Discrete Comput Geom (2016) 56:357–376
DOI 10.1007/s00454-016-9801-7

Space Exploration via Proximity Search

Sariel Har-Peled[1] · **Nirman Kumar**[2] ·
David M. Mount[3] · **Benjamin Raichel**[4]

Received: 13 July 2015 / Revised: 10 May 2016 / Accepted: 8 June 2016 /
Published online: 8 July 2016

Abstract We investigate what computational tasks can be performed on a point set in $\mathbb{R}^d$, if we are only given black-box access to it via nearest-neighbor search. This is a reasonable assumption if the underlying point set is either provided implicitly, or it is stored in a data structure that can answer such queries. In particular, we show the following:

(A) One can compute an approximate bi-criteria k-center clustering of the point set, and more generally compute a greedy permutation of the point set.

(B) One can decide if a query point is (approximately) inside the convex-hull of the point set.

Editor in Charge: Kenneth Clarkson

Sariel Har-Peled
sariel@illinois.edu

Nirman Kumar
nirman@cs.ucsb.edu

David M. Mount
mount@cs.umd.edu

Benjamin Raichel
bar150630@utdallas.edu

[1] Department of Computer Science, University of Illinois, 201 N. Goodwin Avenue, Urbana, IL 61801, USA

[2] Department of Computer Science, University of California, 2120B Harold Frank Hall, Santa Barbara, CA 93106, USA

[3] Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA

[4] Department of Computer Science, University of Texas at Dallas, 800 W. Campbell Rd., MS EC-31, Richardson, TX 75080, USA

We also investigate the problem of clustering the given point set, such that meaningful proximity queries can be carried out on the centers of the clusters, instead of the whole point set.

Keywords Nearest neighbors · Clustering · Approximation algorithms

1 Introduction

Many problems in Computational Geometry involve sets of points in $\mathbb{R}^d$. Traditionally, such a point set is presented explicitly, say, as a list of coordinate vectors. There are, however, numerous applications in science and engineering where point sets are presented *implicitly*. This may arise for various reasons: (i) the point set (which might be infinite) is a physical structure that is represented in terms of a finite set of sensed measurements such as a point cloud, (ii) the set is too large to be stored explicitly in memory, or the set is procedurally generated from a highly compressed form. (A number of concrete examples are described below.)

Access to such an implicitly-represented point set P is performed through an *oracle* that is capable of answering queries of a particular type. We can think of this oracle as a black-box data structure, which is provided to us in lieu of an explicit representation. Various types of probes have been studied (such as finger probes, line probes, and X-ray probes [24]). Most of these assume that P is connected (e.g., a convex polygon) and cannot be applied when dealing with arbitrary point sets. In this paper, we consider *proximity probes*—a natural choice for probing general point sets based on computing nearest neighbors.

More formally, we assume that the point set P is a (not necessarily finite) compact subset of $\mathbb{R}^d$. The point set P is accessible only through a nearest-neighbor data structure, which given a query point q, returns the closest point of P to q. Some of our results assume that the data structure returns an exact nearest neighbor (NN) and others assume that the data structure returns a $(1+\varepsilon)$-approximate nearest-neighbor (ANN). (See Sect. 2 for definitions.) In any probing scenario, it is necessary to begin with a general notion of the set's spatial location. The point set P is contained within a given *domain*, which is a compact subset $\mathcal{D}$ of $\mathbb{R}^d$. Specifically, throughout the paper we assume $\mathcal{D}$ is the unit hypercube $[0, 1]^d$.

The oracle is given as a black-box, and no deletions or insertions are allowed from the data structure. Furthermore, the number of data points in P is not necessarily known, nor is there any assumption on continuity or smoothness. Indeed, most of our results apply to infinite point sets, including volumes or surfaces.

1.1 Prior Work and Applications

Implicitly-represented point sets arise in various applications. One example is that of analyzing a geometric shape through probing. An example of this is Atomic Force Microscopy (AFM) [26]. This technology can reveal the undulations of a surface at the resolution of fractions of a nanometer. It relies on the principle that when an appropriately designed tip (the probe) is brought in the proximity of a surface to scan it, certain atomic forces minutely deflect the tip in the direction of the surface. Since the deflection of the tip is generally to the closest point on the surface, this mode of acquisition is an example of proximity probing. A sufficient number of such samples can be used to reconstruct the surface [4].

The topic of shape analysis through probing has been well studied within the field of computational geometry. The most commonly assumed probe is a *finger probe*, which determines the first point of contact of a ray and the set. Cole and Yap [8] pioneered this area by analyzing the minimum number of finger probes needed to reconstruct a convex polygon. Since then, various alternative probing methods have been considered. For good surveys of this area, see Skiena [24,25].

More recently, Boissonnat et al. [6] presented an algorithm for learning a smooth unknown surface S bounding an object $\mathcal{O}$ in $\mathbb{R}^3$ through the use of finger probes. Under some reasonable assumptions, their algorithm computes a triangulated surface $\widehat{S}$ that approximates S to a given level of accuracy. In contrast to our work, which applies to general point sets, all of these earlier results assume that the set in question is a connected shape or surface.

Implicitly-represented point sets also arise in geometric modeling. Complex geometric sets are often generated from much smaller representations. One example are fractals sets, which are often used to model natural phenomena such as plants, clouds, and terrains [23]. Fractals are often expressed as the limit of an iterative process [18]. Due to their regular, recursive structure it is often possible to answer proximity queries about such a set without generating the set itself.

Two other examples of infinite sets generated implicitly from finite models include (I) subdivision surfaces [1], where a smooth surface is generated by applying a recursive refinement process to a finite set of boundary points, and (II) metaballs [5], where a surface is defined by a blending function applied to a collection of geometric balls. In both cases, it is possible to answer nearest neighbor queries for the underlying object to arbitrarily high precision without the need to generate its boundary.

Proximity queries have been applied before. Panahi et al. [22] use proximity probes on a convex polygon in the plane to reconstruct it exactly. Goel et al. [10], reduce the approximation versions of several problems like diameter, farthest neighbors, discrete center, metric facility location, bottleneck matching and minimum weight matching to nearest neighbor queries. They sometimes require other primitives for their algorithms, for example computation of the minimum enclosing ball or a dynamic version of the approximate nearest-neighbor oracle. Similarly, the computation of the minimum spanning tree [13] can be done using nearest-neighbor queries (but the data structure needs to support deletions). For more details, see the survey by Indyk [16].

1.2 Our Contributions

In this paper we consider a number of problems on implicitly-represented point sets.

k-Center Clustering and the Greedy Permutation. Given a point set P, a *greedy permutation* (informally) is an ordering of the points of P: $p_1, \ldots, p_k, \ldots$, such that for any k, the set of points $\{p_1, \ldots, p_k\}$ is a $O(1)$-approximation to the optimal k-center clustering. This sequence arises in the k-center approximation of Gonzalez [11], and its properties were analyzed by Har-Peled and Mendel [15]. Specifically, if P can be covered by k balls of radius r_k, then the maximum distance of any point of P to its nearest neighbor in $\{p_1, \ldots, p_k\}$ is $O(r_k)$.

In Sect. 3, we show that under reasonable assumptions, in constant dimension, one can compute a permutation that is a bi-criteria approximation to the optimal k center clustering. More formally, we can compute a sequence of points from P, $p_1, p_2, \ldots$, such for any k, the radius of clustering using the centers in $\{p_1, \ldots, p_{ck}\}$ is an $O(1)$-approximation to the optimal k center clustering radius, where c is a constant depending only on the dimension. This result uses exact proximity queries, and only one query per sequence point generated. If the oracle answers $(1+\varepsilon)$-ANN queries only, then for any k, the permutation generated is competitive with the optimal k-center clustering, considering the first $O(k \log_{1/\varepsilon} \Phi)$ points in this permutation, where Φ is (roughly) the spread of the point set. The hidden constant factors grow exponentially in the dimension.

