \leq \frac{\mathrm{d}(q,y)}{2(1+\varepsilon)}$ and therefore cannot belong to $\mathcal{RNN}_P(q)$, by Observation 1. In addition, the points $p \in \mathcal{RNN}_P(q) \cap P_i$ with $i > \lceil \log_{1+\varepsilon} \frac{2\mathrm{d}(q,y)}{\varepsilon} \rceil$ satisfy $\mathrm{d}(p, P) \geq (1+\varepsilon)^{i-1} \geq \frac{2\mathrm{d}(q,y)}{\varepsilon}$ and therefore

belong to $\frac{\varepsilon}{2}$-$\mathcal{RNN}_P(y)$, by Observation 2. Hence, all such points p are considered in the loop of lines 8-12 of Algorithm 4 and are therefore inserted in S. It follows that $\mathcal{RNN}_P(q) \setminus P_{S_1} \subseteq S_2$.

The second inclusion is straightforward: since by construction we have $S_2 \subseteq P_y = \frac{\varepsilon}{2}$-$\mathcal{RNN}_P(y)$, every point $p \in S_2$ satisfies $\mathrm{d}(p,q) \leq \mathrm{d}(p,y) \leq (1+\varepsilon/2)\mathrm{d}(p,P)$, which means that $p \in \frac{\varepsilon}{2}$-$\mathcal{RNN}_P(q)$. □

Our correctness guarantee follows directly from Lemmas 5.1 and 5.2, using the union bound:

Theorem 5.3. *Given a query point $q \in X$, Algorithm 4 outputs a set $S = S_1 \cup S_2$ such that $\mathcal{RNN}_P(q) \subseteq S \subseteq \varepsilon$-$\mathcal{RNN}_P(q)$ with high probability.*

5.4 Complexity

Our analysis will rely on the following important observation: considering an arbitrary set P_i, recall from line 4 of Algorithm 3 that the points $p \in P_i$ satisfy $\mathrm{d}(p,P) \geq (1+\varepsilon)^{i-1}$, so their pairwise distances are at least $(1+\varepsilon)^{i-1}$. As a result, when an exhaustive $(1+\varepsilon)^i$-$\mathcal{PLEB}$ query is solved on the set P_i for some input point q (line 4 of Algorithm 4), all the points of P_i except possibly one lie outside the ball $\mathcal{B}(q,(1+\varepsilon)^{i-1}/2)$. Therefore, by resorting to a family of hash functions that is $((1+\varepsilon)^{i-1}/2,(1+\varepsilon)^i,(1+\varepsilon)^{i+1},p_0,p_1,p_2)$-sensitive, we obtain an expected query time of the order of $\tilde{O}(n^\varrho + n^\alpha|\mathcal{B}_{P_i}(q,(1+\varepsilon)^{i+1})|)$, where $\varrho = \frac{\ln p_1}{\ln p_2}$ and $\alpha = \varrho(1-\frac{\ln p_0}{\ln p_1})$, by Theorem 3.2. Observe now that the points $p \in \mathcal{B}_{P_i}(q,(1+\varepsilon)^{i+1})$ satisfy $\mathrm{d}(p,q) \leq (1+\varepsilon)^{i+1} \leq (1+\varepsilon)^2\mathrm{d}(p,P)$, so we have $\mathcal{B}_{P_i}(q,(1+\varepsilon)^{i+1}) \subseteq \varepsilon(2+\varepsilon)$-$\mathcal{RNN}_P(q)$. Furthermore, since the buckets P_i are pairwise disjoint, so are the sets $\mathcal{B}_{P_i}(q,(1+\varepsilon)^{i+1})$, therefore the expected total time spent in the loop of lines 2-7 of Algorithm 4 is $\tilde{O}(\frac{1}{\varepsilon}n^\varrho + n^\alpha|\varepsilon(2+\varepsilon)$-$\mathcal{RNN}_P(q)|)$, the first term coming from the fact that there are $\tilde{O}(\frac{1}{\varepsilon})$ iterations of the loop, each one involving at most one point lying within distance $(1+\varepsilon)^{i-1}/2$ of q that may collide up to $\tilde{O}(n^\varrho)$ times with q in the hash tables.

