Space-Time Tradeoffs for Proximity Searching in Doubling Spaces

Sunil Arya[⋆1], David M. Mount[⋆⋆2], Antoine Vigneron[⋆⋆⋆3], and Jian Xia[†1]

[1] Department of Computer Science and Engineering,
The Hong Kong University of Science and Technology,
Clear Water Bay, Kowloon, Hong Kong
{arya,piper}@cse.ust.hk

[2] Department of Computer Science and Institute for Advanced Computer Studies,
University of Maryland, College Park, Maryland 20742
mount@cs.umd.edu

[3] INRA, UR341 Mathématiques et Informatique Appliquées,
78352 Jouy-en-Josas, France
antoine.vigneron@jouy.inra.fr

Abstract. We consider approximate nearest neighbor searching in metric spaces of constant doubling dimension. More formally, we are given a set S of n points and an error bound $\varepsilon > 0$. The objective is to build a data structure so that given any query point q in the space, it is possible to efficiently determine a point of S whose distance from q is within a factor of $(1+\varepsilon)$ of the distance between q and its nearest neighbor in S. In this paper we obtain the following space-time tradeoffs. Given a parameter $\gamma \in [2, 1/\varepsilon]$, we show how to construct a data structure of space $n\gamma^{O(\mathrm{dim})}\log(1/\varepsilon)$ space that can answer queries in time $O(\log(n\gamma)) + (1/(\varepsilon\gamma))^{O(\mathrm{dim})}$. This is the first result that offers space-time tradeoffs for approximate nearest neighbor queries in doubling spaces. At one extreme it nearly matches the best result currently known for doubling spaces, and at the other extreme it results in a data structure that can answer queries in time $O(\log(n/\varepsilon))$, which matches the best query times in Euclidean space. Our approach involves a novel generalization of the AVD data structure from Euclidean space to doubling space.

1 Introduction

Nearest neighbor searching is a fundamental problem in computational geometry with numerous applications in areas such as pattern recognition, information retrieval, machine learning, and robotics. The goal is to store a set S of n points so that, for any query point q, we can quickly return its nearest neighbor in S. As the problem is computationally difficult in most settings, researchers have

⋆ Research supported by RGC Grant HKUST6184/04E.

⋆⋆ Research supported in part by NSF grant CCF–0635099.

⋆⋆⋆ Research partially supported by a Marie Curie international reintegration grant.

† Research supported by RGC Grant HKUST6184/04E.

considered a variant in which it suffices to return an approximate answer. Given an error bound $\varepsilon > 0$, a point $p \in S$ is said to be an *ε-approximate nearest neighbor* (denoted *ε-NN*) of q if its distance from q is at most $(1+\varepsilon)$ times the distance between q and its nearest neighbor in S.

Approximate nearest neighbor searching has been studied extensively in Euclidean spaces. Recently there has been considerable interest in metric spaces as well. Data structures for proximity searching in metric spaces have been known for some time (see, e.g., [6, 10, 18]). Clarkson [8] and later Karger and Ruhl [14] introduced models designed to capture the sphere packing and local growth properties of low-dimensional Euclidean spaces. Much of the recent work has focused on metric spaces of low doubling dimension [4, 11]. The *doubling dimension* of a metric space is the minimum value ρ such that every ball in the space can be covered by 2^{ρ} balls of half the radius. This model was applied to various proximity problems by Krauthgamer, Lee, and co-authors [11, 15–17]. The results have been extended by Har-Peled and Mendel [13] and others [5, 9].

The results described in these papers on doubling spaces apply in the so called *black-box model*, in which points of the space can only be accessed through a black box that computes the distance between any two points in constant time. One of the advantages of this approach is that it relies on the barest set of assumptions, and so it is possible to obtain the conceptually simplest and most general algorithms. In this model, it is known that given a set of n points in a metric space of doubling dimension dim, ε-approximate nearest neighbor queries can be answered in time $O(\log n) + (1/\varepsilon)^{O(\dim)}$ using a data structure of linear space [9,13]. It is also observed in [13] that this result is optimal in the black-box model, as there is a lower bound of $\Omega(\log n) + (1/\varepsilon)^{\Omega(\dim)}$ on the query time in this model irrespective of the space used. (These asymptotic bounds, like ours, hide multiplicative factors that depend on the doubling dimension, except for the space bounds of Cole and Gottlieb [9], which are truly $O(n)$, irrespective of the dimension.)

