Robust Proximity Search for Balls Using Sublinear Space*

Sariel Har-Peled[1] and Nirman Kumar[2]

1 **Department of Computer Science**
University of Illinois
sariel@uiuc.edu
2 **Department of Computer Science**
University of Illinois
nkumar5@uiuc.edu

Abstract

Given a set of n disjoint balls $b_1, \ldots, b_n$ in $\mathbb{R}^d$, we provide a data structure, of near linear size, that can answer $(1 \pm \varepsilon)$-approximate kth-nearest neighbor queries in $O(\log n + 1/\varepsilon^d)$ time, where k and ε are provided at query time. If k and ε are provided in advance, we provide a data structure to answer such queries, that requires (roughly) $O(n/k)$ space; that is, the data structure has sublinear space requirement if k is sufficiently large.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Approximate Nearest neighbors, algorithms, data structures

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2014.315

1 Introduction

The *nearest neighbor* problem is a fundamental problem in Computer Science [18, 1]. Here, one is given a set of points P, and given a query point q one needs to output the nearest point in P to q. There is a trivial $O(n)$ algorithm for this problem. Typically the set of data points is fixed, while different queries keep arriving. Thus, one can use preprocessing to facilitate a faster query. There are several applications of nearest neighbor search in computer science including pattern recognition, information retrieval, vector compression, computational statistics, clustering, data mining and learning among many others, see for instance the survey by Clarkson [10] for references. If one is interested in guaranteed performance and near linear space, there is no known way to solve this problem efficiently (i. e., logarithmic query time) for dimension $d > 2$, while using near linear space for the data structure.

In light of the above, major effort has been devoted to develop approximation algorithms for nearest neighbor search [6, 17, 10, 13]. In the $(1 + \varepsilon)$-*approximate nearest neighbor* problem, one is additionally given an approximation parameter $\varepsilon > 0$ and one is required to find a point $\mathsf{u} \in \mathsf{P}$ such that $\mathsf{d}(\mathsf{q}, \mathsf{u}) \leq (1 + \varepsilon)\mathsf{d}(\mathsf{q}, \mathsf{P})$. In d dimensional Euclidean space, one can answer ANN queries in $O(\log n + 1/\varepsilon^{d-1})$ time using linear space [6, 12]. Unfortunately, the constant hidden in the O notation is exponential in the dimension (and this is true for all bounds mentioned in this paper), and specifically because of the $1/\varepsilon^{d-1}$ in the query time, this approach is only efficient in low dimensions. Interestingly, for this data structure,

* Work on this paper was partially support by NSF AF awards CCF-0915984 and CCF-1217462.

34th Int'l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2014).
Editors: Venkatesh Raman and S. P. Suresh; pp. 315–326

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

the approximation parameter ε need not be specified during the construction, and one can provide it during the query. An alternative approach is to use Approximate Voronoi Diagrams (AVD), introduced by Har-Peled [11], which is a partition of space into regions of low total complexity, with a representative point for each region, that is an ANN for any point in the region. In particular, Har-Peled showed that there is such a decomposition of size $O\big((n/\varepsilon^d)\log^2 n\big)$, see also [13]. This allows ANN queries to be answered in $O(\log n)$ time. Arya and Malamatos [2] showed how to build AVDs of linear complexity (i. e., $O(n/\varepsilon^d)$). Their construction uses WSPD (Well Separated Pairs Decomposition) [8]. Further trade-offs between query time and space usage for AVDs were studied by Arya *et al.* [4].

A more general problem is the k-nearest neighbors problem where one is interested in finding the k points in P nearest to the query point q. This is widely used in classification, where the majority label is used to label the query point. A restricted version is to find only the kth-nearest neighbor. This problem and its approximate version have been considered in [3, 14].

Recently, the authors [14] showed that one can compute a (k, ε)-AVD that $(1 + \varepsilon)$-approximates the distance to the kth nearest neighbor, and surprisingly, requires $O(n/k)$ space; that is, sublinear space if k is sufficiently large. For example, for the case $k = \Omega(\sqrt{n})$, which is of interest in practice, the space required is only $O(\sqrt{n})$. Such ANN is of interest when one is worried that there is noise in the data, and thus one is interested in the distance to the kth NN which is more robust and noise resistant. Alternatively, one can think about such data structures as enabling one to summarize the data in a way that still facilitates meaningful proximity queries.

In this paper we consider a generalization of the kth-nearest neighbor problem. Here, we are given a set of n disjoint balls in $\mathbb{R}^d$ and we want to preprocess them, so that given a query point we can find approximately the kth closest ball. The distance of a query point to a ball is defined as the distance to its boundary if the point is outside the ball or 0 otherwise. Clearly, this problem is a generalization of the kth-nearest neighbor problem by viewing points as balls of radius 0. Algorithms for the kth-nearest neighbor for points, do not extend in a straightforward manner to this problem because the distance function is no longer a metric. Indeed, there can be two very far off points both very close to a single ball, and thus the triangle inequality does not hold. The problem of finding the closest ball can also be modeled as a problem of approximating the minimization diagram of a set of functions; here, a function would correspond to the distance from one of the given balls. There has been some recent work by the authors on this topic, see [15], where a fairly general class of functions admits a near-linear sized data structure permitting a logarithmic time query for the problem of approximating the minimization diagram. However, the problem that we consider in this paper does not fall under the framework of [15]. The technical assumptions of [15] mandate that the set of points which form the 0-sublevel set of a distance function, i. e., the set of points at which the distance function is 0 is a single point (or an empty set). This is not the case for the problem we consider here. Also, we are interested in the more general kth-nearest neighbor problem, while [15] only considers the nearest-neighbor problem, i. e., $k = 1$.

We first show how to preprocess the set of balls into a data structure requiring space $O(n)$, in $O(n \log n)$ time, so that given a query point q, a number $1 \leq k \leq n$ and $\varepsilon > 0$, one can compute a $(1 \pm \varepsilon)$-approximate kth closest ball in time $O(\log n + \varepsilon^{-d})$. If both k and ε are available during preprocessing, one can preprocess the balls into a (k, ε)-AVD, using $O(\frac{n}{k\varepsilon^d}\log(1/\varepsilon))$ space, so that given a query point q, a (k, ε)-ANN closest ball can be computed, in $O(\log(n/k) + \log(1/\varepsilon))$ time.

Paper Organization. In Section 2, we define the problem, list some assumptions, and introduce notations. In Section 3, we set up some basic data structures to answer approximate range counting queries for balls. In Section 4, we present the data structure, query algorithm and proof of correctness for our data structure which can compute $(1 \pm \varepsilon)$-approximate kth-nearest neighbors of a query point when k, ε are only provided during query time. In Section 5 we present approximate quorum clustering, see [9, 14], for a set of disjoint balls. Using this, in Section 6, we present the (k, ε)-AVD construction. We conclude in Section 7.