Approximate Convex-Hull Membership. Given a point set P in $\mathbb{R}^d$, consider the problem of deciding whether a given query point $q \in \mathbb{R}^d$ is inside its convex-hull $\mathcal{C} = \mathcal{CH}(P)$. The answer for such a query is ε-approximately correct if the answer is correct whenever the query point's distance from the boundary of $\mathcal{C}$ is at least $\varepsilon \cdot \text{diam}(\mathcal{C})$, i.e., the query point is sufficiently "inside" $\mathcal{C}$, or sufficiently "outside" $\mathcal{C}$. In Sect. 4, we show that, given a constant factor approximation to the diameter and an oracle for $(1+\varepsilon^2/c)$-ANN queries, for some sufficiently large constant c, it is possible to answer approximate convex-hull membership queries using $O(1/\varepsilon^2)$ proximity queries. Remarkably, the number of queries is independent of the dimension of the data. Moreover, the algorithm has only 1-sided error: the output is correct if the algorithm says the query point is outside the hull.

Our algorithm operates iteratively, by employing a gradient descent-like approach. It generates a sequence of points, all within the convex hull, that converges to the query point. When given full access to the point set, it is well known that for any $q \in \mathcal{CH}(P)$, such an iterative approach can be used to find a point q' which is a convex combination of $O(1/\varepsilon^2)$ points of P, and which is within distance $\varepsilon \cdot \text{diam}(\mathcal{C})$ of q. This fact is sometimes referred to as the approximate Carathéodory theorem [2], and it follows from the analysis of the Perceptron algorithm [21]. Similar techniques have been used before for a variety of other problems, and are sometimes referred to as the Frank-Wolfe algorithm. Clarkson provides a survey and some new results of this type [7]. A recent algorithm of this type is the work by Kalantari [17]. Our main new contribution for the convex-hull membership problem is showing that the iterative algorithm can be applied to implicit point sets using nearest-neighbor queries.

Balanced Proximity Clustering We study a problem that involves summarizing a point set in a way that preserves proximity information. Specifically, given a set P of n points in $\mathbb{R}^d$, and a parameter k, the objective is to select m centers from P, such that if we assign every point of P to its nearest center, no center has been selected by more than k points. This problem is related to topic of capacitated clustering from operations research [20].

In Sect. 5, we show that in the plane there exists such a clustering consisting of $O(n/k)$ such centers, and that in higher dimensions one can select $O((n/k) \log(n/k))$

centers (where the constant depends on the dimension). This result is not directly related to the other results in the paper.

Paper Organization In Sect. 2 we review some relevant work on k-center clustering. In Sect. 3 we provide our algorithm to compute an approximate k-center clustering. In Sect. 4 we show how we can decide approximately if a query point is within the convex hull of the given data points in a constant number of queries, where the constant depends on the degree of accuracy desired. Finally, in Sect. 5 we investigate balanced Voronoi partitions, which provides a density-based clustering of the data. Here we assume that all the data is known and the goal is to come up with a useful clustering that can help in proximity search queries.

2 Preliminaries

2.1 Notations

We use $g(n) = O_d(f(n))$ to denote that $g(n)/f(n)$ is bounded (for all n) by a constant that depends on the dimension d (usually exponentially). This is to distinguish from the alternative (standard) case where $g(n) = O(f(n))$ implies that this constant is independent of the dimension.

2.2 Background—*k*-center Clustering and the Greedy Permutation

The following is taken from [12, Chap. 4], and is provided here for the sake of completeness.

In the k-center clustering problem, a set $P \subseteq \mathbb{R}^d$ of n points is provided together with a parameter k. The objective is to find a set of k points, $C \subseteq P$, such that the maximum distance of a point in P to its closest point in C is minimized. Formally, define price $(C, P) = \max_{p \in P} \min_{c \in C} \|p - c\|$. Let C_{opt} denote the set of centers achieving this minimum. The k-center problem can be interpreted as the problem of computing the minimum radius, called the *k-center clustering radius*, such that it is possible to cover the points of P using k balls of this radius, each centered at one of the data points. It is known that k-center clustering is NP-Hard. Even in the plane, it is NP-Hard to approximate to within a factor of $(1 + \sqrt{7})/2 \approx 1.82$ [9].

2.2.1 The Greedy Clustering Algorithm

Gonzalez [11] provided a 2-approximation algorithm for k-center clustering. This algorithm, denoted by GreedyKCenter, repeatedly picks the point farthest away from the current set of centers and adds it to this set. Specifically, it starts by picking an arbitrary point, $\bar{c}_1$, and setting $C_1 = \{\bar{c}_1\}$. For $i > 1$, in the ith iteration, the algorithm computes

$$r_{i-1} = \text{price}\,(C_{i-1}, P) = \max_{p \in P} d\,(p, C_{i-1}) \tag{2.1}$$

and the point $\bar{c}_i$ that realizes it, where $d(p, C_{i-1}) = \min_{c \in C_{i-1}} \|p - c\|$. Next, the algorithm adds $\bar{c}_i$ to C_{i-1} to form the new set C_i. This process is repeated until k points have been collected.

If we run GreedyKCenter till it exhausts all the points of P (i.e., $k = n$), then this algorithm generates a permutation of P; that is, $\langle P \rangle = \langle \bar{c}_1, \ldots, \bar{c}_n \rangle$. We will refer to $\langle P \rangle$ as the *greedy permutation* of P. There is also an associated sequence of radii $\langle r_1, \ldots, r_n \rangle$, and the key property of the greedy permutation is that for each i with $1 \le i \le n$, all the points of P are within a distance at most r_i from the points of $C_i = \langle \bar{c}_1, \ldots, \bar{c}_i \rangle$. The greedy permutation has applications to packings, which we describe next.

Definition 2.1 A set $S \subseteq P$ is an *r-packing* for P if the following two properties hold:

(i) *Covering property*: All the points of P are within a distance at most r from the points of S.
(ii) *Separation property*: For any pair of distinct points $p, x \in S$, we have $\|p - x\| \ge r$.

(For most purposes, one can relax the separation property by requiring that the points of S be at distance $\Omega(r)$ from each other.)

Intuitively, an r-packing of a point set P is a compact representation of P at resolution r. Surprisingly, the greedy permutation of P provides us with such a representation for all resolutions.

Lemma 2.2 [12]

(A) *Let P be a set of n points in $\mathbb{R}^d$, and let its greedy permutation be $\langle \bar{c}_1, \ldots, \bar{c}_n \rangle$ with the associated sequence of radii $\langle r_1, \ldots, r_n \rangle$. For any i, $C_i = \langle \bar{c}_1, \ldots, \bar{c}_i \rangle$ is an r_i-packing of P. Furthermore, r_i is a 2-approximation for the optimal i-center clustering radius of P.*
(B) *For any k, let r_{opt}^k be the radius of the optimal k-center clustering of P. Then, for any constant c, there is a $k' = O_d(c^d k)$ such that, $r_{\mathrm{opt}}^{k'} \le r_{\mathrm{opt}}^k / c$.*
(C) *There exists an $n' = O_d(k/\varepsilon^d)$ such that computing the optimal k-center clustering of the first n' points of the greedy permutation, after appropriate rescaling, results in a $(1+\varepsilon)$-approximation to the optimal k-center clustering of P.*

2.3 Setup

The algorithms operate on a (not necessarily finite) point set P that is contained in a given *domain* $\mathcal{D} \subseteq \mathbb{R}^d$, which is a compact set (i.e., closed and bounded). Throughout we assume that $\mathcal{D}$ is the unit hypercube $[0, 1]^d$.

Given a query point $q \in [0, 1]^d$, let $\mathrm{nn}(q, P) = \arg\min_{p \in P} \|q - p\|$ denote the nearest neighbor (NN) of q. A point x is a $(1+\varepsilon)$-approximate nearest-neighbor (ANN) for q if $\|q - x\| \le (1+\varepsilon) \|q - \mathrm{nn}(q, P)\|$. We assume that the sole access to P is through "black-box" data structures T_{nn} and T_{ann}, which given a query point q, return the NN and ANN, respectively, to q in P.