Unfortunately, this query time compares unfavorably to the fastest query times known for Euclidean spaces. In Euclidean d-space, it is possible to answer ε-approximate nearest neighbor queries in time $O(\log(n/\varepsilon))$ and space roughly $O(n/\varepsilon^{d})$ through the use of a data structure called an *approximate Voronoi diagram* (or *AVD*) [1,2,12]. The difference in query time is quite significant, since in practice factors of the form $(1/\varepsilon)^{d}$ dominate the query time. It is also shown in [2] that space-time tradeoffs can be achieved. Thus, by limiting consideration to the purely implicit black-box model, simplicity and generality are achieved at the expense of efficiency and flexibility.

This raises the important question of whether it is possible to achieve results for approximate nearest neighbor searching that are comparable to the best results for Euclidean space in efficiency and flexibility, but in a model that provides the generality of metric spaces of low doubling dimension. The aforementioned lower bound indicates that this is not possible within the black-box model. In this paper we provide an affirmative answer to this question by strengthening the model slightly, which we call the *weakly explicit model*. In particular, we assume the doubling space is endowed with a *doubling oracle*, which, given any ball in the metric space returns in constant time a covering with a constant number of balls of half the radius (see Section 2).

Our approach is based on generalizing the AVD data structure to metric spaces in the weakly explicit model. We obtain the following space-time tradeoffs for approximate nearest neighbor searching in metric spaces of doubling dimension dim. Given a parameter $\gamma \in [2, 1/\varepsilon]$, we show how to construct AVDs of $n\gamma^{O(\mathrm{dim})} \log(1/\varepsilon)$ space that can answer ε-NN queries in time $O(\log(n\gamma)) + (1/(\varepsilon\gamma))^{O(\mathrm{dim})}$. This is the first result that offers space-time tradeoffs for approximate nearest neighbor queries in doubling spaces. At one extreme ($\gamma = 2$), we obtain an AVD of $O(n \log(1/\varepsilon))$ space that answers queries in time $O(\log n) + (1/\varepsilon)^{O(\mathrm{dim})}$. This result nearly matches the best result currently known for doubling spaces [9, 13], albeit in our stronger model. At the other extreme ($\gamma = 1/\varepsilon$), we obtain an AVD of $n(1/\varepsilon)^{O(\mathrm{dim})}$ space that can answer queries in time $O(\log(n/\varepsilon))$. This matches the query times for AVDs in Euclidean spaces, and overcomes the restrictive lower bound imposed by the black-box model for doubling spaces.

1.1 Overview of Techniques

In Euclidean space, the AVD is a quadtree-based partitioning of space into constant complexity cells, where each cell stores one or more representatives such that, given a query point q that lies within a cell, one of the associated representatives is an ε-NN of q. Queries are answered by first locating the cell that contains the query point and then scanning the list of stored representatives to find the closest one. The key idea underlying the construction of AVDs in Euclidean space is to partition space into cells, such that each cell enjoys certain *separation properties* with respect to the point set S. These separation properties assert that the region surrounding each cell is simple enough that we can answer ε-NN queries with the help of a small set of representatives. The construction is based on the *box-decomposition tree* (or the *compressed quadtree*), which yields a hierarchical partitioning of space into *fat* cells. The construction is bottom-up, first generating quadtree boxes and then building a tree structure over them.

In metric spaces we do not have the same explicit access to the ambient space's structure, and so we need a different approach. While similar in spirit, our generalization of AVDs to doubling metric spaces differs in the types of cells generated, the method used to generate these cells, and the separation properties they satisfy. It will be necessary to relax the AVD's partitioning of space to allow for a covering instead. We know of no analogous decomposition structure to the box-decomposition tree in doubling spaces, and so we have developed a hybrid construction, which is neither purely top-down nor bottom-up. Roughly speaking, the cells corresponding to all the nodes in the hierarchy that are in the vicinity of the point set S are generated right in the beginning. Next, for each such cell, we identify its children independently. We determine both the cells and the child-parent relationships between them on the basis of the well-separated pair decomposition [7, 13] of the point set. The resulting data structure is not a tree, but a rooted directed acyclic graph, which we call a *region-DAG*. The cells associated with the leaves of the region-DAG cover all of space and satisfy certain separation properties with respect to the point set S. This feature enables us to use region-DAGs for constructing AVDs in doubling spaces.

2 Preliminaries

We begin with some definitions. Let (M, d) be a metric space. We let $B(x, r)$ denote the closed ball of radius r centered at x, i.e., $B(x, r) = \{y \in M : d(x, y) \leqslant r\}$. For a ball b and any positive real η, we use ηb to denote the ball with the same center as b and whose radius is η times the radius of b, and $\bar{b}$ to denote the set of points that are not in b.