2 Problem definition and notation

We are given a set of disjoint[1] balls $\mathcal{B} = \{b_1, \ldots, b_n\}$, where $b_i = \mathsf{b}(\mathsf{c}_i, \mathsf{r}_i)$, for $i = 1, \ldots, n$. Here $\mathsf{b}(\mathsf{c}, \mathsf{r}) \subseteq \mathbb{R}^d$ denotes the (closed) ball with center c and radius $\mathsf{r} \geq 0$. Additionally, we are given an approximation parameter $\varepsilon \in (0, 1)$. For a point $\mathsf{q} \in \mathbb{R}^d$, the *distance* of q to a ball $b = \mathsf{b}(\mathsf{c}, \mathsf{r})$ is $\mathsf{d}(\mathsf{q}, b) = \max\Big(\|\mathsf{q} - \mathsf{c}\| - \mathsf{r},\, 0 \Big)$.

▶ **Observation 1.** *For two balls $b_1 \subseteq b_2 \subseteq \mathbb{R}^d$, and any point $\mathsf{q} \in \mathbb{R}^d$, we have $\mathsf{d}(\mathsf{q}, b_1) \geq \mathsf{d}(\mathsf{q}, b_2)$.*

The *kth-nearest neighbor distance* of q to $\mathcal{B}$, denoted by $\mathsf{d}_k(\mathsf{q}, \mathcal{B})$, is the kth smallest number in $\mathsf{d}(\mathsf{q}, b_1), \ldots, \mathsf{d}(\mathsf{q}, b_n)$. Similarly, for a given set of points P, $\mathsf{d}_k(\mathsf{q}, \mathsf{P})$ denotes the kth-nearest neighbor distance of q to P.

We aim to build a data structure to answer $(1 \pm \varepsilon)$-approximate kth-nearest neighbor (i. e., (k, ε)-*ANN*) queries, where for any query point $\mathsf{q} \in \mathbb{R}^d$ one needs to output a ball $b \in \mathcal{B}$ such that, $(1 - \varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq \mathsf{d}(\mathsf{q}, b) \leq (1 + \varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B})$. There are different variants depending on whether ε and k are provided with the query or in advance.

We use *cube* to denote a set of the form $[a_1, a_1 + \ell] \times [a_2, a_2 + \ell] \times \ldots \times [a_d, a_d + \ell] \subseteq \mathbb{R}^d$, where $a_1, \ldots, a_d \in \mathbb{R}$ and $\ell \geq 0$ is the side length of the cube.

▶ **Observation 2.** *For any set of balls $\mathcal{B}$, the function $\mathsf{d}_k(\mathsf{q}, \mathcal{B})$ is a 1-Lipschitz function; that is, for any two points u, v, we have that $\mathsf{d}_k(\mathsf{u}, \mathcal{B}) \leq \mathsf{d}_k(\mathsf{v}, \mathcal{B}) + \|\mathsf{u} - \mathsf{v}\|$.*

▶ **Assumption 3.** *We assume all the balls are contained inside the cube $\Big[1/2 - \delta, 1/2 + \delta\Big]^d$, which can be ensured by translation and scaling (which preserves order of distances), where $\delta = \varepsilon/4$. As such, we can ignore queries outside the unit cube $[0, 1]^d$, as any input ball is a valid answer in this case.*

For a real positive number x and a point $\mathsf{p} = (\mathsf{p}_1, \ldots, \mathsf{p}_d) \in \mathbb{R}^d$, define $\mathsf{G}_x(\mathsf{p})$ to be the grid point $(\lfloor \mathsf{p}_1/x \rfloor\, x, \ldots, \lfloor \mathsf{p}_d/x \rfloor\, x)$. The number x is the *width* or *side length* of the *grid* G_x. The mapping G_x partitions $\mathbb{R}^d$ into cubes that are called grid *cells*.

▶ **Definition 4.** A cube is a *canonical cube* if it is contained inside the unit cube $U = [0, 1]^d$, it is a cell in a grid G_r, and r is a power of two (i. e., it might correspond to a node in a quadtree having $[0, 1]^d$ as its root cell). We will refer to such a grid G_r as a *canonical grid*. Note that all the cells corresponding to nodes of a compressed quadtree are canonical.

[1] Our data structure and algorithm work for the more general case where the balls are interior disjoint, where we define the interior of a "point ball", i. e., a ball of radius 0, as the point itself. This is not the usual topological definition.

▶ **Definition 5.** Given a set $b \subseteq \mathbb{R}^d$, and a parameter $\delta > 0$, let $\mathsf{G}_{\approx}(b, \delta)$ denote the set of canonical grid cells of side length $2^{\lfloor \log_2 \delta \mathrm{diam}(b)/\sqrt{d} \rfloor}$, that intersect b, where $\mathrm{diam}(b) = \max_{\mathsf{p},\mathsf{u} \in b} \|\mathsf{p} - \mathsf{u}\|$ denotes the *diameter* of b. Clearly, the diameter of any grid cell of $\mathsf{G}_{\approx}(b, \delta)$, is at most $\delta \mathrm{diam}(b)$. Let $\mathsf{G}_{\approx}(b) = \mathsf{G}_{\approx}(b, 1)$. It is easy to verify that $|\mathsf{G}_{\approx}(b)| = O(1)$. The set $\mathsf{G}_{\approx}(b)$ is the *grid approximation* to b.

Let $\mathcal{B}$ be a family of balls in $\mathbb{R}^d$. Given a set $X \subseteq \mathbb{R}^d$, let

$$\mathcal{B}(X) = \left\{ b \in \mathcal{B} \;\middle|\; b \cap X \neq \emptyset \right\}$$

denote the set of all balls in $\mathcal{B}$ that intersect X.

For two compact sets $X, Y \subseteq \mathbb{R}^d$, $X \preceq Y$ if and only if $\mathrm{diam}(X) \leq \mathrm{diam}(Y)$. For a set X and a set of balls $\mathcal{B}$, let $\mathcal{B}_{\succeq}(X) = \left\{ b \in \mathcal{B} \;\middle|\; b \cap X \neq \emptyset \text{ and } b \succeq X \right\}$. Let c_d denote the maximum number of pairwise disjoint balls of radius at least r, that may intersect a given ball of radius r in $\mathbb{R}^d$. Clearly, we have $|\mathcal{B}_{\succeq}(b)| \leq \mathsf{c}_d$ for any ball b. The proof of the following lemma appears in the full version [16].

▶ **Lemma 6.** *$2 \leq \mathsf{c}_d \leq 3^d$ for all d.*

▶ **Definition 7.** For a parameter $\delta \geq 0$, a function $f : \mathbb{R}^+ \to \mathbb{R}^+$ is *δ-monotonic*, if for every $x \geq 0$, $f(x/(1+\delta)) \leq f(x)$.