3 Using Proximity Search to Compute k-center Clustering

The Problem Our purpose is to compute (or approximately compute) a k-center clustering of P using the given ANN black box, where k is a parameter between 1 and n.

3.1 Greedy Permutation via NN queries: GreedyPermutNN

Let q_0 be an arbitrary point in $\mathcal{D}$. Let ν_0 be its nearest-neighbor in P computed using the provided NN data structure T_{nn}. Let $b_0 = \text{ball}\,(q_0, \|q_0 - \nu_0\|)$ be the *open* ball of radius $\|q_0 - \nu_0\|$ centered at q_0. Finally, let $G_0 = \{\nu_0\}$, and let $\mathcal{D}_0 = \mathcal{D} \setminus b_0$.

In the ith iteration, for $i > 0$, let q_i be the point in $\mathcal{D}_{i-1}$ farthest away from G_{i-1}. Formally, this is the point in $\mathcal{D}_{i-1}$ that maximizes $d\,(q_i, G_{i-1})$, where $d\,(q, X) = \min_{c \in X} \|c - q\|$. Let $\nu_i = \text{nn}\,(q_i, P)$ denote the nearest-neighbor ν_i to q_i in P, computed using T_{nn}. Let

$$\begin{aligned} r_i &= d\,(q_i, G_{i-1})\,, \quad \ell_i = \|q_i - \nu_i\|\,, \quad b_i = \text{ball}\,(q_i, \ell_i)\,, \\ G_i &= G_{i-1} \cup \{\nu_i\}\,, \quad \text{and} \quad \mathcal{D}_i = \mathcal{D}_{i-1} \setminus b_i. \end{aligned}$$

Left to its own devices, this algorithm computes a sequence of not necessarily distinct points $\nu_0, \nu_1, \ldots$ of P. If P is not finite then this sequence may also have infinitely many distinct points. Furthermore, $\mathcal{D}_0 \supseteq \mathcal{D}_1 \supseteq \ldots$ is a sequence of outer approximations to P.

The execution of this algorithm is illustrated in Fig. 1, where P is shown as a squiggly curve, the points of q_i are shown as small circles, and the points of G_i are shown as small squares.

3.2 Analysis

Let $\mathcal{O} = \{o_1, \ldots, o_k\}$ be an optimal set of k centers of P. Formally, it is a set of k points in P that minimizes the quantity $r^k_{\text{opt}} = \max_{q \in P} d\,(q, \mathcal{O})$. Specifically, r^k_{opt} is the smallest possible radius such that k closed balls of that radius centered at points in P, cover P. Our claim is that after $O_d(k)$ iterations of the algorithm GreedyPermutNN, the sequence of points provides a similar quality clustering of P.

For any given point $p \in \mathbb{R}^d$ we can cover the sphere of directions centered at p by narrow cones of angular diameter at most $\pi/12$. We fix such a covering, denoting the set of cones by $\mathcal{C}_p$, and observe that the number of such cones is a constant c_d that depends on the dimension. Moreover, by simple translation we can transfer such a covering to be centered at any point $p' \in \mathbb{R}^d$.

Lemma 3.1 *For any $k \geq 1$, there is a number $\mu\,(k) = O_d(k)$ such that after $\mu\,(k)$ iterations, for any optimal center $o_i \in \mathcal{O}$, we have $d(o_i, G_{\mu(k)}) \leq 3r^k_{\text{opt}}$.*

Proof Fix a value of k, and let r_{opt} denote r^k_{opt}. If for any $j \leq \mu\,(k)$, we have $r_j \leq 3r_{\text{opt}}$ then all the points of $\mathcal{D}_{j-1} \supseteq P$ are within distance at most $3r_{\text{opt}}$ from G_j, and the claim trivially holds as $\mathcal{O} \subseteq P$.

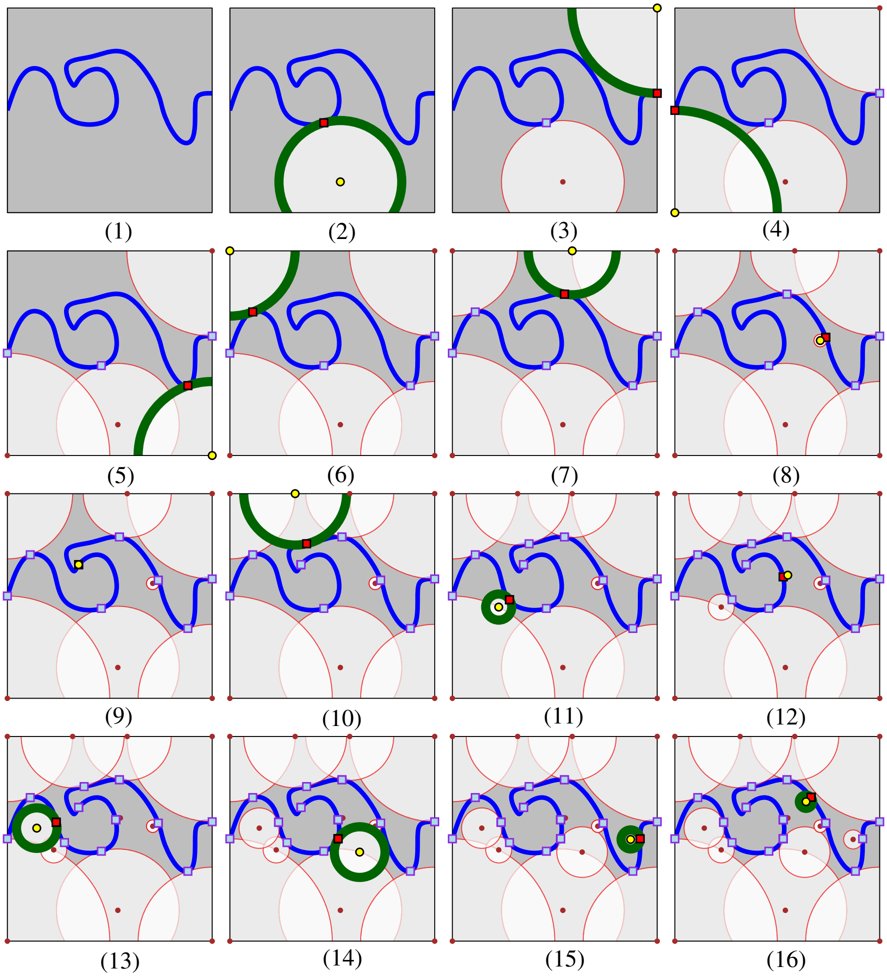

Fig. 1 An example of the execution of the algorithm GreedyPermutNN of Sect. 3.1

Let o be an optimal center and let P_o be the set of points of P that are closer to o than to any other center of $\mathcal{O}$, i.e., P_o is the cluster of o in the optimal clustering. Fix a cone ϕ from $\mathcal{C}_o$ (ϕ's apex is at o). Consider the output sequence $\nu_0, \nu_1, \ldots$, and the corresponding query sequence $q_0, q_1, \ldots$ computed by the algorithm. In the following, we use the property of the algorithm that $r_1 \geq r_2 \geq \cdots$, where $r_i = d(q_i, G_{i-1})$. A point q_j is *admissible* for o and ϕ if (i) $\nu_j \in P_o$, and (ii) $q_j \in \phi$ (in particular, ν_j is not necessarily in ϕ) (see Fig. 2).