The rest of the analysis is standard. Using families of hash functions that are $(r, r(1+\varepsilon), p_1, p_2)$-sensitive (probabilities p_1 and p_2 are the same as above), we perform the ε-$\mathcal{NN}$ query on line 1 of Algorithm 4 in $\tilde{O}(n^\varrho)$ time, by Corollary 2.4. In addition, since the points $p \in P_y$ have been sorted by increasing distances $\mathrm{d}(p,P)$ (line 11 of Algorithm 3), looking up for the subset of those points that satisfy $\mathrm{d}(p,P) \geq \frac{2\mathrm{d}(q,y)}{\varepsilon}$ takes $O(\log_2|P_y|) = O(\log_2 n)$ time. Now, for every such point p we have $\mathrm{d}(p,y) \geq \mathrm{d}(p,P) \geq \frac{2\mathrm{d}(q,y)}{\varepsilon}$, so $\mathrm{d}(p,q) \leq \mathrm{d}(p,y) + \mathrm{d}(y,q) \leq (1+\frac{\varepsilon}{2})\mathrm{d}(p,y) \leq (1+\frac{\varepsilon}{2})^2\mathrm{d}(p,P)$ since by construction $P_y = \frac{\varepsilon}{2}$-$\mathcal{RNN}_P(y)$. It follows that $p \in \varepsilon(1+\frac{\varepsilon}{4})$-$\mathcal{RNN}_P(q)$. Hence, the total number of points considered in the loop of lines 8-12 of Algorithm 4 is at most $|\varepsilon(1+\frac{\varepsilon}{4})$-$\mathcal{RNN}_P(q)|$. We thus obtain the following bound on the expected total query time[4]:

Theorem 5.4. *If the ambient space admits $(r/2(1+\varepsilon), r, r(1+\varepsilon), p_0, p_1, p_2)$-sensitive families of hash functions for all values $r > 0$, then the expected running time of Algorithm 4 is $\tilde{O}(\frac{1}{\varepsilon}n^\varrho + n^\alpha|\varepsilon(2+\varepsilon)$-$\mathcal{RNN}_P(q)| + |\varepsilon(1+\frac{\varepsilon}{4})$-$\mathcal{RNN}_P(q)|)$, where $\varrho = \frac{\ln p_1}{\ln p_2}$ and $\alpha = \varrho(1-\frac{\ln p_0}{\ln p_1})$. Else, if (X,d) only admits $(r, r(1+\varepsilon), p_1, p_2)$-sensitive families of hash functions for all values $r > 0$, then the expected running time of Algorithm 4 is $\tilde{O}(n^\varrho(\frac{1}{\varepsilon} + |\varepsilon(2+\varepsilon)$-$\mathcal{RNN}_P(q)|) + |\varepsilon(1+\frac{\varepsilon}{4})$-$\mathcal{RNN}_P(q)|)$.*

As mentioned in Section 5.2, the $\mathcal{RNNDS}(P,\varepsilon)$ data structure consists of a collection of pairwise-disjoint non-empty buckets, of total cardinality at most n, and for each bucket P_i an $\mathcal{A}(P_i,(1+\varepsilon)^i,\varepsilon)$ data structure of size $\tilde{O}(n_i^{1+\varrho})$ where $n_i = |P_i|$, by Theorem 3.2. This gives a

[4]Note that we do not need to use the same conditional expectation argument as in Lemma 4.2 here. The reason is because Algorithm 4 is designed so that the outcome of line 1 does not affect the expected running time of the rest of the query, even though it impacts the quality of the output.

total size of $\tilde{O}(\sum_i n_i^{1+\varrho}) = \tilde{O}(n^{1+\varrho})$. In addition, $\mathcal{RNNDS}(P,\varepsilon)$ stores the tree structure $\mathcal{T}(P,\varepsilon)$ and its associated (r,ε)-$\mathcal{PLEB}$ data structures, whose total size is $\tilde{O}(\frac{1}{\varepsilon}n^{1+\varrho})$ by Corollary 2.4. Finally, $\mathcal{RNNDS}(P,\varepsilon)$ stores the set P_y for each point $y \in P$, which takes a total size of $\tilde{O}(\sum_{y\in P}|P_y|)$. Combining these results, we obtain the following bound on the size of the data structure:

Theorem 5.5. *The size of the data structure $\mathcal{RNNDS}(P,\varepsilon)$ built by Algorithm 3 is $\tilde{O}(\frac{1}{\varepsilon}n^{1+\varrho} + \sum_{y\in P}|P_y|) = \tilde{O}(\frac{1}{\varepsilon}n^{1+\varrho} + n^2)$, where ϱ is defined as in Theorem 5.4.*

5.5 Special cases

When (X,d) is the Euclidean space $\mathbb{R}^d$ or the Hamming cube $\mathbb{H}^d$, we use the families of hash functions introduced in Sections 4.1 and 4.2 respectively. At no extra cost, the analysis detailed in these sections gives an upper bound on α that is slightly smaller than $\frac{1}{2}$, meaning that the cost incurred per point of the output set is of the order of $\sqrt{n}$, where n is the number of input points[5]. So our query time remains sublinear in n as long as the size of the output set remains less than $\sqrt{n}$ in order of magnitude.