3 Approximate range counting for balls

▶ **Data-structure 8.** For a given set of disjoint balls $\mathcal{B} = \{b_1, \ldots, b_n\}$ in $\mathbb{R}^d$, we build the following data structure, that is useful in performing several of the tasks at hand.

(A) **Store balls in a (compressed) quadtree**. For $i = 1, 2, \ldots, n$, let $G_i = \mathsf{G}_{\approx}(b_i)$, and let $G = \bigcup_{i=1}^{n} G_i$ denote the union of these cells. Let $\mathcal{T}$ be a compressed quadtree decomposition of $[0,1]^d$, such that all the cells of G are cells of $\mathcal{T}$. We preprocess $\mathcal{T}$ to answer point location queries for the cells of G. This takes $O(n \log n)$ time, see [12].

(B) **Compute list of "large" balls intersecting each cell.** For each node u of $\mathcal{T}$, there is a list of balls registered with it. Formally, *register* a ball b_i with all the cells of G_i. Clearly, each ball is registered with $O(1)$ cells, and it is easy to see that each cell has $O(1)$ balls registered with it, since the balls are disjoint.
Next, for a cell $\square$ in $\mathcal{T}$ we compute a list storing $\mathcal{B}_{\succeq}(\square)$, and these balls are *associated* with this cell. These lists are computed in a top-down manner. To this end, propagate from a node u its list $\mathcal{B}_{\succeq}(\square)$ (which we assume is already computed) down to its children. For a node receiving such a list, it scans it, and keep only the balls that intersect its cell (adding to this list the balls already registered with this cell). For a node $\nu \in \mathcal{T}$, let $\mathcal{B}_\nu$ be this list.

(C) **Build compressed quadtree on centers of balls.** Let $\mathcal{C}$ be the set of centers of the balls of $\mathcal{B}$. Build, in $O(n \log n)$ time, a compressed quadtree $\mathcal{T}_{\mathcal{C}}$ storing $\mathcal{C}$.

(D) **ANN for centers of balls.** Build a data structure $\mathcal{D}$, for answering 2-approximate k-nearest neighbor distances on $\mathcal{C}$, the set of centers of the balls, see [14], where k and ε are provided with the query. The data structure $\mathcal{D}$, returns a point $\mathsf{c} \in \mathcal{C}$ such that, $\mathsf{d}_k(\mathsf{q}, \mathcal{C}) \leq \mathsf{d}(\mathsf{q}, \mathsf{c}) \leq 2\mathsf{d}_k(\mathsf{q}, \mathcal{C})$.

(E) **Answering approximate range searching for the centers of balls.** Given a query ball $b_{\mathsf{q}} = \mathsf{b}(\mathsf{q}, x)$ and a parameter $\delta > 0$, one can, using $\mathcal{T}_{\mathcal{C}}$, report (approximately), in $O(\log n + 1/\delta^d)$ time, the points in $b_{\mathsf{q}} \cap \mathcal{C}$. Specifically, the query process computes $O(1/\delta^d)$ sets of points, such that their union X, has the property that $b_{\mathsf{q}} \cap \mathcal{C} \subseteq X \subseteq$

$(1+\delta)b_{\mathsf{q}} \cap \mathcal{C}$, where $(1+\delta)b_{\mathsf{q}}$ is the scaling of b_{q} by a factor of $1+\delta$ around its center. Indeed, compute the set $\mathsf{G}_{\approx}\big(b_{\mathsf{q}}\big)$, and then using cell queries in $\mathcal{T}_{\mathcal{C}}$ compute the corresponding cells (this takes $O(\log n)$ time). Now, descend to the relevant level of the quadtree to all the cells of the right size, that intersect b_{q}. Clearly, the union of points stored in their subtrees are the desired set. This takes overall $O(\log n + 1/\delta^d)$ time.
A similar data structure for approximate range searching is provided by Arya and Mount [5], and our description above is provided for the sake of completeness.

Overall, it takes $O(n \log n)$ time to build this data structure.

We denote the collection of data structures above by $\mathcal{DS}_8$ and where necessary, specific functionality it provides, say for finding the large balls intersecting a cell, by $\mathcal{DS}_8$ (2).

3.1 Approximate range counting among balls

We need the ability to answer approximate range counting queries on a set of disjoint balls. Specifically, given a set of disjoint balls $\mathcal{B}$, and a query ball b, the target is to compute the size of the set $b \cap \mathcal{B} = \Big\{ b' \in \mathcal{B} \;\Big|\; b' \cap b \neq \emptyset \Big\}$. To make this query computationally fast, we allow an approximation. More precisely, for a ball b a set $\widetilde{b}$ is a $(1+\delta)$-*ball* of b, if $b \subseteq \widetilde{b} \subseteq (1+\delta)b$, where $(1+\delta)b$ is the $(1+\delta)$-scaling of b around its center. The purpose here, given a query ball b, is to compute the size of the set $\widetilde{b} \cap \mathcal{B}$ for some $(1+\delta)$-ball $\widetilde{b}$ of b.

The proofs of the following two lemmas appear in the full version [16].

▶ **Lemma 9.** *Given a compressed quadtree $\mathcal{T}$ of size n, a convex set X, and a parameter $\delta > 0$, one can compute the set of nodes in $\mathcal{T}$, that realizes $\mathsf{G}_{\approx}(X, \delta)$ (see Defnition 5), in $O\big(\log n + 1/\delta^d\big)$ time. Specifically, this outputs a set X_N of nodes, of size $O\big(1/\delta^d\big)$, such that their cells intersect $\mathsf{G}_{\approx}(X, \delta)$, and their parents cell diameter is larger than $\delta\mathrm{diam}(X)$. Note that the cells in X_N might be significantly larger if they are leaves of $\mathcal{T}$.*

▶ **Lemma 10.** *Let X be any convex set in $\mathbb{R}^d$, and let $\delta > 0$ be a parameter. Using $\mathcal{DS}_8$, one can compute, in $O\big(\log n + 1/\delta^d\big)$ time, all the balls of $\mathcal{B}$ that intersect X, with diameter $\geq \delta\mathrm{diam}(X)$.*

3.2 Answering a query

Given a query ball $b_{\mathsf{q}} = \mathsf{b}(\mathsf{q}, x)$, and an approximation parameter $\delta > 0$, our purpose is to compute a number N, such that $\Big|\mathcal{B}\Big(\mathsf{b}(\mathsf{q}, x)\Big)\Big| \leq N \leq \Big|\mathcal{B}\Big(\mathsf{b}(\mathsf{q}, (1+\delta)x)\Big)\Big|$.
The query algorithm works as follows:

- **(A)** Using Lemma 10, compute a set X of all the balls that intersect b_{q} and are of radius $\geq \delta x/4$.
- **(B)** Using $\mathcal{DS}_8$, compute $O(1/\delta^d)$ cells of $\mathcal{T}_{\mathcal{C}}$ that corresponds to $\mathsf{G}_{\approx}\big(b_{\mathsf{q}}(1+\delta/4), \delta/4\big)$. Let N' be the total number of points in $\mathcal{C}$ stored in these nodes.
- **(C)** The quantity $N' + |X|$ is almost the desired quantity, except that we might be counting some of the balls of X twice. To this end, let N'' be the number of balls in X with centers in $\mathsf{G}_{\approx}\big(b_{\mathsf{q}}(1+\delta/4), \delta/4\big)$
- **(D)** Let $N \leftarrow N' + |X| - N''$. Return N.