We proceed to show that there are at most $O_d(1)$ admissible points for any fixed cone, which by a packing argument will imply the claim as every q_j is admissible for exactly one cone of one optimal center. Consider the induced subsequence of the output sequence restricted to the admissible points of ϕ: $\nu'_1, \nu'_2, \ldots$, and let $q'_1, q'_2, \ldots$ be the corresponding query points used by the algorithm. Formally, for a point ν'_i in

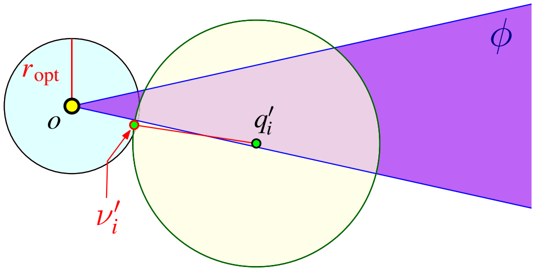

Fig. 2 Illustration for proof of lemma 3.1

this sequence, let iter (i) be the iteration of the algorithm it was created. Thus, for all i, we have $q'_i = q_{\mathrm{iter}(i)}$ and $\nu'_i = \nu_{\mathrm{iter}(i)}$.

Observe that $P_o \subseteq P \cap \mathrm{ball}\,(o, r_{\mathrm{opt}})$. This implies that

$$\|\nu'_j - o\| \le r_{\mathrm{opt}} \qquad \text{for all } j.$$

Let $\ell'_i = \|q'_i - \nu'_i\|$ and $r'_i = d(q'_i G_{\mathrm{iter}(i)-1})$. Observe that for $i > 1$, we have $\ell'_i \le r'_i \le \ell'_i+2r_{\mathrm{opt}}$, as $\nu'_{i-1} \in P_o$. Hence, if $\ell'_i \le r_{\mathrm{opt}}$, then $r'_i \le 3r_{\mathrm{opt}}$, and we are done. Thus, for any i, j, such that $1 < i < j$, we may assume that $\|q'_i - q'_j\| \ge \ell'_i > r_{\mathrm{opt}}$, as the algorithm carves out a ball of radius ℓ'_i around q'_i, and q'_j must be outside this ball.

By a standard packing argument, there can be only $O_d(1)$ points in the sequence $q'_2, q'_3, \ldots$ that are within distance at most $10r_{\mathrm{opt}}$ from o. If there are no points beyond this distance, we are done. Otherwise, let $i > 1$ be the minimum index, such that q'_i is at distance larger than $10r_{\mathrm{opt}}$ from o. We now prove that the points of $\phi \setminus \mathrm{ball}(q'_i, \ell'_i)$ are of two types—those contained within ball $(o, 3r_{\mathrm{opt}})$ and those that lie at distance greater than $(4/3)\ell'_i$ from o.

To see this, observe that since the angle of the cone, β, was chosen to be sufficiently small, ball (q'_i, ℓ'_i) splits ϕ into two components, where all the points in the component containing o are at distance less than $3r_{\mathrm{opt}}$ from o. The minimum distance to o (from a point in the component not containing o) is realized when q'_i is on the boundary of ϕ and o is on the boundary of ball (q'_i, ℓ'_i). Then the distance of any point of $\phi \setminus$ ball (q'_i, ℓ'_i) from o is at least $2\ell'_i \cos(\beta) \ge 2\ell'_i\sqrt{3/4} \ge 1.73\ell'_i$, as the opening angle of the cone is at most $\pi/12$ (see Fig. 3). The general case is somewhat more complicated as o might be within distance at most r_{opt} from the boundary of ball (q'_i, ℓ'_i), but as $\ell'_i \ge 9r_{\mathrm{opt}}$ by the triangle inequality, the claim still holds—we omit the tedious but straightforward calculations.

In particular, this implies that any later point q'_k in the sequence (i.e., $k > i$) is either one of the $O_d(1)$ close points to o, or it must greater than $(4/3)\ell'_i$ from o. In the latter case, by the triangle inequality $r'_k = d\,(q'_k, G_{\mathrm{iter}(k)-1}) \ge \|q'_k - o\| - r_{\mathrm{opt}} > \frac{4}{3}\ell'_i - r_{\mathrm{opt}}$. However, again by the triangle inequality $\ell'_i \ge 9r_{\mathrm{opt}}$, and so $r'_k > \frac{4}{3}\ell'_i - r_{\mathrm{opt}} = \ell'_i + (\frac{1}{3}\ell'_i - r_{\mathrm{opt}}) \ge \ell'_i + 2r_{\mathrm{opt}}$. As mentioned above, $\ell'_i + 2r_{\mathrm{opt}} \ge r'_i$ for any $i > 1$, and hence $r'_k > r'_i$, which is a contradiction as $r_2 \ge r_3 \ge \cdots$ (as r'_i appears before r'_k in this sequence). □

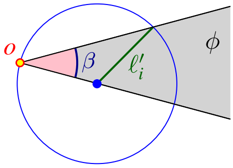

Fig. 3 Splitting of cone ϕ

The above lemma readily implies the following.

Theorem 3.2 *Let $P \subseteq \mathcal{D}$ be a given set of points in $\mathbb{R}^d$ (not necessarily finite), where $\mathcal{D}$ is a bounded set in $\mathbb{R}^d$. Furthermore, assume that P can be accessed only via a data structure T_{nn} that answers exact nearest-neighbor* (NN) *queries on P. The algorithm* GreedyPermutNN*, described in Sect. 3.1, computes a permutation $\langle v_0, \ldots\rangle$ of P, such that, for any $k > 0$, $P \subseteq \bigcup_{i=1}^{ck} \operatorname{ball}(v_i, r_{\mathrm{opt}}^k)$, where c is a constant (independent of k), and r_{opt}^k is the minimum radius of k balls (of the same radius) needed to cover P.*

The algorithm can be implemented, such that running it for i iterations, takes polynomial time in i and involves i calls to T_{nn}.

Proof Using Lemma 2.2 (B) in Lemma 3.1 implies the result. As for the running time, observe that each point q_i can be chosen from among the vertices of a suitable arrangement of $O_d(i)$ algebraic surfaces in $\mathbb{R}^{d+1}$, as follows. Ignoring the balls b_j for now, the point of the domain $\mathcal{D}$ that maximizes the distance to the closest point of G_{i-1} is a vertex of the Voronoi diagram of this set, trimmed to the boundary of $\mathcal{D}$ (assuming that $\mathcal{D}$ is the unit hypercube). It is well known that this Voronoi diagram is combinatorially equivalent to the lower envelope of a collection of cones in $\mathbb{R}^{d+1}$, where the apex of each cone is a point of G_{i-1}, and the axis of each cone is parallel to the $(d+1)$-st coordinate axis. Thinking of this coordinate axis as vertical, the highest vertex of this envelope when projected back down to $\mathbb{R}^d$ is the point that maximizes the distance to its closest point in G_{i-1}.

Now, in order to restrict q_i to lie outside of the union of balls $\{b_0, \ldots, b_{i-1}\}$, we trim the envelope by removing the union of vertical cylinders, each of whose base is one of these balls in $\mathbb{R}^d$. The desired point q_i is the vertical projection of the highest vertex of the trimmed envelope. This vertex can be found by computing the arrangement of these $i-1$ cones and $i-1$ cylinders in $\mathbb{R}^{d+1}$. This can be done in time that is polynomial in i and exponential in d through the use of standard methods, such as the Collins cylindrical algebraic decomposition [3]. □

Observation *If P is finite of size n, the above theorem implies that after $i \geq \mu(n)$ iterations, one can recover the entire point set P (as $r_{\mathrm{opt}}^n = 0$), where $\mu(n) = O_d(n)$ (see Lemma 3.1). Therefore $O_d(n)$ is an upper bound on the number of queries for any problem. Note however that in general our goal is to demonstrate when problems can be solved using a significantly smaller amount of* NN *queries.*

Observation 3.4 *The proof of Lemma 3.1 implies that for any $k \geq 1$, after $i > \mu(k)$ iterations of the algorithm, it must be the case that $r_i \leq 3r_{\mathrm{opt}}^k$.*

The above also implies an algorithm for approximating the diameter.