In fact, we can obtain much better results with some extra work. More precisely, using an extra embedding of the data into higher dimension, we can reduce α down to ε. This new embedding is not isometric, so its impact on the behavior of the algorithm is a bit more subtle to analyze than in Section 4.2. Yet, it remains simple enough to be implemented in practice. The technical details of the analysis depend on the ambient space considered, but the spirit stays the same. Below we analyze two practical settings (Hamming and Euclidean) separately, for completeness.

Hamming case. Recall that Algorithm 4 uses the $\mathcal{A}(P_i, r_i, \varepsilon)$ data structures to answer exhaustive r_i-$\mathcal{PLEB}$ queries, where $r_i = (1+\varepsilon)^i$. As mentioned at the beginning of Section 5.4, all but one points of P_i lie at least $\frac{1}{2}(1+\varepsilon)^{i-1} = \frac{r_i}{2(1+\varepsilon)}$ away from the query point. Let us call p_i this one data point lying closer to q, if it exists.

Assume from now on that the data and query points lie in the Hamming cube $\mathbb{H}^d$. At preprocessing time, we embed each set P_i into $\mathbb{H}^{d'_i}$, with $d'_i = d + \frac{r_i}{2\varepsilon}$, by adding $d'_i - d$ coordinates equal to 0 to every point. We then build an $\mathcal{A}(P'_i, r'_i, \varepsilon')$ data structure, where P'_i denotes the image of P_i through the embedding, and where $r'_i = r_i(1+\frac{1}{2\varepsilon})$ and $\varepsilon' = \frac{\varepsilon^2}{1+\varepsilon}$. At query time, when an exhaustive r_i-$\mathcal{PLEB}$ query on the set P_i needs to be answered, we first embed the query point q into $\mathbb{H}^{d'_i}$ by adding $d'_i - d$ coordinates equal to 1, then we solve an exhaustive r'_i-$\mathcal{PLEB}$ query in $\mathbb{H}^{d'_i}$ using the $\mathcal{A}(P'_i, r'_i, \varepsilon')$ data structure.

Note that the embedding into $\mathbb{H}^{d'_i}$ is not isometric, since it does not preserve the distances of q to the data points. However, it increases these distances by the same amount $\frac{r_i}{2\varepsilon}$, therefore it preserves their order. We then have the following easy properties, where $x' \in \mathbb{H}^{d'_i}$ denotes the image of any point $x \in P_i \cup \{q\}$ through the embedding:

(i) $\forall p \in P_i$, $\mathrm{d_H}(p,q) \le r_i \Leftrightarrow \mathrm{d_H}(p',q') \le r_i(1+\frac{1}{2\varepsilon}) = r'_i$;

(ii) $\forall p \in P_i$, $\mathrm{d_H}(p',q') \le r'_i(1+\varepsilon') \Rightarrow \mathrm{d_H}(p,q) \le r'_i(1+\varepsilon') - \frac{r_i}{2\varepsilon} = r_i(1+\frac{1}{2\varepsilon})(1+\frac{\varepsilon^2}{1+\varepsilon}) - \frac{r_i}{2\varepsilon} = r_i(1+\frac{\varepsilon}{2(1+\varepsilon)} + \frac{\varepsilon^2}{1+\varepsilon}) \le r_i(1+\varepsilon)$;

(iii) $\forall p \in P_i \setminus \{p_i\}$, $\mathrm{d_H}(p',q') \ge \frac{r_i}{2(1+\varepsilon)} + \frac{r_i}{2\varepsilon} = \frac{r_i}{1+\varepsilon}(\frac{1}{2} + \frac{1+\varepsilon}{2\varepsilon}) = \frac{r_i}{1+\varepsilon}(1+\frac{1}{2\varepsilon}) = \frac{r'_i}{1+\varepsilon}$.