The *doubling dimension* of M, denoted $\dim(M)$, is the minimum value ρ such that every ball in M can be covered by 2^{ρ} balls of half the radius. When there is no ambiguity, we will write dim instead of $\dim(M)$. We say that M is a *doubling space* if it has constant doubling dimension.

Throughout this paper, we will assume that the metric space M is doubling. As mentioned earlier, our constructions will assume the existence of a *doubling oracle*, which given any ball b of radius r in M, returns in $2^{O(\dim(M))}$ time a set of $2^{O(\dim(M))}$ balls of radius $r/2$ covering b. Note that the centers of these balls are not necessarily in the input point set. We view the points (data, query, and covering-ball centers) as being drawn from some ambient metric space to which this oracle has access. This motivates our use of the term *weakly explicit* to describe this model.

A subset $S \subseteq M$ is defined to be an *r-net* of M if (i) every point of M is covered by a ball of radius r centered at some point of S and (ii) the pairwise distance between any two points of S is $\Omega(r)$. It is well-known that such nets always exist for any $r > 0$.

Throughout, we treat n, ε and γ as asymptotic quantities. The constant factors hidden by the $O(\cdot)$ notation are independent of n, ε and γ, but may depend on the doubling dimension.

2.1 The Well-Separated Pair Decomposition

We briefly review the notion of well-separated pair decomposition, as our constructions rely on it. Let S be a set of n points in the doubling space M. We say that two sets of points $X \subseteq S$ and $Y \subseteq S$ are *well-separated* if there exist two disjoint balls of radius r covering X and Y respectively, such that the distance between the centers of these balls is at least σr, where $\sigma \geqslant 2$ is a real parameter called the *separation factor*. We refer to (X, Y) as a *well-separated pair*. In Euclidean space, if we imagine joining the centers of these two balls by a line segment, the resulting geometric shape resembles a *dumbbell*. The balls are the heads of the dumbbell. The *length* of a dumbbell is defined as the distance between the centers of the balls.

Let x and y be two points in S. We say that a well-separated pair (X, Y) *contains* x if $x \in X \cup Y$, and we say that it *separates* x and y if $(x, y) \in (X \times Y) \cup (Y \times X)$. These notions can also be applied in a natural way to the dumbbell associated with a well-separated pair.

A *well-separated pair decomposition* (WSPD) of S is a set $\mathcal{P}_{S,\sigma} = \{\{X_1, Y_1\}, \ldots, \{X_m, Y_m\}\}$ of pairs of subsets of S such that (i) for $1 \leqslant i \leqslant m$, X_i and Y_i are well-separated, and (ii) for any distinct points $x, y \in S$, there exists a unique pair (X_i, Y_i) that separates x and y. Given any n-point set in constant-dimensional Euclidean space, Callahan and Kosaraju [7] showed that there exists a WSPD of linear size. This result was generalized to doubling spaces by Har-Peled and Mendel [13], who showed that the number of pairs in the WSPD of S is $\sigma^{O(\dim)}n$ and it can be constructed in $2^{O(\dim)}n \log n + \sigma^{O(\dim)}n$ time. For each pair, their construction also provides the corresponding dumbbell satisfying the separation criteria mentioned above. Furthermore, the centers of both the dumbbell heads are points of S.

The following preliminary lemma will be useful for us. It follows from the definition of well-separatedness and the triangle inequality.

Lemma 1. *Consider the WSPD of S with separation factor $\sigma \geqslant 16$. Consider the dumbbell for a pair $P = (X, Y)$ in this WSPD. Let x and y denote the centers of the dumbbell heads, and let $\ell = d(x, y)$ be the length of the dumbbell. Then for any $x' \in X$ and $y' \in Y$ we have $d(x, x') \leqslant \ell/16$ and $7\ell/8 \leqslant d(x', y') \leqslant 9\ell/8$.*

3 The Region-DAG

In this section, we describe our construction of the *region-DAG*, which can be viewed as a generalization of the box-decomposition tree [3] to doubling spaces. Our AVD construction in doubling spaces described in Section 4 will rely crucially on this data structure.

Let S be a set of n points in a doubling space (M, d). The *region-DAG* for S is a directed acyclic graph in which each node is associated with a region of space called a *cell*, which is the difference of two concentric balls, an *outer ball* and an (optional) *inner ball*. If the inner ball exists, its radius is at most half the radius of the outer ball. If a cell has no inner ball, we call it a *simple cell* (the corresponding node is called a *simple node*), otherwise we call it a *doughnut cell* (the corresponding node is called a *doughnut node*). Throughout this paper, for a simple node u, we let b_u denote the associated cell. The *size* of a cell (and the corresponding node) is defined to be the radius of its outer ball. If a cell contains no points of S, we say that it is *empty*, otherwise it is *nonempty*. If there is an edge from node u to node v, we say that v is a *child* of u. If a node has no children, it is called a *leaf*, otherwise it is an *internal node*.