Lemma 3.5 *Consider the setting of Theorem 3.2 using an exact nearest-neighbor oracle. Suppose that the algorithm is run for* $m = \mu(1) + 1$ *iterations, and let* $\nu_1, \ldots, \nu_m$ *be the set of output centers and* $r_1, \ldots, r_m$ *be the corresponding distances. Then,* $\operatorname{diam}(P)/3 \le \max(\operatorname{diam}(\nu_1, \ldots, \nu_m), r_m) \le 3 \operatorname{diam}(P)$.

Proof Since the discrete one-center clustering radius lies in the interval $\left[\operatorname{diam}(P)/2, \operatorname{diam}(P)\right]$, Observation 3.4 implies that $r_m \le 3r_{\text{opt}} \le 3\operatorname{diam}(P)$. Moreover, each ν_i is in P, and so $\operatorname{diam}(\nu_1, \ldots, \nu_m) \le \operatorname{diam}(P)$. Thus the upper bound follows.

For the lower bound, observe that if $\operatorname{diam}(\nu_1, \ldots, \nu_m) < \operatorname{diam}(P)/3$, as well as $r_m < \operatorname{diam}(P)/3$, then it must be true that $P \subseteq \mathcal{D}_{m-1} \subseteq \bigcup_{j=1}^{l} \operatorname{ball}(\nu_j, r_m)$ has diameter less than $\operatorname{diam}(P)$, a contradiction. □

3.3 Using Approximate Nearest-Neighbor Search

If we are using an $(1+\varepsilon)$-ANN black box T_{ann} to implement the algorithm, one can no longer scoop away the ball $b_i = \operatorname{ball}(q_i, \|q_i - \nu_i\|)$ at the ith iteration, as it might contain some of the points of P. Instead, one has to be more conservative, and use the ball $b'_i = \operatorname{ball}(q_i, (1-\varepsilon)\|q_i - \nu_i\|)$ Now, we might need to perform several queries till the volume being scooped away is equivalent to a single exact query.

Specifically, let P be a finite set, and consider its associated *spread*, defined as

$$\Phi = \frac{\operatorname{diam}(\mathcal{D}_0)}{\min_{p,x\in P}\|p-x\|}.$$

It is no longer true, as in Lemma 3.1, that each cone would be visited only one time (or constant number of times). Instead, it is easy to verify that each query point in the cone, shrinks the diameter of the domain restricted to the cone by a factor of roughly ε. (To see why, consider a query point q_i at distance ℓ_i from ν_i, where $\ell_i \gg r^k_{\text{opt}}$. The removal of ball b_i leaves a subcone of length roughly $\varepsilon\ell_i$, from which q_{i+1} could then be chosen.) As such, at most $O\left(\log_{1/\varepsilon}\Phi\right)$ query points would be associated with each cone.

Corollary 3.6 *Consider the setting of Theorem 3.2, with the modification that we use a* $(1+\varepsilon)$-ANN *data structure* T_{ann} *to access* P*. Then, for any* k, $P \subseteq \bigcup_{i=1}^{f(k)} \operatorname{ball}(\nu_i, r^k_{\text{opt}})$*, where* $f(k) = O_d(k \log_{1/\varepsilon}\Phi)$.

4 Convex-Hull Membership Queries via Proximity Queries

Let P be a set of n points in $\mathbb{R}^d$, let Δ denote P's diameter, and let $\varepsilon > 0$ be a prespecified parameter. We assume that the value of Δ is known, although a constant approximation to this value is sufficient for our purposes.

Let $\mathcal{C} = \mathcal{CH}(P)$ denote P's convex hull. Given a query point $q \in \mathbb{R}^d$, the task at hand is to determine whether q is in $\mathcal{C}$. As before, we assume that our only access to P is via an ANN data structure. There are two possible outputs:

(A) In: if $q \in \mathcal{C}$, and
(B) Out: if q is outside $\mathcal{C}$ and at distance greater than $\varepsilon\Delta$ from the boundary of $\mathcal{C}$,

Either answer is acceptable if $q \notin \mathcal{C}$ but lies within distance $\varepsilon\Delta$ of $\partial\mathcal{C}$.

4.1 Convex Hull Membership Queries Using Exact Extremal Queries

We first solve the problem using exact extremal queries and then later show these queries can be answered approximately with ANN queries.

4.1.1 The Algorithm

We construct a sequence of points $p_0, p_1, \ldots$ each guaranteed to be in the convex hull $\mathcal{C}$ of P and use them to determine whether $q \in \mathcal{C}$. The algorithm is as follows. Let p_0 be an arbitrary point of P. For $i > 0$, in the ith iteration, the algorithm checks whether $\|p_{i-1} - q\| \le \varepsilon\Delta$, and if so the algorithm outputs In and stops.

Otherwise, consider the ray ψ_i emanating from p_{i-1} in the direction of q. The algorithm computes the point $z_i \in P$ that is extremal in the direction of this ray. If the projection z_i' of z_i on the line supporting ψ_i is between p_{i-1} and q, then q is outside the convex-hull $\mathcal{C}$, and the algorithm stops and returns Out. Otherwise, the algorithm sets p_i to be the projection of q on the line segment $p_{i-1}z_i$, and continues to the next iteration (see Figs. 4 and 5).

For a suitable constant c (see Lemma 4.2), if the algorithm does not terminate after c/ε^2 iterations, it stops and returns Out.

4.1.2 Analysis

Lemma 4.1 *If the algorithm runs for more than i iterations, then* $\mathsf{d}_i < \left(1 - \frac{\varepsilon^2}{2}\right)\mathsf{d}_{i-1}$, *where* $\mathsf{d}_i = \|q - p_i\|$.

Proof By construction, p_i, p_{i-1}, and q form a right angle triangle. The proof now follows by a direct trigonometric argument. Consider Fig. 5. We have the following properties:

(A) The triangles $\triangle p_{i-1} z_i' z_i$ and $\triangle p_{i-1} p_i q$ are similar.

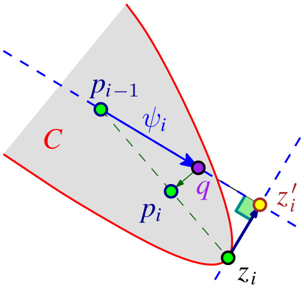

Fig. 4 Construction of p_i from p_{i-1}

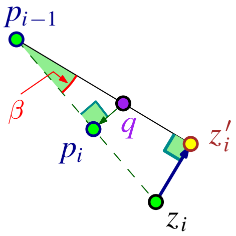

Fig. 5 Illustration for the proof of Lemma 4.1

(B) Because the algorithm has not terminated in the ith iteration, $\|p_{i-1} - q\| > \varepsilon\Delta$.
(C) The point q must be between p_{i-1} and z_i', as otherwise the algorithm would have terminated. Thus, $\|p_{i-1} - z_i'\| \geq \|p_{i-1} - q\| > \varepsilon\Delta$.
(D) We have $\|p_{i-1} - z_i\| \leq \Delta$, since both points are in $\mathcal{C}$.

We conclude that $\cos\beta = \dfrac{\|p_{i-1} - z_i'\|}{\|p_{i-1} - z_i\|} > \dfrac{\varepsilon\Delta}{\Delta} = \varepsilon$. Now, we have

$$\begin{aligned}\|q - p_i\| = \|q - p_{i-1}\| \sin\beta = \|q - p_{i-1}\| \sqrt{1 - \cos^2\beta} < \sqrt{1 - \varepsilon^2}\, \|q - p_{i-1}\| \\ < \left(1 - \frac{\varepsilon^2}{2}\right) \|q - p_{i-1}\|,\end{aligned}$$

since $(1 - \varepsilon^2/2)^2 > 1 - \varepsilon^2$. □

Lemma 4.2 *Either the algorithm stops within $O(1/\varepsilon^2)$ iterations with a correct answer, or the query point is outside $\mathcal{C}$ and lies at distance more than $\varepsilon\Delta$ from the boundary of the convex hull $\mathcal{C}$; in the latter case, since the algorithm says* Out *its output is correct.*

Proof If the algorithm stops before it completes the maximum number of iterations, it can be verified that the output is correct as there is an easy certificate for this in each of the possible cases.