[5]The reason why α is now of the order of $\frac{1}{2}$ and no longer of ε is that, for every exhaustive r-$\mathcal{PLEB}$ query considered, the data points now only lie $\frac{r}{2(1+\varepsilon)}$ away from the query point, whereas they used to lie $\frac{r}{1+\varepsilon}$ away from it in Section 4.

It follows from (i) that $\mathcal{B}_{P'_i}(q', r'_i)$ is the image of $\mathcal{B}_{P_i}(q, r_i)$ through the embedding. Hence, by Lemma 3.3, with high probability the output set of our exhaustive r'_i-$\mathcal{PLEB}$ query in $\mathbb{H}^{d'_i}$ is the image, through the embedding, of the output set of the original exhaustive r_i-$\mathcal{PLEB}$ query in $\mathbb{H}^d$. Plugging this fact into the analysis of Section 5.3 (more precisely, into the proof of Lemma 5.1), we conclude that the output of Algorithm 4 is correct with high probability.

Let us now evaluate ϱ and α. By definition, we have $p_1 = 1 - \frac{r'_i}{d'_i}$, $p_2 = 1 - \frac{r'_i}{d'_i}(1+\varepsilon') = 1 - \frac{r'_i}{d'_i}(1 + \frac{\varepsilon^2}{1+\varepsilon})$, and (iii) gives $p_0 = 1 - \frac{r'_i}{d'_i(1+\varepsilon)}$. Hence,

$$\begin{aligned} \varrho &= \tfrac{\ln(1-r'_i/d'_i)}{\ln(1-r'_i(1+\varepsilon^2/(1+\varepsilon))/d'_i)} \leq \tfrac{1+\varepsilon}{1+\varepsilon+\varepsilon^2}, \\ \alpha &= \varrho\left(1 - \tfrac{\ln(1-r'_i/d'_i(1+\varepsilon))}{\ln(1-r'_i/d'_i)}\right). \end{aligned} \tag{2}$$

Note that $\varrho < 1$ whenever $\varepsilon > 0$, as in the previous sections. An easy computation shows that $\varrho \leq \frac{1}{1+\varepsilon^2/2}$ when $\varepsilon \leq 1$ and $\varrho \leq \frac{1}{1+\varepsilon/2}$ otherwise. Observe now that the ratio $\frac{\alpha}{\varrho}$ has the same expression as in Eq. (1), with $\frac{r}{d}$ replaced by $\frac{r'_i}{d'_i}$. It follows, by the same calculation as in Section 4.2, that we have $\alpha \leq \varepsilon\varrho$ whenever $\frac{r'_i}{d'_i} \leq \varepsilon$. Hence, following Section 4.2, we perform an extra isometric embedding of the data and query points into $\mathbb{H}^{d''_i}$, where $d''_i = \frac{d'_i}{\varepsilon}$, in order to ensure that $\alpha \leq \varepsilon\varrho$.

Let us point out that the embeddings into $\mathbb{H}^{d'_i}$ and $\mathbb{H}^{d''_i}$ are encoded implicitly in practice, using the trick of Section 4.2. The embedding into $\mathbb{H}^{d'_i}$ increases the number of dimensions in the vectors of hash functions as follows:

$$\left\lfloor \frac{\ln n}{-\ln\left(1 - \frac{r'_i}{d'_i}(1+\frac{\varepsilon^2}{1+\varepsilon})\right)} \right\rfloor \leq \left\lfloor \frac{\ln n}{-\ln\left(1 - \frac{r_i}{d}(1+\frac{\varepsilon^2}{1+\varepsilon})\right)} \right\rfloor \leq \left\lfloor \frac{(1+\varepsilon)^2}{1+\varepsilon+\varepsilon^2} \frac{\ln n}{-\ln\left(1 - \frac{r_i}{d}(1+\varepsilon)\right)} - \frac{1}{\ln \frac{\varepsilon}{(1+\varepsilon)^2}} \right\rfloor,$$

which is bounded by $\left\lfloor 1 + 2\frac{\ln n}{-\ln\left(1-\frac{r_i}{d}(1+\varepsilon)\right)} \right\rfloor$. Thus, the embedding into $\mathbb{H}^{d'_i}$ only increases the number of dimensions by a constant factor. Differently, as we saw in Section 4.2, the embedding into $\mathbb{H}^{d''_i}$ yields an extra $\frac{1}{\varepsilon}$ factor.