Our construction of the region-DAG involves two parameters $\gamma \geqslant 2$ and $\beta \geqslant \gamma$. These parameters help to control the degree of separation enjoyed by the leaf cells with respect to the points of S. As we will see later, varying these parameters enables us to achieve space-time tradeoffs in our AVD constructions.

The key properties satisfied by the region-DAG are given below. We provide some intuition on how these properties aid in constructing AVDs in doubling spaces. Property (i) says that there is a node whose associated ball, which is called the *root ball*, contains the point set S close to its center. This property is useful for answering queries when the query point q lies outside the root ball. If, however, q lies inside the root ball, then we first find a leaf cell containing q. Such a leaf cell must exist because, by property (iii), the cell associated with any internal node is covered by the cells associated with its children. Property (iv) guarantees that we can find this leaf cell quickly (even though the depth of the region-DAG can be large). Property (ii) describes the separation properties satisfied by the leaves, which help in answering queries efficiently.

(i) There exists a node whose associated cell is a ball b, which is centered at a point of S and which satisfies $S \subseteq \frac{1}{\beta}b$. (We maintain a pointer to one node satisfying this property, which is called the *root* of the region-DAG. The cell associated with the root is called the *root ball*.)

(ii) There are two kinds of leaves, simple leaves and doughnut leaves, with the following separation properties. (See Figure 1.)

(a) Let ball b denote the cell associated with a simple leaf. Then either the ball γb is empty, or it contains one point of S, which is the center of b.

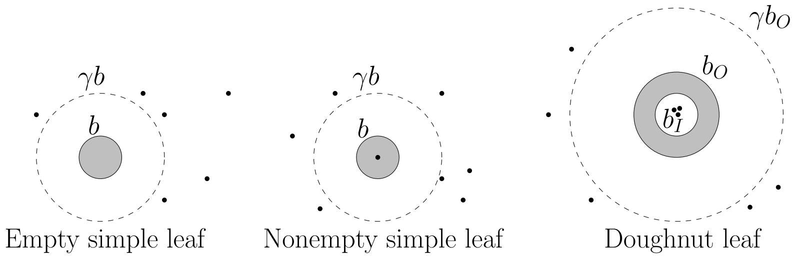

Fig. 1. Separation properties of leaf cells.

(b) Let b_O and b_I denote the outer and inner ball, respectively, of the cell associated with a doughnut leaf. Then $S \cap \gamma b_O \subseteq \left(\frac{1}{\beta}\right) b_I$. (Note that the doughnut cell $b_O \setminus b_I$ is empty.)

(iii) The cell associated with an internal node is always simple, and is covered by the cells associated with its children. More precisely, there are two kinds of internal nodes, *splitting nodes* and *shrinking nodes*, with the following properties. Let the cell associated with an internal node u be a ball b of radius r.

(a) If u is a splitting node, then it has a constant number of children (depending on the doubling dimension). Moreover, each child is simple, and its size is in $[r/64, r/2]$.

(b) If u is a shrinking node, then it has two children. One of these children is a doughnut leaf. The outer ball associated with this leaf is b, and the inner ball associated with it is a ball b', whose radius is at most $r/2$. (That is, the doughnut leaf cell is $b \setminus b'$.) The cell associated with the other child is a ball covering b', having radius at most $r/2$. We refer to the child that is a doughnut leaf as the *outer child* of u and refer to the other child as the *inner child* of u.

(iv) Let b denote the root ball defined in property (i). Given a point $q \in b$, we can find a leaf cell containing q in $O(\log(n\gamma))$ time.

It is clear from property (iii) that the size of a node is always smaller than that of its parent by a factor of at least two, except for a doughnut leaf, whose size is the same as that of its parent. It follows that the region-DAG has no cycle.

In Section 3.1, we will establish the following theorem, which shows that any set S in doubling space admits a region-DAG of size linear in n, and it can be constructed efficiently.