Otherwise, suppose that the query point is either inside $\mathcal{C}$, or is outside $\mathcal{C}$ but within $\varepsilon\Delta$ of $\mathcal{C}$. We argue that this leads to a contradiction; thus the query point must be outside $\mathcal{C}$ and more than $\varepsilon\Delta$ far from $\mathcal{C}$ and the output of the algorithm is correct. Observe that d_i is a monotone decreasing quantity that starts at values $\leq \Delta$ (i.e, $\mathsf{d}_0 \leq \Delta$), since otherwise the algorithm terminates after the first iteration, as z_1' would be between q and p_0 on ψ_1; moreover, the output of the algorithm would be correct in this case.

Consider the jth *epoch* to be block of iterations of the algorithm, where $2^{-j}\Delta < \mathsf{d}_i \leq 2^{-j+1}\Delta$. Following the proof of Lemma 4.1, one observes that during the jth epoch one can set $\varepsilon_j = 1/2^j$ in place of ε, and using the argument it is easy to show that the jth epoch lasts $O(1/\varepsilon_j^2)$ iterations. By assumption, since the algorithm continued for the maximum number of iterations we have $\mathsf{d}_i > \varepsilon\Delta$, and so the maximum number of epochs is $\lceil \lg(1/\varepsilon) \rceil$. As such, the total number of iterations is $\sum_{j=1}^{\lceil \lg(1/\varepsilon) \rceil} O(1/\varepsilon_j^2) = O(1/\varepsilon^2)$. Since the algorithm did not stop, this is a contradiction. □

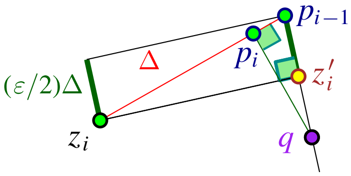

Fig. 6 Worst case if extremal queries are approximate

4.1.3 Approximate Extremal Queries

For our purposes, approximate extremal queries on P are sufficient.

Definition 4.3 A data structure provides ε *-approximate extremal queries* for P, if for any query unit vector v, it returns a point p, such that

$$\forall x \in P, \qquad \langle \mathsf{v}, x\rangle \le \langle \mathsf{v}, p\rangle + \varepsilon \cdot \operatorname{diam}(P),$$

where $\langle \mathsf{v}, x\rangle$ denotes the dot-product of v with x.

One can now modify the algorithm of Sect. 4.1.1 to use, say, $\varepsilon/4$-approximate extremal queries on P. Indeed, one modifies the algorithm so it stops only if z_i' is on the segment $p_{i-1}q$, and it is within distance more than $\varepsilon\Delta/4$ away from q. Otherwise the algorithm continues. It is straightforward but tedious to prove that the same algorithm performs asymptotically the same number of iterations (intuitively, all that happens is that the constants get slightly worse). The worst case as far progress in a single iteration is depicted in Fig. 6.

Lemma 4.4 *The algorithm of Sect. 4.1.1 can be modified to use $\varepsilon/4$-approximate extremal queries and output a correct answer after performing $O(1/\varepsilon^2)$ iterations.*

4.2 Convex-Hull Membership via ANN Queries

4.2.1 Approximate Extremal Queries via ANN *Queries*

The basic idea is to replace the extremal empty half-space query, by an ANN query. Specifically, a $(1+\delta)$-ANN query performed at q returns us a point p, such that

$$\forall x \in P, \qquad \|q - p\| \le (1+\delta)\,\|q - x\|\,.$$

Namely, ball $\left(q, \frac{\|q-p\|}{1+\delta}\right)$ does not contain any points of P. Locally, a ball looks like a halfspace, and so by taking the query point to be sufficiently far and the approximation parameter to be sufficiently small, the resulting empty ball and its associated ANN can be used as the answer to an extremal direction query.

4.2.2 The Modified Algorithm

Assume the algorithm is given a data structure T_{ann} that can answer $(1+\delta)$-ANN queries on P. Also assume that it is provided with an initial point $p_0 \in P$, and a value Δ' that is, say, a 2-approximation to $\Delta = \operatorname{diam}(P)$, that is $\Delta \le \Delta' \le 2\Delta$.

In the ith iteration, the algorithm considers (again) the ray ψ_i starting from p_i, in the direction of q. Let q_i be the point within distance, say,

$$\tau = c\Delta'/\varepsilon \tag{4.1}$$

from p_{i-1} along ψ_i, where c is an appropriate constant to be determined shortly. Next, let z_i be the $(1+\delta)$-ANN returned by T_{ann} for the query point q_i, where the value of δ would be specified shortly. The algorithm now continues as before, by setting p_i to be the nearest point on $p_{i-1}z_i$ to q. Naturally, if $\|q - p_i\|$ falls below $\varepsilon\Delta'/2$, the algorithm stops, and returns In, and otherwise the algorithm continues to the next iteration. As before, for a suitable constant c, if the algorithm does not terminate after c/ε^2 iterations, it stops and returns Out.

4.2.3 Analysis

Lemma 4.5 *Let* $0 < \varepsilon \leq 1$ *be a prespecified parameter, and let* $\delta = \varepsilon^2/(32-\varepsilon)^2 = O(\varepsilon^2)$. *Then, a* $(1+\delta)$-ANN *query done using* q_i (*as defined in Sect.* 4.2.2), *returns a point* z_i *which is a valid* ε*-approximate extremal query on* P*, in the direction of* ψ_i.

Proof Consider the extreme point $y_i \in P$ in the direction of ψ_i. Let y_i' be the projection of y_i to the segment $p_{i-1}q_i$, and let $\ell = \|q_i - y_i\|$ (see Fig. 7).

The $(1+\delta)$-ANN to q_i (i.e., the point z_i), must be inside the ball $b = \mathrm{ball}(q_i, (1+\delta)\ell)$, and let z_i' be its projection to the segment $p_{i-1}q_i$.

Now, if we interpret z_i as the returned answer for the approximate extremal query, then the error is the distance $\|z_i' - y_i'\|$, which is maximized if z_i' is as close to p_{i-1} as possible. In particular, let u be the point within distance $\|(1+\delta)\|$ from q_i along the segment $p_{i-1}q_i$. We then have that $\|z_i' - y_i'\| \leq h = \|u - y_i'\|$. Now, since $\|y_i' - y_i\| \leq \|p_{i-1} - y_i\| \leq \Delta'$, we have

$$\begin{aligned} h = \|u - y_i'\| &\leq (1+\delta)\ell - \|y_i' - q_i\| = (1+\delta)\ell - \sqrt{\ell^2 - \|y_i' - y_i{}^2\|} \\ &\leq (1+\delta)\ell - \sqrt{\ell^2 - (\Delta')^2} = \frac{(1+\delta)^2\ell^2 - \ell^2 + (\Delta')^2}{(1+\delta)\ell + \sqrt{\ell^2 - (\Delta')^2}} \leq \frac{(2\delta + \delta^2)\ell^2 + \left(\sqrt{\delta}\ell\right)^2}{\ell} \\ &\leq \frac{4\delta\ell^2}{\ell} = 4\delta\ell, \end{aligned}$$

since $\delta \leq 1$, and assuming that $\Delta' \leq \sqrt{\delta}\ell$. For our purposes, we need that $4\delta\ell \leq \varepsilon\Delta$. Both of these constraints translate to the inequalities, $\left(\frac{\Delta'}{\ell}^2\right) \leq \delta \leq \frac{\varepsilon\Delta}{4\ell}$. Observe that, by the triangle inequality, it follows that

$$\ell = \|q_i - y_i\| \leq \|q_i - p_{i-1}\| + \|p_{i-1} - y_i\| \leq \tau + \Delta.$$