Finally, the running times of the exhaustive r'_i-$\mathcal{PLEB}$ queries are influenced by the sizes of the sets $\mathcal{B}_{P'_i}(q', r'_i(1+\varepsilon'))$. We know from (ii) that the points of P'_i lying within distance $r'_i(1+\varepsilon')$ of q' are images of points of $\mathcal{B}_{P_i}(q, r_i(1+\varepsilon))$ through the embedding. Therefore, we have $|\mathcal{B}_{P'_i}(q', r'_i(1+\varepsilon'))| \leq |\mathcal{B}_{P_i}(q, r_i(1+\varepsilon))| = |\mathcal{B}_{P_i}(q, (1+\varepsilon)^{i+1})|$, and so the expected running time of an exhaustive r'_i-$\mathcal{PLEB}$ query (after embedding the data and query points into $\mathbb{H}^{d''_i}$) is $\tilde{O}(\frac{1}{\varepsilon}n^\varrho + \frac{1}{\varepsilon}n^\alpha|\mathcal{B}_{P_i}(q, (1+\varepsilon)^{i+1})|)$. Plugging this bound into the analysis of Section 5.4 (which remains unchanged), we obtain the following complexity bounds for Algorithm 4:

Theorem 5.6. *Given a finite set P with n points in the Hamming cube $\mathbb{H}^d$, and a parameter $\varepsilon > 0$, our procedure answers ε-$\mathcal{RNN}$ queries correctly with high probability in expected $\tilde{O}(\frac{1}{\varepsilon^2}n^\varrho + \frac{1}{\varepsilon}n^\alpha|\varepsilon(2+\varepsilon)\text{-}\mathcal{RNN}_P(q)| + |\varepsilon(1+\frac{\varepsilon}{4})\text{-}\mathcal{RNN}_P(q)|)$ time, using $\tilde{O}(\frac{1}{\varepsilon}n^{1+\varrho} + \sum_{y\in P}|P_y|) = \tilde{O}(\frac{1}{\varepsilon}n^{1+\varrho} + n^2)$ space, where $\varrho \leq \max\{\frac{1}{1+\varepsilon/2}, \frac{1}{1+\varepsilon^2/2}\}$ and $\alpha \leq \varepsilon\varrho \leq \varepsilon$.*

Euclidean case. We apply the same kind of non-isometric embedding as in the Hamming case. If $\mathbb{R}^d$ were endowed with the l_1-distance, then the embedding would be exactly the same. However, here we endow $\mathbb{R}^d$ with the l_2-distance, so the details differ, and we recall them for completeness.

At preprocessing time, we embed each set P_i into a higher-dimensional Euclidean space $\mathbb{R}^{d'_i}$, where $d'_i = d + \frac{r_i}{2}\sqrt{\frac{3}{\varepsilon(2+\varepsilon)}}$, by adding $d'_i - d$ coordinates equal to 0 to every point. We then build an $\mathcal{A}(P'_i, r'_i, \varepsilon')$ data structure, where P'_i denotes the image of P_i through the embedding, and where $r'_i = r_i\sqrt{1 + \frac{3}{4\varepsilon(2+\varepsilon)}}$ and $\varepsilon' = \frac{\varepsilon^2}{1+\varepsilon}$. At query time, when an exhaustive r_i-$\mathcal{PLEB}$ query on the set P_i needs to be answered, we first embed the query point q into $\mathbb{R}^{d'_i}$ by adding $d'_i - d$ coordinates equal to 1, then we solve an exhaustive r'_i-$\mathcal{PLEB}$ query in $\mathbb{R}^{d'_i}$ using the $\mathcal{A}(P'_i, r'_i, \varepsilon')$ data structure.

Here again, the embedding into $\mathbb{R}^{d'_i}$ is not isometric since it does not preserve the distances of q to the data points. Yet, it preserves their order since the map $r \mapsto \sqrt{r^2 + \frac{3r_i^2}{4\varepsilon(2+\varepsilon)}}$ increases monotonically with r. We then have the following easy properties, where $x' \in \mathbb{R}^{d'_i}$ denotes the image of any point $x \in P_i \cup \{q\}$ through the embedding:

(i) $\forall p \in P_i$, $\mathrm{d_H}(p,q) \leq r_i \Leftrightarrow \mathrm{d_H}(p',q') \leq r_i\sqrt{1 + \frac{3}{4\varepsilon(2+\varepsilon)}} = r'_i$;