Theorem 1. *Let $\gamma \geqslant 2$ and $\beta \geqslant \gamma$ be two real parameters. Given a set S of n points in doubling space M, there exists a region-DAG of size $n\gamma^{O(\dim)} \log \beta$ satisfying all of the above properties. Furthermore, this structure can be constructed in time $O(n \log n) + n\gamma^{O(\dim)} \log^2 \beta$.*

3.1 Construction

Recall that our construction uses two parameters $\gamma \geqslant 2$ and $\beta \geqslant \gamma$ that determine the separation properties of the leaves with respect to the points of S. Before constructing the region-DAG, we first construct a WSPD for S using $\sigma = 16$. The number of pairs in the WSPD is $O(n)$ and the time to construct it is $O(n \log n)$. We associate each pair in this WSPD with several balls as follows. Let $x, y \in S$ denote the points at the centers of the heads of the dumbbell corresponding to a pair, and let $\ell = d(x, y)$ denote the length of this dumbbell. Then the associated balls are the balls of radius $2^i \ell$ centered at x and y, for all integers i such that $\left\lfloor \log\left(\frac{1}{c_1\beta}\right)\right\rfloor \leqslant i \leqslant \lceil \log(c_2\beta)\rceil$, where $c_1, c_2 \geqslant 1$ are suitable large positive constants. We will refer to these balls as *type-1* balls. We associate a unique node in the DAG with each distinct type-1 ball. Note that for this purpose, we treat any two type-1 balls as distinct if they have different centers or radii or are generated by different pairs in the WSPD. We will refer to these nodes as *type-1* nodes. Since there are $O(n)$ pairs in the WSPD and we generate $O(\log \beta)$ balls for each pair, the total number of type-1 nodes is $O(n \log \beta)$. Since there is a point of S at the center of each type-1 ball, these nodes are always nonempty. Besides the type-1 nodes, we will also create some new nodes in the DAG during the construction, which will always be empty (but not necessarily leaves). We will call them *type-2* nodes.

We process each type-1 node u as follows. Recall that b_u denotes the cell associated with u. We assume that b_u is a ball of radius r centered at a point $p \in S$. Roughly speaking, if all the points of $S \setminus \{p\}$ are very far from p, we will make u a leaf, and if all the points of $S \cap \gamma b_u$ are very close to p, we will make it a shrinking node. Otherwise, if there are points of S at intermediate distances (i.e., neither too far nor too close), then we will make u a splitting node. Since it is too time consuming to examine the points of S for the purpose of these tests, we will instead examine certain well-separated pairs containing p, which yield sufficient information on the position of the points.

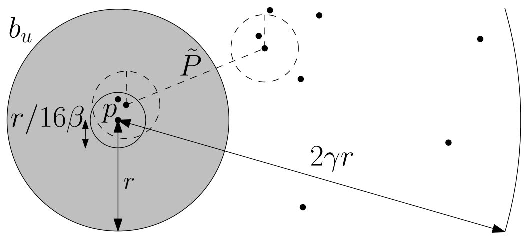

Fig. 2. Case where u is a splitting node.

We begin by finding the shortest dumbbell $\tilde{P}$ in the WSPD that contains p and has length at least $r/(16\beta)$. If $\tilde{P}$ has length at most $2\gamma r$, then u is made into a splitting node. (See Figure 2.) Otherwise, it is clear that there are no dumbbells containing p of length between $r/(16\beta)$ and $2\gamma r$. We then look for the longest dumbbell $\hat{P}$ containing p that has length at most $r/(16\beta)$. If we find such a dumbbell, then u is made into a shrinking node (See Figure 3), otherwise it is made into a simple leaf. We will establish property (ii.a) for the case when u is made into a simple leaf. After that we will describe how children are assigned when u is a shrinking and splitting node, respectively, and establish properties (ii) and (iii) for these cases.

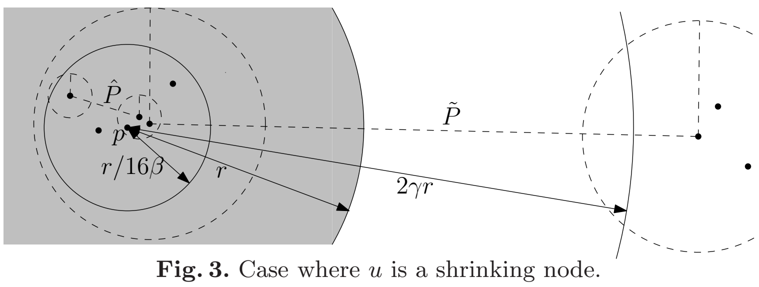

Fig. 3. Case where u is a shrinking node.

u is a leaf. We first consider the case when u is made into a leaf. Recall that in this case there are no dumbbells containing p of length at most $2\gamma r$. By Lemma 1, it follows that the distance between p and any other point of S is at least $(7/8)2\gamma r$. Thus, all the points of $S \setminus \{p\}$ lie outside the ball γb_u, which proves that u satisfies property (ii.a).