A similar argument implies that $\ell \geq \tau - \Delta$. In particular, it is enough to satisfy the constraint $\left(\frac{\Delta'}{\tau - \Delta}^2\right) \leq \delta \leq \frac{\varepsilon\Delta}{4(\tau + \Delta)}$, which is satisfied if $\left(\frac{\Delta'}{\tau - \Delta'}^2\right) \leq \delta \leq \frac{\varepsilon\Delta'/2}{4(\tau + \Delta')}$, as $\Delta \leq \Delta' \leq 2\Delta$. Substituting the value of $\tau = c\Delta'/\varepsilon$, see Eq. (4.1), this is equivalent

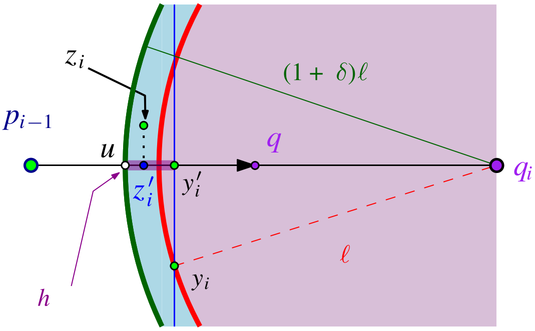

Fig. 7 Illustration of the proof of Lemma 4.5

to $\left(\frac{1}{c/\varepsilon-1}\right)^2 \leq \delta \leq \frac{\varepsilon/2}{4(c/\varepsilon+1)}$, which holds for $c = 32$, as can be easily verified, and setting $\delta = \varepsilon^2/(32-\varepsilon)^2 = O(\varepsilon^2)$. □

Theorem 4.6 *Given a set P of n points in $\mathbb{R}^d$, let $\varepsilon \in (0, 1]$ be a parameter, and let Δ' be a constant approximation to the diameter of P. Assume that you are given a data structure that can answer $(1+\delta)$-ANN queries on P, for $\delta = O(\varepsilon^2)$. Then, given a query point q, one can decide, by performing $O(1/\varepsilon^2)$ $(1+\delta)$-ANN queries whether q is inside the convex-hull $\mathcal{C} = \mathcal{CH}(P)$. Specifically, the algorithm returns*

- IN*: if $q \in \mathcal{C}$, and*
- OUT*: if q is more than $\varepsilon\Delta$ away from $\mathcal{C}$, where $\Delta = \mathrm{diam}(P)$.*

The algorithm is allowed to return either answer if $q \notin \mathcal{C}$, but the distance of q from the boundary of $\mathcal{C}$ is at most $\varepsilon\Delta$.

5 Density Clustering

5.1 Definition

Given a set P of n points in $\mathbb{R}^d$, and a parameter σ, with $1 \leq \sigma \leq n$, we are interested in computing a set $C \subseteq P$ of "centers", such that each center is assigned at most σ points, and the number of centers is (roughly) n/σ. In addition, we require that:

(A) A point of P is assigned to its nearest neighbor in C (i.e., C induces a *Voronoi partition* of P).
(B) The centers come from the original point set.

Intuitively, this clustering tries to capture the local density—in areas where the density is low, the clusters can be quite large (in the volume they occupy), but in regions with high density the clusters have to be tight and relatively "small".

Formally, given a set of centers C, and a center $c \in C$, its *cluster* is

$$P_c = \{p \in P \mid \|c - p\| < d(p, C \setminus \{c\})\},$$

where $d(p, X) = \min_{x \in X} \|p - x\|$ (and assuming for the sake of simplicity of exposition that all distances are distinct). The resulting *clustering* is $\Pi(P, C) =$

$\{P_c \mid c \in C\}$. A set of points P, and a set of centers $C \subseteq P$ is a *σ-density clustering* of P if for any $c \in C$, we have $|P_c| \le \sigma$. As mentioned, we want to compute a balanced partitioning, i.e., one where the number of centers is roughly n/σ. We show below that this is not always possible in high enough dimensions.

5.1.1 A Counterexample in High Dimension

Lemma 5.1 *For any integer $n > 0$, there exists a set P of n points in $\mathbb{R}^n$, such that for any $\sigma < n$, a σ-density clustering of P must use at least $n - \sigma + 1$ centers.*

Proof Let n be a parameter. For $i = 1, \ldots, n$, let $\ell_i = \sqrt{1 - 2^{-i-1}}$, and let p_i be a point of $\mathbb{R}^n$ that has zero in all coordinates except the ith one, where its value is ℓ_i. Let $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^n$. Now, $d_{i,j} = \|p_i - p_j\| = \sqrt{\ell_i^2 + \ell_j^2} = \sqrt{2 - 2^{-i-1} - 2^{-j-1}}$. For $i < j$ and $i' < j'$, we have

$$\begin{aligned} d_{i,j} < d_{i',j'} &\iff 2 - 2^{-i-1} - 2^{-j-1} < 2 - 2^{-i'-1} - 2^{-j'-1} \\ &\iff 2^{-i'} + 2^{-j'} < 2^{-i} + 2^{-j} \\ &\iff \begin{cases} i = i' \text{ and } j < j', \textit{or} \\ i < i'. \end{cases} \end{aligned}$$

That is, the distance of the ith point to all the following points $\mathsf{S}_{i+1} = \{p_{i+1}, \ldots, p_n\}$ is smaller than the distance between any pair of points of S_{i+1}.

Now, consider any set of centers $C \subseteq P$, and let $c = p_i$ be the point with the lowest index that belongs to C. Clearly, $P_c = (P \setminus C) \cup \{c\}$; that is, all the non-center points of P, get assigned by the clustering to c, implying the claim. □

5.2 Algorithms

5.2.1 Density Clustering via Nets

Lemma 5.2 *For any set of n points P in $\mathbb{R}^d$, and a parameter $\sigma < n$, there exists a σ-density clustering with $O_d\left(\frac{n}{\sigma} \log \frac{n}{\sigma}\right)$ centers.*

Proof Consider the hypercube $[-1, 1]^d$. Cover its outer faces (which are $(d-1)$-dimensional hypercubes) by a grid of side length $1/3\sqrt{d}$. Consider a cell C in this grid—it has diameter $\le 1/3$, and it is easy to verify that the cone $\phi = \{tp \mid p \in C, t \ge 0\}$ formed by the origin and C has angular diameter $< \pi/3$. This results in a set $\mathcal{C}$ of $N = O(d^d)$ cones covering $\mathbb{R}^d$.

Fix a cone $\phi \in \mathcal{C}$. For a point $p \in \mathbb{R}^d$, let ϕ_p denote the translation of ϕ such that p is its apex. Note that ϕ is formed by the intersection of $2(d-1)$ halfspaces. As such, the range space consisting of all ranges ϕ_p, such that $p \in \mathbb{R}^d$, has VC dimension at most $d' = O(d^2 \log d)$ [12, Theorem 5.22]. For a radius r and point p, let a *ϕ-slice* be the set $s_\phi(p, r) = \phi_p \cap \operatorname{ball}(p, r)$, i.e. the set formed by intersecting ϕ_p with a ball centered at p and of radius r. The range space of all ϕ-slices, $S_\phi = \{s_\phi(p, r) | p \in \mathbb{R}^d, r \ge 0\}$,

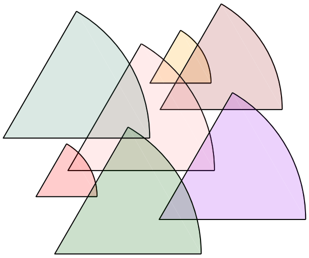

Fig. 8 ϕ-slices are pseudo-disks

has VC dimension $d'' = O(d+2+d') = O(d^2 \log d)$, since the VC dimension of balls in $\mathbb{R}^d$ is $d+2$, and one can combine range spaces as done above, see the book [12] for background on this.