We only sketch the proof, as the proof is straightforward. Indeed, the union of the cells of $\mathsf{G}_{\approx}\big(b_{\mathsf{q}}(1+\delta/4), \delta/4\big)$ contains $\mathsf{b}(\mathsf{q}, x(1+\delta/4))$ and is contained in $\mathsf{b}(\mathsf{q}, (1+\delta)x)$. All the balls with radius smaller than $\delta x/4$ and intersecting $\mathsf{b}(\mathsf{q}, x)$ have their centers in cells of $\mathsf{G}_{\approx}\big(b_{\mathsf{q}}(1+\delta/4), \delta/4\big)$, and their number is computed correctly. Similarly, the

"large" balls are computed correctly. The last stage ensures we do not over-count by 1 each large ball that also has its center in $\mathsf{G}_{\approx}\big(b_{\mathsf{q}}(1+\delta/4), \delta/4\big)$. It is also easy to check that $|\mathcal{B}(\mathsf{b}(\mathsf{q}, x))| \leq N \leq |\mathcal{B}(\mathsf{b}(\mathsf{q}, x(1+\delta)))|$. The same result can be used for $x/(1+\delta)$ to get δ-monotonicity of N.

We now analyze the running time. Computing all the cells of $\mathsf{G}_{\approx}\big(b_{\mathsf{q}}(1+\delta/4), \delta/4\big)$ takes $O(\log n + 1/\delta^d)$ time. Computing the "large" balls takes $O\big(\log n + 1/\delta^d\big)$ time. Checking for each large ball if it is already counted by the "small" balls takes $O(1/\delta^d)$ by using a grid. We denote the above query algorithm by $\mathbf{rangeCount}(\mathsf{q}, x, \delta)$.

The above implies the following.

▶ **Lemma 11.** *Given a set $\mathcal{B}$ of n disjoint balls in $\mathbb{R}^d$, it can be preprocessed, in $O(n \log n)$ time, into a data structure of size $O(n)$, such that given a query ball $\mathsf{b}(\mathsf{q}, x)$ and approximation parameter $\delta > 0$, the query algorithm $\mathbf{rangeCount}(\mathsf{q}, x, \delta)$ returns, in $O(\log n + 1/\delta^d)$ time, a number N satisfying the following:*

- **(A)** $N \leq |\mathcal{B}(\mathsf{b}(\mathsf{q}, (1+\delta)x))|$,
- **(B)** $|\mathcal{B}(\mathsf{b}(\mathsf{q}, x))| \leq N$, *and*
- **(C)** *for a query ball $\mathsf{b}(\mathsf{q}, x)$ and δ, the number N is δ-monotonic as a function of x, see Defnition 7.*

4 Answering k-ANN queries among balls

4.1 Computing a constant factor approximation to $\mathsf{d}_k(\mathsf{q}, \mathcal{B})$

The proof of the following lemma appears in the full version [16].

▶ **Lemma 12.** *Let $\mathcal{B}$ be a set of disjoint balls in $\mathbb{R}^d$, and consider a ball $b = \mathsf{b}(\mathsf{q}, r)$ that intersects at least k balls of $\mathcal{B}$. Then, among the k nearest neighbors of q from $\mathcal{B}$, there are at least $\max(0, k - \mathsf{c}_d)$ balls of radius at most r. The centers of all these balls are in $\mathsf{b}(\mathsf{q}, 2r)$.*

▶ **Corollary 13.** *Let $\gamma = \min(k, \mathsf{c}_d)$. Then, $\mathsf{d}_{k-\gamma}(\mathsf{q}, \mathcal{C})/2 \leq \mathsf{d}_k(\mathsf{q}, \mathcal{B})$.*

The basic observation is that we only need a rough approximation to the right radius, as using approximate range counting (i. e., Lemma 11), one can improve the approximation.

Let x_i denote the distance of q to the ith closest center in $\mathcal{C}$. Let $d_k = \mathsf{d}_k(\mathsf{q}, \mathcal{B})$. Let i be the minimum index, such that $d_k \leq x_i$. Since $d_k \leq x_k$, it must be that $i \leq k$. There are several possibilities:

- **(A)** If $i \leq k - \mathsf{c}_d$ (i. e., $d_k \leq x_{k-\mathsf{c}_d}$) then, by Lemma 12, the ball $\mathsf{b}(\mathsf{q}, 2d_k)$ contains at least $k - \mathsf{c}_d$ centers. As such, $d_k < x_{k-\mathsf{c}_d} \leq 2d_k$, and $x_{k-\mathsf{c}_d}$ is a good approximation to d_k.
- **(B)** If $i > k - \mathsf{c}_d$, and $d_k \leq 4x_{i-1}$, then x_{i-1} is the desired approximation.
- **(C)** If $i > k - \mathsf{c}_d$, and $d_k \geq x_i/4$, then x_i is the desired approximation.
- **(D)** Otherwise, it must be that $i > k - \mathsf{c}_d$, and $4x_{i-1} < d_k < x_i/4$. Let $b_j = \mathsf{b}(\mathsf{c}_j, \mathsf{r}_j)$ be the jth closest ball to q, for $j = 1, \ldots, k$. It must be that $b_i, \ldots, b_k$ are much larger than $\mathsf{b}(\mathsf{q}, d_k)$. But then, the balls $b_i, \ldots, b_k$ must intersect $\mathsf{b}(\mathsf{q}, x_i/2)$, and their radius is at least $x_i/2$. We can easily compute these big balls using $\mathcal{DS}_8$ (2), and the number of centers of the small balls close to query, and then compute d_k exactly.

We build $\mathcal{DS}_8$ in $O(n \log n)$ time.