u is a shrinking node. We next consider the case when u is made into a shrinking node. Recall that in this case there are no dumbbells containing p that have length between $r/(16\beta)$ and $2\gamma r$. Recall also that we have already found the longest dumbbell $\hat{P}$ containing p that has length at most $r/(16\beta)$. We will assign two children to node u. Before describing these children, we first show that all the points of S in γb_u are very close to p. Let $\hat{\ell}$ denote the length of $\hat{P}$ and let $\hat{b}$ denote the ball $B(p, 2\hat{\ell})$. We claim that $S \cap \gamma b_u \subseteq \hat{b}$. To prove this claim, let x be any point of $S \cap \gamma b_u$. Since $d(p,x) \leqslant \gamma r$, it follows from Lemma 1 that the length ℓ of the dumbbell separating p and x is at most $8\gamma r/7 < 2\gamma r$. By our earlier remarks, there are no dumbbells containing p that have length between $r/(16\beta)$ and $2\gamma r$. Therefore, $\ell < r/(16\beta)$. Since $\hat{P}$ is the longest dumbbell containing p that has length at most $r/(16\beta)$, it follows that $\ell \leqslant \hat{\ell}$. Again, applying Lemma 1, it follows that $d(p,x) \leqslant 9\ell/8 \leqslant 9\hat{\ell}/8 \leqslant 2\hat{\ell}$. Thus, $x \in \hat{b}$, which proves the claim.

We can now describe the two children of u. For one of these children, we create a new node in the region-DAG whose associated cell is $b_u \setminus b'$, where b' is the ball $\beta\hat{b}$. We make this child a doughnut leaf whose only parent is u. By the claim above, $S \cap \gamma b_u \subseteq \left(\frac{1}{\beta}\right) b'$, and so (ii.b) holds. Further, since the radius of b' is $2\beta\hat{\ell}$ and $\hat{\ell} \leqslant r/(16\beta)$, it follows that the radius of b' does not exceed $r/8 < r/2$. Thus the condition given in (iii.b) for this child is satisfied.

We now describe the other child of u. Let p' denote the point of S at the center of that head of dumbbell $\hat{P}$ that contains p, and let b'' denote the ball $B(p', 2^{\lceil \log 3\beta \rceil}\hat{\ell})$. Assuming that $c_2 \geqslant 3$, it is easy to see that b'' is one of the type-1 balls associated with dumbbell $\hat{P}$ and so must have a unique corresponding node in the region-DAG. We make this type-1 node the second child of u. To establish property (iii.b), we need to show that b'' covers b' and has radius at most $r/2$. Clearly, the radius of b'' is at most $6\beta\hat{\ell}$. Since $\hat{\ell} \leqslant r/(16\beta)$, it follows that the radius of b'' is at most $3r/8 < r/2$. By Lemma 1 we have $d(p,p') \leqslant \hat{\ell}/16$. Using this fact and the triangle inequality, it follows that

$$b' = B(p, 2\beta\hat{\ell}) \subseteq B\left(p', \frac{\hat{\ell}}{16} + 2\beta\hat{\ell}\right) \subseteq B(p', 3\beta\hat{\ell}) \subseteq b''.$$

This establishes property (iii.b) and completes the description of the processing required for a shrinking node.

u is a splitting node. Finally, we consider the case when u is made into a splitting node. Recall that in this case there exists a dumbbell containing p that has length between $r/(16\beta)$ and $2\gamma r$. In the full version, we show that this fact implies that node u can be assigned $O(1)$ children, whose associated cells together cover the ball b_u and satisfy certain properties. Some of these children are of type-1 while the rest are newly created type-2 nodes associated with empty balls. Roughly speaking, the role of the type-1 children is to cover the parts of b_u that lie close to the points of S and the role of the type-2 children is to cover the parts of b_u that remain uncovered. More precisely, we have the following lemma.

Lemma 2. *There exists a set $\mathcal{B}_1$ of type-1 balls and a set $\mathcal{B}_2$ of type-2 balls such that (i) the total number of balls of $\mathcal{B}_1$ and $\mathcal{B}_2$ is $2^{O(\dim)}$, (ii) any ball of $\mathcal{B}_1 \cup \mathcal{B}_2$ has radius between $r/64$ and $r/2$, (iii) the balls of $\mathcal{B}_1$ and $\mathcal{B}_2$ together cover b_u, and (iv) for any ball $b \in \mathcal{B}_2$, there are no points of S in the ball $4b$.*

The nodes corresponding to the balls of $\mathcal{B}_1$ and $\mathcal{B}_2$ are made children of u. From the above lemma, it is easy to see that property (iii) holds for u.