Now, for $\varepsilon = (\sigma/N)/n = \sigma/(nN)$, consider an ε-net R of the point set P for ϕ-slices. The size of such a net is $|\mathsf{R}| = O\left((d''/\varepsilon)\log \varepsilon^{-1}\right) = O\left(\frac{nNd^2 \log d}{\sigma} \log \frac{nN}{\sigma}\right) = O\left(d^{O(d)} \frac{n}{\sigma} \log \frac{n}{\sigma}\right) = O_d\left(\frac{n}{\sigma} \log \frac{n}{\sigma}\right)$, by the ε-net theorem.

Consider a point $p \in P$ that is in R. Let ν_ϕ be the nearest point to p in the set $\{\mathsf{R} \setminus \{p\}\} \cap \phi_p$. The key observation is that any point in $P \cap \phi_p$ that is farther away from p than ν_ϕ, is closer to ν_ϕ than to p; that is, only points closer to p than ν_ϕ might be assigned to p in the Voronoi clustering. Since R is an ε-net for ϕ-slices, $s_\phi(p, \|p - \nu_\phi\|) = \phi_p \cap \text{ball}\,(p, \|p - \nu_\phi\|)$, contains at most $\varepsilon n = \sigma/N$ points of P. It follows that at most σ/N points of $P \cap \phi_p$ are assigned to the cluster associated with p. By summing over all N cones, at most $(\sigma/N)N = \sigma$ points are assigned to p, as desired. □

5.2.2 The Planar Case

Lemma 5.3 *For any set of n points P in $\mathbb{R}^2$, and a parameter σ with $1 \le \sigma \le n$, there exists a σ-density clustering with $O(n/\sigma)$ centers.*

Proof Consider the plane, and fix a cone $\phi \in \mathcal{C}$ as used in the proof of Lemma 5.2. For a point $p \in \mathbb{R}^2$, and a radius r, let ϕ_p denote the translated cone having p as an apex, and let $s_\phi(p, r) = \phi_p \cap \text{ball}\,(p, r)$ be a ϕ-slice induced by this cone. Consider the set of all possible ϕ-slices in the plane:

$$S_\phi = \{s_\phi(p, r) \mid p \in \mathbb{R}^2, r \ge 0\}.$$

It is easy to verify that the family S_ϕ behaves like a system of *pseudo-disks*; that is, the boundary of a pair of such regions intersects in at most two points (see Fig. 8). As such, the range space having P as the ground set, and S_ϕ as the set of possible ranges, has an ε-net of size $O(1/\varepsilon)$ [19]. Let $\varepsilon = (\sigma/N)/n$, as in the proof of Lemma 5.2, where $N = |\mathcal{C}|$. Computing such an ε-net, for every cone of $\mathcal{C}$, and taking their union results in the desired set of cluster centers. □

Acknowledgments N.K. would like to thank Anil Gannepalli for telling him about Atomic Force Microscopy. The full paper is available online [14]. Work on this paper by S. Har-Peled was partially supported by NSF AF Awards CCF-1421231, and CCF-1217462. Work on this paper by N. Kumar was partially supported by a NSF AF Award CCF-1217462 while the author was a student at UIUC, and by NSF Grant CCF-1161495 and a grant from DARPA while the author has been a postdoc at UCSB. Work on this paper by D. M. Mount was partially supported by NSF Award CCF-1117259 and ONR Award N00014-08-1-1015. Work on this paper by B. Raichel was partially supported by NSF AF Awards CCF-1421231, CCF-1217462, and the University of Illinois Graduate College Dissertation Completion Fellowship.

References

1. Andersson, L.-E., Stewart, N.F.: Introduction to the Mathematics of Subdivision Surfaces. SIAM, Philadelphia (2010)
2. Barman, S.: Approximating nash equilibria and dense bipartite subgraphs via an approximate version of Caratheodory's theorem. In: Proceedings of 47th Annual Symposium on the Theory of Computing (STOC), pp. 361–369 (2015)
3. Basu, S., Pollack, R., Roy, M.F.: Algorithms in Real Algebraic Geometry. Algorithms and Computation in Mathematics. Springer, Berlin (2006)
4. Binnig, G., Quate, C.F., Gerber, Ch.: Atomic force microscope. Phys. Rev. Lett. **56**, 930–933 (1986)
5. Blinn, J.F.: A generalization of algebraic surface drawing. ACM Trans. Graph. **1**, 235–256 (1982)
6. Boissonnat, J.-D., Guibas, L.J., Oudot, S.: Learning smooth shapes by probing. Comput. Geom. Theory Appl. **37**(1), 38–58 (2007)
7. Clarkson, K.L.: Coresets, sparse greedy approximation, and the Frank–Wolfe algorithm. ACM Trans. Algorithms **6**(4), 63 (2010)
8. Cole, R., Yap, C.K.: Shape from probing. J. Algorithms **8**(1), 19–38 (1987)
9. Feder, T., Greene, D. H.: Optimal algorithms for approximate clustering. In: Proceedings of 20th Annual ACM Aymposium on Theory of computing (STOC), pp. 434–444 (1988)
10. Goel, A., Indyk, P., Varadarajan, K. R.: Reductions among high dimensional proximity problems. In: Proceedings of 12th ACM–SIAM Symposium on Discrete Algorithms (SODA), pp. 769–778, (2001)
11. Gonzalez, T.: Clustering to minimize the maximum intercluster distance. Theor. Comput. Sci. **38**, 293–306 (1985)
12. Har-Peled, S.: Geometric Approximation Algorithms. Mathematical Surveys and Monographs, vol. 173. American Mathematical Society, Providence (2011)
13. Har-Peled, S., Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. Theory Comput. **8**, 321–350 (2012). Special issue in honor of Rajeev Motwani
14. Har-Peled, S., Kumar, N., Mount, D., Raichel, B.: Space exploration via proximity search. CoRR, http://arxiv.org/abs/1412.1398 (2014)
15. Har-Peled, S., Mendel, M.: Fast construction of nets in low dimensional metrics, and their applications. SIAM J. Comput. **35**(5), 1148–1184 (2006)
16. Indyk, P.: Nearest neighbors in high-dimensional spaces. In: Goodman, J.E., O'Rourke, J. (eds.) Handbook of Discrete and Computational Geometry, Chapter 39, 2nd edn, pp. 877–892. CRC Press, Boca Raton (2004)
17. Kalantari, B.: A characterization theorem and an algorithm for a convex hull problem. Ann. Oper. Res. **226**(1), 301–349 (2015)
18. Mandelbrot, B.B.: The Fractal Geometry of Nature. Macmillan, New York (1983)
19. Matoušek, J., Seidel, R., Welzl, E.: How to net a lot with little: small ε-nets for disks and halfspaces. In: Proceedings of 6th Annual ACM Symposium on Computational Geometry (SoCG), pp. 16–22 (1990)
20. Mulvey, J.M., Beck, M.P.: Solving capacitated clustering problems. Eur. J. Oper. Res. **18**, 339–348 (1984)
21. Novikoff, A.B.J.: On convergence proofs on perceptrons. Proc. Symp. Math. Theo. Automata **12**, 615–622 (1962)
22. Panahi, F., Adler, A., van der Stappen, A. F., Goldberg, K.: An efficient proximity probing algorithm for metrology. In: Proceedings of IEEE International Conference on Automation Science and Engineering (CASE), pp. 342–349 (2013)
23. Smelik, R. M., De Kraker, K. J., Groenewegen, S. A., Tutenel, T., Bidarra, R.: A survey of procedural methods for terrain modelling. In: Proceedings of the CASA. Workshop on 3D Advanced Media In Gaming and Simulation (2009)
24. Skiena, S.S.: Problems in geometric probing. Algorithmica **4**, 599–605 (1989)

25. Skiena, S.S.: Geometric reconstruction problems. In: Goodman, J.E., O'Rourke, J. (eds.) Handbook of Discrete and Computational Geometry, Chapter 26, pp. 481–490. CRC Press LLC, Boca Raton (1997)
26. Wikipedia. Atomic force microscopy—Wikipedia, The Free Encyclopedia (2014)