First we introduce some notation. For $x \geq 0$, let $N(x)$ denote the number of balls in $\mathcal{B}$ that intersect $\mathsf{b}(\mathsf{q}, x)$; that is $N(x) = \left|\left\{ b \in \mathcal{B} \;\middle|\; b \cap \mathsf{b}(\mathsf{q}, x) \neq \emptyset \right\}\right|$, and $C(x)$ denote the number of centers in $\mathsf{b}(\mathsf{q}, x)$, i. e., $C(x) = |\mathcal{C} \cap \mathsf{b}(\mathsf{q}, x)|$. Also, let $\#(x)$ denote the 2-approximation to the number of balls of $\mathcal{B}$ intersecting $\mathsf{b}(\mathsf{q}, x)$, as computed by Lemma 11; that is $N(x) \leq \#(x) \leq N(2x)$.

We now provide our algorithm to answer a query. We are given a query point $\mathsf{q} \in \mathbb{R}^d$ and a number k.

Using $\mathcal{DS}_8$, compute a 2-approximation for the smallest ball containing $k-i$ centers of $\mathcal{B}$, for $i = 0, \ldots, \gamma$, where $\gamma = \min(k, \mathsf{c}_d)$, and let r_{k-i} be this radius. That is, for $i = 0, \ldots, \gamma$, we have $C(r_{k-i}/2) \leq k - i \leq C(r_{k-i})$. For $i = 0, \ldots, \gamma$, compute $N_{k-i} = \#(r_{k-i})$ (Lemma 11).

Let α be the maximum index such that $N_{k-\alpha} \geq k$. Clearly, α is well defined as $N_k \geq k$. The algorithm is executed in the following steps.

(A) If $\alpha = \gamma$ we return $2r_{k-\gamma}$.

(B) If $\#(r_{k-\alpha}/4) < k$, we return $2r_{k-\alpha}$.

(C) Otherwise, compute all the balls of $\mathcal{B}$ that are of radius at least $r_{k-\alpha}/4$ and intersect the ball $\mathsf{b}(\mathsf{q}, r_{k-\alpha}/4)$, using $\mathcal{DS}_8$ (2). For each such ball b, compute the distance $\zeta = \mathsf{d}(\mathsf{q}, b)$ of q to it. Return 2ζ for the minimum such number ζ such that $\#(\zeta) \geq k$.

The proof of the following lemma appears in the full version [16].

▶ **Lemma 14.** *Given a set of n disjoint balls $\mathcal{B}$ in $\mathbb{R}^d$, one can preprocess them, in $O(n \log n)$ time, into a data structure of size $O(n)$, such that given a query point $\mathsf{q} \in \mathbb{R}^d$, and a number k, one can compute, in $O(\log n)$ time, a number x such that, $x/4 \leq \mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq 4x$.*

We now show how to refine the approximation in the following lemma, whose proof appears in the full version [16].

▶ **Lemma 15.** *Given a set $\mathcal{B}$ of n balls in $\mathbb{R}^d$, it can be preprocessed, in $O(n \log n)$ time, into a data structure of size $O(n)$. Given a query point q, numbers k, x, and an approximation parameter $\varepsilon > 0$, such that $x/4 \leq \mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq 4x$, one can find a ball $b \in \mathcal{B}$ such that, $(1-\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq \mathsf{d}(\mathsf{q}, b) \leq (1+\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B})$, in $O\big(\log n + 1/\varepsilon^d\big)$ time.*

4.2 The result

▶ **Theorem 16.** *Given a set of n disjoint balls $\mathcal{B}$ in $\mathbb{R}^d$, one can preprocess them in time $O(n \log n)$ into a data structure of size $O(n)$, such that given a query point $\mathsf{q} \in \mathbb{R}^d$, a number k with $1 \leq k \leq n$ and $\varepsilon > 0$, one can find in time $O\big(\log n + \varepsilon^{-d}\big)$ a ball $b \in \mathcal{B}$, such that, $(1-\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq \mathsf{d}(\mathsf{q}, b) \leq (1+\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B})$.*

5 Quorum clustering

We are given a set $\mathcal{B}$ of n disjoint balls in $\mathbb{R}^d$, and we describe how to compute quorum clustering for them quickly.

Let ξ be some constant. Let $\mathcal{B}_0 = \emptyset$. For $i = 1, \ldots, m$, let $\mathcal{R}_i = \mathcal{B} \setminus (\bigcup_{j=0}^{i-1} \mathcal{B}_j)$, and let $\Lambda_i = \mathsf{b}(\mathsf{w}_i, \mathsf{x}_i)$ be any ball that satisfies,

(A) Λ_i contains $\min(k - \mathsf{c}_d, |\mathcal{R}_i|)$ balls of $\mathcal{R}_i$ completely inside it,

(B) Λ_i intersects at least k balls of $\mathcal{B}$, and

(C) the radius of Λ_i is at most ξ times the radius of the smallest ball satisfying the above conditions.

Next, we remove any $k - \mathsf{c}_d$ balls that are contained in Λ_i from $\mathcal{R}_i$ to get the set $\mathcal{R}_{i+1}$. We call the removed set of balls $\mathcal{B}_i$. We repeat this process till all balls are extracted. Notice that at each step i, we only require that the Λ_i intersects k balls of $\mathcal{B}$ (and not $\mathcal{R}_i$), but that it must contain $k - \mathsf{c}_d$ balls from $\mathcal{R}_i$. Also, the last quorum ball may contain fewer balls. The balls $\Lambda_1, \ldots, \Lambda_m$, are the resulting *$\xi$-approximate quorum clustering*.

5.1 Computing an approximate quorum clustering

▶ **Definition 17.** For a set P of n points in $\mathbb{R}^d$, and an integer ℓ, with $1 \leq \ell \leq n$, let $r_{\mathrm{opt}}(\mathsf{P}, \ell)$ denote the radius of the smallest ball which contains at least ℓ points from P, i. e., $r_{\mathrm{opt}}(\mathsf{P}, \ell) = \min_{\mathsf{q} \in \mathbb{R}^d} \mathsf{d}_\ell(\mathsf{q}, \mathsf{P})$.

Similarly, for a set $\mathcal{R}$ of n balls in $\mathbb{R}^d$, and an integer ℓ, with $1 \leq \ell \leq n$, let $R_{\mathrm{opt}}(\mathcal{R}, \ell)$ denote the radius of the smallest ball which completely contains at least ℓ balls from $\mathcal{R}$.