It remains to discuss the processing for the type-2 children of u. Observe that we cannot make these nodes into leaves because their γ-expansion may contain points of S and so they do not necessarily satisfy property (ii.a). However, by Lemma 2(iv), we do know that a 4-expansion of any ball in $\mathcal{B}_2$ is free of points of S. To increase this expansion factor to γ, we proceed as follows for each type-2 child v of u. Let b_v denote the ball associated with v, and let r' denote its radius. Using the doubling oracle, in $2^{O(\dim)}$ time we can find $2^{O(\dim)}$ balls of radius $r'/2$ which overlap b_v. We create type-2 nodes for these balls and make them all children of v. We apply this procedure recursively to the children of v, terminating when we finally reach nodes of size $r'/2^{\lceil \log \gamma \rceil}$, which are made leaves of the region-DAG. It is easy to see that v is the root of a subtree with $\gamma^{O(\dim)}$ nodes and $\lceil \log \gamma \rceil + 1$ levels. All nodes in this subtree, except at the bottom level, are splitting nodes, and clearly satisfy property (iii). Applying Lemma 2(iv) and noting that the radii of the associated balls decrease by at least a factor of 2 as we descend this subtree, it is easy to show that the leaves satisfy property (ii.a).

Next we bound the size of the region-DAG.

Lemma 3. *The size of the region-DAG for an n-point set is $n\gamma^{O(\dim)} \log \beta$.*

Proof: Recall that the region-DAG has $O(n \log \beta)$ type-1 nodes. It is clear from our discussion above that a shrinking node acquires one child that is not of type-1 (this child is a doughnut leaf), and a type-1 splitting node acquires $\gamma^{O(\dim)}$ descendants that are not of type-1. Therefore, the size of the region-DAG is $n\gamma^{O(\dim)} \log \beta$. □

In the full version, we show that the region-DAG for an n-point set can be constructed in time $O(n \log n) + n\gamma^{O(\dim)} \log^2 \beta$, and also satisfies properties (i) and (iv). This completes the proof of Theorem 1.

4 Approximate Voronoi Diagrams

In this section we show how to construct approximate Voronoi diagrams in doubling spaces. Let (M, d) be a metric space with constant doubling dimension. Our main result is as follows.

Theorem 2. *Let S be a set of n points in M, and let $0 < \varepsilon \leqslant 1/2$ and $2 \leqslant \gamma \leqslant 1/\varepsilon$ be two real parameters. We can construct an AVD of $n\gamma^{O(\dim)} \log(1/\varepsilon)$ space that allows us to answer ε-approximate nearest neighbor queries in time $O(\log(n\gamma)) + (1/(\varepsilon\gamma))^{O(\dim)}$. The time to construct the AVD is $n(1/\varepsilon)^{O(\dim)} \log n$.*

Given the region-DAG, the proof of this theorem is straightforward by adapting the ideas used previously for Euclidean AVDs [1,2]. We sketch the main ideas briefly. Given the point set S and parameters $0 < \varepsilon \leqslant 1/2$ and $2 \leqslant \gamma \leqslant 1/\varepsilon$, we construct the region-DAG described in Theorem 1 for $\beta = 1/\varepsilon$. The number of nodes in the region-DAG is $n\gamma^{O(\dim)} \log(1/\varepsilon)$. Recall that the leaves of this structure satisfy certain separation properties with respect to S (region-DAG property (ii)). These properties enable us to answer queries efficiently with the help of a sparse set of representatives stored with each leaf. The following lemma provides a bound on the number of representatives we need to store with each cell. Given a set X of points and a point q, let $\mathrm{NN}_q(X)$ be the distance from q to its nearest neighbor in X. We say that a subset $R \subseteq S$ is an ε-representative set for a region w (with respect to S) if for any query point $q \in w$, we have $\mathrm{NN}_q(R) \leqslant (1+\varepsilon)\mathrm{NN}_q(S)$.

Lemma 4 (Concentric Ball Lemma). *Let $0 < \varepsilon \leqslant 1/2$ and $\gamma \geqslant 2$ be two real parameters. Let S be a set of points in M. Let b_1 and b_2 be two concentric balls of radius r and γr, respectively. Then there exist subsets $R_1, R_2 \subseteq S$ each consisting of at most $(1 + 1/(\varepsilon\gamma))^{O(\dim)}$ points such that (i) R_1 is an ε-representative set for b_1 with respect to $S \cap \overline{b_2}$, and (ii) R_2 is an ε-representative set for $\overline{b_2}$ with respect to $S \cap b_1$.*