(ii) $\forall p \in P_i$, $\mathrm{d_H}(p',q') \leq r'_i(1+\varepsilon') \Rightarrow \mathrm{d_H}(p,q) \leq \sqrt{{r'_i}^2(1+\varepsilon')^2 - \frac{3r_i^2}{4\varepsilon(2+\varepsilon)}} = r_i\sqrt{(1 + \frac{3}{4\varepsilon(2+\varepsilon)})(1+\frac{\varepsilon^2}{1+\varepsilon})^2 - \frac{3}{4\varepsilon(2+\varepsilon)}} \leq r_i(1+\varepsilon)$;

(iii) $\forall p \in P_i \setminus \{p_i\}$, $\mathrm{d_H}(p',q') \geq r_i\sqrt{\frac{1}{4(1+\varepsilon)^2} + \frac{3}{4\varepsilon(2+\varepsilon)}} = \frac{r_i}{1+\varepsilon}\sqrt{\frac{8\varepsilon+4\varepsilon^2+3}{4\varepsilon(2+\varepsilon)}} = \frac{r_i}{1+\varepsilon}\sqrt{1 + \frac{3}{4\varepsilon(2+\varepsilon)}} = \frac{r'_i}{1+\varepsilon}$.

It follows from (i) that $\mathcal{B}_{P'_i}(q', r'_i)$ is the image of $\mathcal{B}_{P_i}(q, r_i)$ through the embedding. Hence, by Lemma 3.3, with high probability the output set of our exhaustive r'_i-$\mathcal{PLEB}$ query in $\mathbb{R}^{d'_i}$ is the image, through the embedding, of the output set of the original exhaustive r_i-$\mathcal{PLEB}$ query in $\mathbb{R}^d$. Plugging this fact into the analysis of Section 5.3 (more precisely, into the proof of Lemma 5.1), we conclude that the output of Algorithm 4 is correct with high probability.

Let us point out that the embedding into $\mathbb{R}^{d'_i}$ can be encoded implicitly in practice, as in the previous sections, but this time the encoding requires somewhat more care. To be more specific, we only keep the original d coordinates of each data and query point, as before, but for each hash function $f_{a',b} : \mathbb{R}^{d'_i} \to \mathbb{R}$ we store the first d coordinates of vector a' together with the sum of its remaining coordinates: $\sum_{j=d+1}^{d'_i} a'_j$. Then, for every data point $p \in P$ we have $p' \cdot a' = p \cdot a$, while for every query point $q \in \mathbb{R}^d$ we have $q' \cdot a' = q \cdot a + \sum_{j=d+1}^{d'_i} a'_j$, where a denotes the restriction of a' to the subspace $\mathbb{R}^d$ spanned by the first d coordinates. So, storing the data and query points does not cost more than in $\mathbb{R}^d$, and the cost of storing or evaluating one hash function is also of the same order of magnitude as in $\mathbb{R}^d$.

Let us now evaluate ϱ and α. By rescaling the point cloud P'_i so that $r'_i = 1$ as in Section 4.1, we get $p_1 = P(1)$, $p_2 = P(1+\varepsilon') = P(1 + \frac{\varepsilon^2}{1+\varepsilon})$, and (iii) gives $p_0 = P(\frac{1}{1+\varepsilon})$, where by definition $P(l) = \int_0^w \frac{1}{l}\mathrm{f}_{\mathcal{N}}(\frac{t}{l})(1 - \frac{t}{w})dt = 1 - 2F_{\mathcal{N}}(-w/l) - \frac{2}{\sqrt{2\pi}w/l}(1 - e^{-w^2/2l^2})$. Observe that p_0 and p_1 are the same as in Section 4.1, while p_2 is the same with ε replaced by $\frac{\varepsilon^2}{1+\varepsilon}$. It follows therefore from the analysis of Section 4.1 that, by choosing $w = 2\max\{\frac{1}{\varepsilon}, 1, \frac{\varepsilon^2}{1+\varepsilon}\}$, we ensure that $\varrho \leq \frac{1}{1+\varepsilon^2/(1+\varepsilon)}$ and $\alpha \leq \varepsilon\varrho$, at the price of an increase of the dimensionality of the vectors of hash functions, which is now $O(\frac{1}{\varepsilon}\ln n) = \tilde{O}(\frac{1}{\varepsilon})$.