▶ **Lemma 18** ([14]). *Given a set P of n points in $\mathbb{R}^d$ and integer ℓ, with $1 \leq \ell \leq n$, one can compute, in $O(n \log n)$ time, a sequence of $\lceil n/\ell \rceil$ balls, $\mathsf{o}_1 = \mathsf{b}(\mathsf{u}_1, \psi_1), \ldots, \mathsf{o}_{\lceil n/\ell \rceil} = \mathsf{b}(\mathsf{u}_{\lceil n/\ell \rceil}, \psi_{\lceil n/\ell \rceil})$, such that, for all $i, 1 \leq i \leq \lceil n/\ell \rceil$, we have*

- **(A)** *For every ball o_i, there is an associated subset P_i of $\min(\ell, |\mathsf{Q}_i|)$ points of $\mathsf{Q}_i = \mathsf{P} \setminus (\mathsf{P}_i \cup \ldots \cup \mathsf{P}_{i-1})$, that it covers.*
- **(B)** *The ball $\mathsf{o}_i = \mathsf{b}(\mathsf{u}_i, \psi_i)$ is a 2-approximation to the smallest ball covering $\min(\ell, |\mathsf{Q}_i|)$ points in Q_i; that is, $\psi_i/2 \leq r_{\mathrm{opt}}(\mathsf{Q}_i, \min(\ell, |\mathsf{Q}_i|)) \leq \psi_i$.*

The algorithm to construct an approximate quorum clustering is as follows. We use the algorithm of Lemma 18 with the set of points $\mathsf{P} = \mathcal{C}$, and $\ell = k - \mathsf{c}_d$ to get a list of $m = \lceil n/(k - \mathsf{c}_d) \rceil$ balls $\mathsf{o}_1 = \mathsf{b}(\mathsf{u}_1, \psi_1), \ldots, \mathsf{o}_m = \mathsf{b}(\mathsf{u}_m, \psi_m)$, satisfying the conditions of Lemma 18. Next we use the algorithm of Theorem 16, to compute (k, ε)-ANN distances from the centers $\mathsf{u}_1, \ldots, \mathsf{u}_m$, to the balls of $\mathcal{B}$.

Thus, we get numbers γ_i satisfying, $(1/2)\mathsf{d}_k(\mathsf{u}_i, \mathcal{B}) \leq \gamma_i \leq (3/2)\mathsf{d}_k(\mathsf{u}_i, \mathcal{B})$. Let $\zeta_i = \max(2\gamma_i, 3\psi_i)$, for $i = 1, \ldots, m$. Sort $\zeta_1, \ldots, \zeta_m$ (we assume for the sake of simplicity of exposition that ζ_m, being the radius of the last cluster is the largest number). Suppose the sorted order is the permutation π of $\{1, \ldots, m\}$ (by assumption $\pi(m) = m$). We output the balls $\Lambda_i = \mathsf{b}(\mathsf{u}_{\pi(i)}, \zeta_{\pi(i)})$, for $i = 1, \ldots, m$, as the approximate quorum clustering.

5.2 Correctness

The following lemmas prove the correctness of our approximate quorum clustering algorithm. Their proofs appear in the full version [16].

▶ **Lemma 19.** *Let $\mathcal{B} = \{b_1, \ldots, b_n\}$ be a set of n disjoint balls, where $b_i = \mathsf{b}(\mathsf{c}_i, \mathsf{r}_i)$, for $i = 1, \ldots, n$. Let $\mathcal{C} = \{\mathsf{c}_1, \ldots, \mathsf{c}_n\}$ be the set of centers of these balls. Let $b = \mathsf{b}(\mathsf{c}, \mathsf{r})$ be any ball that contains at least ℓ centers from $\mathcal{C}$, for some $2 \leq \ell \leq n$. Then $\mathsf{b}(\mathsf{c}, 3\mathsf{r})$ contains the ℓ balls that correspond to those centers.*

▶ **Lemma 20.** *Let $\mathcal{B} = \{b_1 = \mathsf{b}(\mathsf{c}_1, \mathsf{r}_1), \ldots, b_n = \mathsf{b}(\mathsf{c}_n, \mathsf{r}_n)\}$ be a set of n disjoint balls in $\mathbb{R}^d$. Let $\mathcal{C} = \{\mathsf{c}_1, \ldots, \mathsf{c}_n\}$ be the corresponding set of centers, and let ℓ be an integer with $2 \leq \ell \leq n$. Then, $r_{\mathrm{opt}}(\mathcal{C}, \ell) \leq R_{\mathrm{opt}}(\mathcal{B}, \ell) \leq 3r_{\mathrm{opt}}(\mathcal{C}, \ell)$.*

▶ **Lemma 21.** *The balls $\Lambda_1, \ldots \Lambda_m$ computed above are a 12-approximate quorum clustering of $\mathcal{B}$.*

▶ **Lemma 22.** *Given a set $\mathcal{B}$ of n disjoint balls in $\mathbb{R}^d$, such that $(k - \mathsf{c}_d)|n$, and a number k with $2\mathsf{c}_d < k \leq n$, in $O(n \log n)$ time, one can output a sequence of $m = n/(k - \mathsf{c}_d)$ balls $\Lambda_1, \ldots, \Lambda_m$, such that*

- **(A)** *For each ball Λ_i, there is an associated subset $\mathcal{B}_i$ of $k - \mathsf{c}_d$ balls of $\mathcal{R}_i = \mathcal{B} \setminus (\mathcal{B}_1 \cup \ldots \cup \mathcal{B}_{i-1})$, that it completely covers.*
- **(B)** *The ball Λ_i intersects at least k balls from $\mathcal{B}$.*
- **(C)** *The radius of the ball Λ_i is at most 12 times that of the smallest ball covering $k - \mathsf{c}_d$ balls of $\mathcal{R}_i$ completely, and intersecting k balls of $\mathcal{B}$.*

Proof. The correctness was proved in Lemma 21. To see the time bound is also easy as the computation time is dominated by the time in Lemma 18, which is $O(n \log n)$. ◀

6 Construction of the sublinear space data structure for (k, ε)-ANN

Here we show how to compute an approximate Voronoi diagram for approximating the kth-nearest ball, that takes $O(n/k)$ space. We assume $k > 2\mathsf{c}_d$ without loss of generality, and we let $m = \lceil n/(k - \mathsf{c}_d) \rceil = O(n/k)$. Here k and ε are prespecified in advance.

6.1 Preliminaries

The following notation was introduced in [14]. A ball b of radius r in $\mathbb{R}^d$, centered at a point c, can be interpreted as a point in $\mathbb{R}^{d+1}$, denoted by $b' = (\mathsf{c}, \mathsf{r})$. For a regular point $\mathsf{p} \in \mathbb{R}^d$, its corresponding image under this transformation is the *mapped* point $\mathsf{p}' = (\mathsf{p}, 0) \in \mathbb{R}^{d+1}$, i. e., we view it as a ball of radius 0 and use the mapping defined on balls. Given point $\mathsf{u} = (\mathsf{u}_1, \ldots, \mathsf{u}_d) \in \mathbb{R}^d$ we will denote its Euclidean norm by $\|\mathsf{u}\|$. We will consider a point $\mathsf{u} = (\mathsf{u}_1, \mathsf{u}_2, \ldots, \mathsf{u}_{d+1}) \in \mathbb{R}^{d+1}$ to be in the product metric of $\mathbb{R}^d \times \mathbb{R}$ and endowed with the product metric norm

$$\|\mathsf{u}\|_{\oplus} = \sqrt{\mathsf{u}_1^2 + \cdots + \mathsf{u}_d^2} + |\mathsf{u}_{d+1}| \,.$$

It can be verified that the above defines a norm, and for any $\mathsf{u} \in \mathbb{R}^{d+1}$ we have $\|\mathsf{u}\| \leq \|\mathsf{u}\|_{\oplus} \leq \sqrt{2}\,\|\mathsf{u}\|$.