In part (i), the set R_1 is formed by choosing an $(\varepsilon/2)$-NN of each point in an $(\varepsilon\gamma r/c)$-net for b_1, where c is a suitable constant. Applying the triangle inequality, it is easy to prove part (i). The proof of part (ii) is analogous. For each leaf cell u, we can use the above lemma to find an ε-representative set R for u with respect to S. We illustrate this for the case of a doughnut leaf cell u (the case where u is a simple leaf is easier and is omitted). Let b_O and b_I denote the outer and inner ball, respectively, for u. Recall that $u = b_O \setminus b_I$. It follows from region-DAG property (ii.b) that all the points of S are either outside γb_O or inside εb_I. By Lemma 4(i), there exists an ε-representative set R_1 of size $(1/(\varepsilon\gamma))^{O(\dim)}$ for u with respect to $S \cap \overline{\gamma b_O}$, and by Lemma 4(ii), there exists an ε-representative set R_2 of size $O(1)$ for u with respect to $S \cap \varepsilon b_I$. Clearly, the set $R = R_1 \cup R_2$ is an ε-representative set of size $(1/(\varepsilon\gamma))^{O(\dim)}$ for u with respect to S. We store the set R with u. The resulting AVD can be used for answering ε-NN queries as follows. Suppose that the query point q lies inside the root ball. By region-DAG property (iv), we can find a leaf that contains q in $O(\log(n\gamma))$ time. Then we return the closest representative stored with this leaf cell as the answer. The total query time is $O(\log(n\gamma)) + (1/(\varepsilon\gamma))^{O(\dim)}$. If q lies outside the root ball, a similar approach works using region-DAG property (i).

Consider next the space used by this AVD. A naive analysis of the space bound is provided by the product of the number of nodes in the region-DAG and the maximum number of representatives per cell, which yields a total of $n/\varepsilon^{O(\dim)}$. We can improve this bound significantly by applying a charging technique similar to that employed earlier in the Euclidean context [2]. This technique shows that although for a given cell, $(1/(\varepsilon\gamma))^{\Omega(\dim)}$ representatives may be needed, this cannot be the case for most of the cells. We omit the details due to lack of space. Applying this technique we can show that the total number of representatives summed over all the cells is $n\gamma^{O(\dim)}\log(1/\varepsilon)$, and they can be computed in time $n(1/\varepsilon)^{O(\dim)}\log n$. This completes the proof of Theorem 2.

References

1. S. Arya and T. Malamatos. Linear-size approximate Voronoi diagrams. In *Proc. 13th ACM-SIAM Sympos. Discrete Algorithms*, pages 147–155, 2002.
2. S. Arya, T. Malamatos, and D. M. Mount. Space-efficient approximate Voronoi diagrams. In *Proc. 34th Annu. ACM Sympos. Theory Comput.*, pages 721–730, 2002.
3. S. Arya, D. M. Mount, N. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. In *Proc. 5th ACM-SIAM Sympos. Discrete Algorithms*, pages 573–582, 1994.
4. P. Assouad. Plongements lipschitziens dans $\mathbb{R}^n$. *Bull. Soc. Math. France*, 111(4):429–448, 1983.
5. A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 97–104, 2006.
6. S. Brin. Near neighbor search in large metric spaces. In *Proc. 21st International Conf. on Very Large Data Bases*, pages 574–584, 1995.
7. 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.
8. K. L. Clarkson. Nearest neighbor queries in metric spaces. *Discrete Comput. Geom.*, 22(1):63–93, 1999.
9. R. Cole and L. Gottlieb. Searching dynamic point sets in spaces with bounded doubling dimension. In *Proc. 38th Annu. ACM Sympos. Theory Comput.*, pages 574–583, 2006.
10. C. D. Feustel and L. G. Shapiro. The nearest neighbor problem in an abstract metric space. *Pattern Recognition Letters*, 1(2):125–128, 1982.
11. A. Gupta, R. Krauthgamer, and J. R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Proc. 44th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 534–543, 2003.
12. S. Har-Peled. A replacement for Voronoi diagrams of near linear size. In *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 94–103, 2001.
13. S. Har-Peled and M. Mendel. Fast construction of nets in low dimensional metrics, and their applications. *SIAM J. Comput.*, 35(5):1148–1184, 2006.
14. D. R. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proc. 34th Annu. ACM Sympos. Theory Comput.*, pages 741–750, 2002.
15. R. Krauthgamer and J. R. Lee. Navigating nets: simple algorithms for proximity search. In *Proc. 15th ACM-SIAM Sympos. Discrete Algorithms*, pages 798–807, 2004.
16. R. Krauthgamer and J. R. Lee. The black-box complexity of nearest-neighbor search. *Theoretical Computer Science*, 348(2-3):262–276, 2005.
17. R. Krauthgamer and J. R. Lee. Algorithms on negatively curved spaces. In *Proc. 47th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 119–132, 2006.
18. P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, pages 311–321, 1993.