Finally, the running times of the exhaustive r'_i-$\mathcal{PLEB}$ queries are influenced by the sizes of the sets $\mathcal{B}_{P'_i}(q', r'_i(1+\varepsilon'))$. We know from (ii) that the points of P'_i lying within distance $r'_i(1+\varepsilon')$ of q' are images of points of $\mathcal{B}_{P_i}(q, r_i(1+\varepsilon))$ through the embedding. Therefore, we

have $|\mathcal{B}_{P'_i}(q', r'_i(1+\varepsilon'))| \leq |\mathcal{B}_{P_i}(q, r_i(1+\varepsilon))| = |\mathcal{B}_{P_i}(q, (1+\varepsilon)^{i+1})|$, and so the expected running time of an exhaustive r'_i-$\mathcal{PLEB}$ query is $\tilde{O}(\frac{1}{\varepsilon}n^{\varrho} + \frac{1}{\varepsilon}n^{\alpha}|\mathcal{B}_{P_i}(q, (1+\varepsilon)^{i+1})|)$. Plugging this bound into the analysis of Section 5.4 (which remains unchanged), we obtain the following complexity bounds for Algorithm 4:

Theorem 5.7. *Given a finite set P with n points in the Euclidean space $\mathbb{R}^d$, and a parameter $\varepsilon > 0$, our procedure answers ε-$\mathcal{RNN}$ queries correctly with high probability in expected $\tilde{O}(\frac{1}{\varepsilon^2}n^{\varrho} + \frac{1}{\varepsilon}n^{\alpha}|\varepsilon(2+\varepsilon)\text{-}\mathcal{RNN}_P(q)| + |\varepsilon(1+\frac{\varepsilon}{4})\text{-}\mathcal{RNN}_P(q)|)$ time, using $\tilde{O}(\frac{1}{\varepsilon}n^{1+\varrho} + \sum_{y\in P}|P_y|) = \tilde{O}(\frac{1}{\varepsilon}n^{1+\varrho} + n^2)$ space, where $\varrho \leq \frac{1}{1+\varepsilon^2/(1+\varepsilon)}$ and $\alpha \leq \varepsilon\varrho \leq \varepsilon$.*

6 Open questions

• Can we prove our complexity bounds to be tight or close to being tight?
• In the bichromatic version of the $\mathcal{RNN}$ problem we no longer have that the points of P_i lie at least $\frac{r_i}{2(1+\varepsilon_i)}$ away from the query point. In fact, they may lie very close to one another and to q. As a result, (r_0, r_1, r_2)-sensitive families of hash functions cannot be used directly, and we once again have a factor of n^{ϱ} in the output-sensitive term of the complexity bound. Can a clever partitioning of the data points reduce α down to $O(\varepsilon)$? If so, at what cost?

Acknowledgements

A preliminary version of the paper was written in collaboration with Aneesh Sharma, and further experiments were undertaken by Maxime Brénon. The authors wish to acknowledge their respective contributions to this work.

References

[1] Elke Achtert, Christian Böhm, Peer Kröger, Peter Kunath, Alexey Pryakhin, and Matthias Renz. Efficient Reverse k-Nearest Neighbor Search in Arbitrary Metric Spaces. *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pages 515–526, 2006. 4

[2] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 459–468, 2006. doi: http://dx.doi.org/10.1109/FOCS.2006.49. 3, 14

[3] Alexandr Andoni and Piotr Indyk. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *Communications of the ACM*, 51(1):117, 2008. 3

[4] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998. 3

[5] Rimantas Benetis, Christian Jensen, Gytis Karĉiauskas, and Simonas Ŝaltenis. Nearest and Reverse Nearest Neighbor Queries for Moving Objects. *The International Journal on Very Large Data Bases*, 15(3):229–249, 2006. 4

[6] Jean-Daniel Boissonnat, Leonidas J. Guibas, and Steve Y. Oudot. Manifold Reconstruction in Arbitrary Dimensions using Witness Complexes. *Discrete and Computational Geometry*, 42(1):37–70, 2009. 4

[7] Sergio Cabello, José Miguel Díaz-Báñez, Stefan Langerman, Carlos Seara, and Inma Ventura. Facility Location Problems in the Plane Based on Reverse Nearest Neighbor Queries. *European Journal of Operational Research*, 2009. 4

[8] Otfried Cheong, Antoine Vigneron, and Juyoung Yon. Reverse nearest neighbor queries in fixed dimension. *International Journal of Computational Geometry and Applications*. URL `http://arxiv.org/abs/0905.4441v2`. 4

[9] Kenneth L. Clarkson. An Algorithm for Approximate Closest-point Queries. *Proceedings of the Tenth Annual Symposium on Computational Geometry*, pages 160–164, 1994. 3