6.2 Construction

The input is a set $\mathcal{B}$ of n disjoint balls in $\mathbb{R}^d$, and parameters k and ε.

The construction of the data structure is similar to the construction of the kth-nearest neighbor data structure from the authors' paper [14]. We compute, using Lemma 22, a ξ-approximate quorum clustering of $\mathcal{B}$ with $m = n/(k - \mathsf{c}_d) = O(n/k)$ balls, $\Sigma = \{\Lambda_1 = \mathsf{b}(\mathsf{w}_1, \mathsf{x}_1), \ldots, \Lambda_m = \mathsf{b}(\mathsf{w}_m, \mathsf{x}_m)\}$, where $\xi \leq 12$. The algorithm then continues as follows:

(A) Compute an exponential grid around each quorum cluster. Specifically, let

$$\mathcal{I} = \bigcup_{i=1}^{m} \bigcup_{j=0}^{\lceil \log(32\xi/\varepsilon) \rceil} \mathsf{G}_{\approx}\left(\mathsf{b}(\mathsf{w}_i, 2^j \mathsf{x}_i), \frac{\varepsilon}{\zeta_1}\right) \tag{6.1}$$

be the set of grid cells covering the quorum clusters and their immediate environ, where ζ_1 is a sufficiently large constant (say, $\zeta_1 = 256\xi$).

(B) Intuitively, $\mathcal{I}$ takes care of the region of space immediately next to a quorum cluster[2]. For the other regions of space, we can apply a construction of an approximate Voronoi diagram for the centers of the clusters (the details are somewhat more involved). To this end, lift the quorum clusters into points in $\mathbb{R}^{d+1}$, as follows

$$\Sigma' = \{\Lambda'_1, \ldots, \Lambda'_m\} \,,$$

[2] That is, intuitively, if the query point falls into one of the grid cells of $\mathcal{I}$, we can answer a query in constant time.

where $\Lambda'_i = (\mathsf{w}_i, \mathsf{x}_i) \in \mathbb{R}^{d+1}$, for $i = 1, \ldots, m$. Note that all points in Σ' belong to $U' = [0,1]^{d+1}$ by Assumption 3. Now build a $(1+\varepsilon/8)$-AVD for Σ' using the algorithm of Arya and Malamatos [2], for distances specified by the $\|\cdot\|_{\oplus}$ norm. The AVD construction provides a list of canonical cubes covering $[0,1]^{d+1}$ such that in the smallest cube containing the query point, the associated point of Σ', is a $(1+\varepsilon/8)$-ANN to the query point. (Note that these cubes are not necessarily disjoint. In particular, the smallest cube containing the query point q is the one that determines the assigned approximate nearest neighbor to q.)

Clip this collection of cubes to the hyperplane $x_{d+1} = 0$ (i. e., throw away cubes that do not have a face on this hyperplane). For a cube $\square$ in this collection, denote by $\mathrm{nn}'(\square)$, the point of Σ' assigned to it. Let $\mathcal{S}$ be this resulting set of canonical d-dimensional cubes.

(C) Let $\mathcal{W}$ be the space decomposition resulting from overlaying the two collection of cubes, i. e. $\mathcal{I}$ and $\mathcal{S}$. Formally, we compute a compressed quadtree $\mathcal{T}$ that has all the canonical cubes of $\mathcal{I}$ and $\mathcal{S}$ as nodes, and $\mathcal{W}$ is the resulting decomposition of space into cells. One can overlay two compressed quadtrees representing the two sets in linear time [7, 12]. Here, a cell associated with a leaf is a canonical cube, and a cell associated with a compressed node is the set difference of two canonical cubes. Each node in this compressed quadtree contains two pointers – to the smallest cube of $\mathcal{I}$, and to the smallest cube of $\mathcal{S}$, that contains it. This information can be computed by doing a BFS on the tree.

For each cell $\square \in \mathcal{W}$ we store the following.

- **(I)** An arbitrary representative point $\square_{\mathsf{rep}} \in \square$.
- **(II)** The point $\mathrm{nn}'(\square) \in \Sigma'$ that is associated with the smallest cell of $\mathcal{S}$ that contains this cell. We also store an arbitrary ball, $\mathbf{b}(\square) \in \mathcal{B}$, that is one of the balls completely inside the cluster specified by $\mathrm{nn}'(\square)$ – we assume we stored such a ball inside each quorum cluster, when it was computed.
- **(III)** A number $\beta_k(\square_{\mathsf{rep}})$ that satisfies $\mathsf{d}_k(\square_{\mathsf{rep}}, \mathcal{B}) \leq \beta_k(\square_{\mathsf{rep}}) \leq (1+\varepsilon/4)\mathsf{d}_k(\square_{\mathsf{rep}}, \mathcal{B})$, and a ball $\mathrm{nn}_k(\square_{\mathsf{rep}}) \in \mathcal{B}$ that realizes this distance. In order to compute $\beta_k(\square_{\mathsf{rep}})$ and $\mathrm{nn}_k(\square_{\mathsf{rep}})$ use the data structure of Section 4, see Theorem 16.

6.3 Answering a query

Given a query point q, compute the leaf cell (equivalently the smallest cell) in $\mathcal{W}$ that contains q by performing a point-location query in the compressed quadtree $\mathcal{T}$. Let $\square$ be this cell. Let,

$$\lambda^* = \min\left(\|\mathsf{q}' - \mathrm{nn}'(\square)\|_{\oplus}, \beta_k(\square_{\mathsf{rep}}) + \|\mathsf{q} - \square_{\mathsf{rep}}\|\right). \tag{6.2}$$

If $\mathsf{diam}(\square) \leq (\varepsilon/8)\lambda^*$ we return $\mathrm{nn}_k(\square_{\mathsf{rep}})$ as the approximate kth-nearest neighbor, else we return $\mathbf{b}(\square)$.

6.4 Correctness

▶ **Lemma 23.** *The number* $\lambda^* = \min\left(\|\mathsf{q}' - \mathrm{nn}'(\square)\|_{\oplus}, \beta_k(\square_{\mathsf{rep}}) + \|\mathsf{q} - \square_{\mathsf{rep}}\|\right)$ *satisfies,* $\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq \lambda^*$.

Proof. This follows by the Lipschitz property, see Observation 2. ◀

The proofs of the following lemmas appear in the full version [16].

▶ **Lemma 24.** *Let $\square \in \mathcal{W}$ be any cell containing q. If $\mathsf{diam}(\square) \leq \varepsilon \mathsf{d}_k(\mathsf{q}, \mathcal{B})/4$, then $\mathrm{nn}_k(\square_{\mathsf{rep}})$ is a valid $(1 \pm \varepsilon)$-approximate kth-nearest neighbor of q.*

▶ **Lemma 25.** *For any point $\mathsf{q} \in \mathbb{R}^d$ there is a quorum ball $\Lambda_i = \mathsf{b}(\mathsf{w}_i, \mathsf{x}_i)$ such that (A) Λ_i intersects $\mathsf{b}(\mathsf{q}, \mathsf{d}_k(\mathsf{q}, \mathcal{B}))$, (B) $\mathsf{x}_i \leq 3\xi \mathsf{d}_k(\mathsf{q}, \mathcal{B})$, and (C) $\|\mathsf{q} - \mathsf{w}_i\| \leq 4\xi \mathsf{d}_k(\mathsf{q}, \mathcal{B})$.*

▶ **Definition 26.** For a given query point, any quorum cluster that satisfies the conditions of Lemma 25 is defined to be an *anchor cluster*. By Lemma 25 an anchor cluster always exists.

The following lemma, whose proof appears in the full version [16], gives a condition under which the output of the algorithm is correct.

▶ **Lemma 27.** *Suppose that among the quorum cluster balls $\Lambda_1, \ldots, \Lambda_m$, there is some ball $\Lambda_i = \mathsf{b}(\mathsf{w}_i, \mathsf{x}_i)$ which satisfies that $\|\mathsf{q} - \mathsf{w}_i\| \leq 8\xi \mathsf{d}_k(\mathsf{q}, \mathcal{B})$ and $\varepsilon \mathsf{d}_k(\mathsf{q}, \mathcal{B})/4 \leq \mathsf{x}_i \leq 8\xi \mathsf{d}_k(\mathsf{q}, \mathcal{B})$ then the output of the algorithm is correct.*

The next lemma, whose proof appears in the full version [16], proves the correctness for the general case.

▶ **Lemma 28.** *The query algorithm always outputs a correct approximate answer, i. e., the output ball b satisfies $(1 - \varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq \mathsf{d}(\mathsf{q}, b) \leq (1 + \varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B})$.*

6.5 The result

The following theorem encapsulates our main result for this section. Its proof appears in the full version [16].

▶ **Theorem 29.** *Given a set $\mathcal{B}$ of n disjoint balls in $\mathbb{R}^d$, a number k, with $1 \leq k \leq n$, and $\varepsilon \in (0, 1)$, one can preprocess $\mathcal{B}$, in $O\left(n \log n + \frac{n}{k} C_\varepsilon \log n + \frac{n}{k} C'_\varepsilon\right)$ time, where $C_\varepsilon = O\left(\varepsilon^{-d} \log \varepsilon^{-1}\right)$ and $C'_\varepsilon = O\left(\varepsilon^{-2d} \log \varepsilon^{-1}\right)$. The space used by the data structure is $O(C_\varepsilon n/k)$. Given a query point q, this data structure outputs a ball $b \in \mathcal{B}$ in $O\left(\log \frac{n}{k\varepsilon}\right)$ time, such that $(1 - \varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq \mathsf{d}(\mathsf{q}, b) \leq (1 + \varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B})$.*

Note that the space decomposition generated by Theorem 29 can be interpreted as a space decomposition of complexity $O(C_\varepsilon n/k)$, where every cell has two input balls associated with it, which are the candidates to be the desired (k, ε)-ANN. That is, Theorem 29 computes a $(k.\varepsilon)$-AVD of the input balls.

7 Conclusions

In this paper, we presented a generalization of the usual $(1 \pm \varepsilon)$-approximate kth-nearest neighbor problem in $\mathbb{R}^d$, where the input are balls of arbitrary radius, while the query is a point. We first presented a data structure that takes $O(n)$ space, and the query time is $O(\log n + \varepsilon^{-d})$. Here, both k and ε could be supplied at query time. Next we presented an (k, ε)-AVD taking $O(n/k)$ space. Thus showing, surprisingly, that the problem can be solved in sublinear space if k is sufficiently large.

References

1 A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.

2 S. Arya and T. Malamatos. Linear-size approximate Voronoi diagrams. In *Proc. 13th ACM-SIAM Symp. Discrete Algs.*, pages 147–155, 2002.

3 S. Arya, T. Malamatos, and D. M. Mount. Space-time tradeoffs for approximate spherical range counting. In *Proc. 16th ACM-SIAM Symp. Discrete Algs.*, pages 535–544, 2005.

4 S. Arya, T. Malamatos, and D. M. Mount. Space-time tradeoffs for approximate nearest neighbor searching. *J. Assoc. Comput. Mach.*, 57(1):1–54, 2009.

5 S. Arya and D. M. Mount. Approximate range searching. *Comput. Geom. Theory Appl.*, 17:135–152, 2000.

6 S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *J. Assoc. Comput. Mach.*, 45(6):891–923, 1998.

7 M. de Berg, H. Haverkort, S. Thite, and L. Toma. Star-quadtrees and guard-quadtrees: I/O-efficient indexes for fat triangulations and low-density planar subdivisions. *Comput. Geom. Theory Appl.*, 43:493–513, July 2010.

8 P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. Assoc. Comput. Mach.*, 42:67–90, 1995.

9 P. Carmi, S. Dolev, S. Har-Peled, M. J. Katz, and M. Segal. Geographic quorum systems approximations. *Algorithmica*, 41(4):233–244, 2005.

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

11 S. Har-Peled. A replacement for Voronoi diagrams of near linear size. In *Proc. 42nd Annual IEEE Symp. Found. Comput. Sci.*, pages 94–103, 2001.

12 S. Har-Peled. *Geometric Approximation Algorithms*, volume 173 of *Mathematical Surveys and Monographs*. Amer. Math. Soc., 2011.

13 S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. *Theory Comput.*, 8:321–350, 2012. Special issue in honor of Rajeev Motwani.

14 S. Har-Peled and N. Kumar. Down the rabbit hole: Robust proximity search in sublinear space. In *Proc. 53rd Annual IEEE Symp. Found. Comput. Sci.*, pages 430–439, 2012.

15 S. Har-Peled and N. Kumar. Approximating minimization diagrams and generalized proximity search. In *Proc. 54th Annual IEEE Symp. Found. Comput. Sci.*, pages 717–726, 2013.

16 S. Har-Peled and N. Kumar. Robust proximity search for balls using sublinear space. *CoRR*, abs/1401.1472, 2014.

17 P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th Annual ACM Symp. Theory Comput.*, pages 604–613, 1998.

18 G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)*. The MIT Press, 2006.