[10] Kenneth L. Clarkson. Nearest neighbor queries in metric spaces. *Discrete and Computational Geometry*, 22:63–93, 1999. 3

[11] Kenneth L. Clarkson. Nearest neighbor searching in metric spaces: Experimental results for $sb(s)$. Preliminary version presented at ALENEX99, 2003. URL `http://www.almaden.ibm.com/u/kclarkson/Msb/readme.html`. 4

[12] Kenneth L. Clarkson. Nearest-neighbor searching and metric space dimensions. In Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk, editors, *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, pages 15–59. MIT Press, 2006. ISBN 0-262-19547-X. 3

[13] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. *SCG '04: Proceedings of the Twentieth Annual Symposium on Computational Geometry*, pages 253–262, 2004. doi: http://doi.acm.org/10.1145/997817.997857. 3, 6, 14, 15

[14] Karina Figueroa and Rodrigo Paredes. Approximate direct and reverse nearest neighbor queries, and the k-nearest neighbor graph. *Proceedings of the 2nd International Workshop on Similarity Search and Applications (SISAP 2009)*, 2009. 4

[15] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity Search in High Dimensions via Hashing. *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 518–529, 1999. 6, 7

[16] Leonidas J. Guibas and Steve Y. Oudot. Reconstruction using Witness Complexes. *Discrete and Computational Geometry*, 40(3):325–356, 2009. 4

[17] Sariel Har-Peled. A Replacement for Voronoi Diagrams of Near Linear Size. *Annual Symposium on Foundations of Computer Science*, 42:94–105, 2001. URL `http://valis.cs.uiuc.edu/~sariel/research/papers/01/avoronoi/avoronoi.pdf`. 6

[18] Piotr Indyk. Nearest Neighbors in High-dimensional Spaces. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 877–892. CRC Press, 2004. 3, 7

[19] Piotr Indyk and Rajeev Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 604–613, 1998. 3, 6, 7, 15

[20] James M. Kang, Mohamed F. Mokbel, Shashi Shekhar, Tian Xia, and Donghui Zhang. Continuous Evaluation of Monochromatic and Bichromatic Reverse Nearest Neighbors. *Proceedings of the IEEE 23rd International Conference on Data Engineering*, 2007. 4

[21] Jon M. Kleinberg. Two Algorithms for Nearest-Neighbor Search in High Dimensions. *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 599–608, 1997. 3

[22] Flip Korn and S. Muthukrishnan. Influence Sets Based on Reverse Nearest Neighbor Queries. *ACM SIGMOD Record*, 29(2):201–212, 2000. 4, 18

[23] Yokesh Kumar, Ravi Janardan, and Prosenjit Gupta. Efficient Algorithms for Reverse Proximity Query Problems. *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2008. 4

[24] Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient Search for Approximate Nearest Neighbor in High Dimensional Spaces. *SIAM Journal on Computing*, 30(2):457–474, 2000. 3

[25] Anil Maheshwari, Jan Vahrenhold, and Norbert Zeh. On Reverse Nearest Neighbor Queries. *Proceedings of Canadian Conference on Computational Geometry*, pages 128–132, 2002. 4

[26] Florian Pfender and Günter M. Ziegler. Kissing Numbers, Sphere Packings, and Some Unexpected Proofs. *Notices-American Mathematical Society*, 51:873–883, 2004. 4

[27] G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest-neighbor Methods in Learning and Vision: Theory and Practice*. MIT Press, 2005. 3, 4, 8

[28] Amit Singh, Hakan Ferhatosmanoglu, and Ali Şaman Tosun. High Dimensional Reverse Nearest Neighbor Queries. *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, pages 91–98, 2003. 4

[29] Ioana Stanoi, Divyakant Agrawal, and Amr El Abbadi. Reverse Nearest Neighbor Queries for Dynamic Databases. *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 44–53, 2000. 4

[30] Yufei Tao, Dimitris Papadias, and Xiang Lian. Reverse k-NN Search in Arbitrary Dimensionality. *Proceedings of the Thirtieth international Conference on Very Large Databases*, 30:744–755, 2004. 4

[31] Yufei Tao, Man Lung Yiu, and Nikos Mamoulis. Reverse Nearest Neighbor Search in Metric Spaces. *IEEE Transactions on Knowledge and Data Engineering*, pages 1239–1252, 2006. 4

[32] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB '98: Proceedings of the 24rd International Conference on Very Large Data Bases*, pages 194–205, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1-55860-566-5. 3

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 Orsay Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399