# Space-Time Tradeoffs for Approximate Spherical Range Counting*

Sunil Arya† Theocharis Malamatos‡ David M. Mount§

University of Maryland Technical Report CS–TR–4842 and UMIACS–TR–2006–57
November 2006

**Abstract**

We present space-time tradeoffs for approximate spherical range counting queries. Given a set $S$ of $n$ data points in $\mathbb{R}^d$ along with a positive approximation factor $\varepsilon$, the goal is to preprocess the points so that, given any Euclidean ball $B$, we can return the number of points of any subset of $S$ that contains all the points within a $(1-\varepsilon)$-factor contraction of $B$, but contains no points that lie outside a $(1+\varepsilon)$-factor expansion of $B$.

In many applications of range searching it is desirable to offer a tradeoff between space and query time. We present here the first such tradeoffs for approximate range counting queries. Given $0 < \varepsilon \leq 1/2$ and a parameter $\gamma$, where $2 \leq \gamma \leq 1/\varepsilon$, we show how to construct a data structure of space $O(n\gamma^d \log(1/\varepsilon))$ that allows us to answer $\varepsilon$-approximate spherical range counting queries in time $O(\log(n\gamma) + 1/(\varepsilon\gamma)^{d-1})$. The data structure can be built in time $O(n\gamma^d \log(n/\varepsilon)\log(1/\varepsilon))$. Here $n$, $\varepsilon$, and $\gamma$ are asymptotic quantities, and the dimension $d$ is assumed to be a fixed constant.

At one extreme (low space), this yields a data structure of space $O(n\log(1/\varepsilon))$ that can answer approximate range queries in time $O(\log n + (1/\varepsilon)^{d-1})$ which, up to a factor of $O(\log 1/\varepsilon)$ in space, matches the best known result for approximate spherical range counting queries. At the other extreme (high space), it yields a data structure of space $O((n/\varepsilon^d)\log(1/\varepsilon))$ that can answer queries in time $O(\log n + \log 1/\varepsilon)$. This is the fastest known query time for this problem.

Our approach is broadly based on methods developed for approximate Voronoi diagrams (AVDs), but it involves a number of significant extensions from the context of nearest neighbor searching to range searching. These include generalizing AVD node-separation properties from leaves to internal nodes of the tree and constructing efficient generator sets through a radial decomposition of space. We have also developed new arguments to analyze the time and space requirements in this more general setting.

*A preliminary version of this paper appeared in *Proc. 16th Annu. ACM-SIAM Sympos. Discrete Algorithms*, 2005, 535–544.

†Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. Supported by the Research Grants Council, Hong Kong, China (HKUST6080/01E). Email: arya@cs.ust.hk.

‡Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany. Email: tmalamat@mpi-sb.mpg.de.

§Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, Maryland 20742. Partially supported by the National Science Foundation under grant CCR-0098151. Email: mount@cs.umd.edu.

# 1 Introduction

Answering range counting queries is among the most fundamental problems in spatial information retrieval and computational geometry. The objective is to store a finite set of points so that it is possible to quickly count the points lying inside a given query range. Examples of ranges include rectangles, spheres, halfspaces, and simplices. In this paper we consider the weighted, counting version of the problem for spherical ranges. More generally, we assume each point stores an element of a semigroup, and the objective is to compute the semigroup sum of points in the range.

Range searching is a well studied problem in computational geometry, and nearly matching upper and lower bounds exist for many formulations. The most relevant case for us is that of halfspace range counting queries. Matoušek [17] has shown that in dimension $d$, with space $O(m)$ it is possible to achieve a query time of $O(n/m^{1/d}\log(m/n))$, where $n \leq m \leq n^d$. Nearly matching lower bounds have been given in the semigroup arithmetic model, first for simplex range searching by Chazelle [10] and later for halfspace range searching by Brönnimann, Chazelle and Pach [8].

*Spherical range searching* involves ranges that are (closed) Euclidean balls. It is well known that by projecting the points onto an appropriate paraboloid, spherical range searching can be reduced to halfspace searching in $\mathbb{R}^{d+1}$ [11]. Since a halfspace can be viewed as a sphere of infinite radius, lower bounds on halfspace range queries apply to spherical range queries as well. Unfortunately, the lower bounds on halfspace range searching destroy any reasonable hope of achieving the ideal of answering multidimensional spherical range queries in logarithmic query times using roughly linear storage space. This suggests the importance of pursuing approximation algorithms. Achieving speed-ups through approximation is reasonable in many applications in engineering and science where the data or ranges are imprecise [14, 16] and also in exact algorithms where approximations are used to obtain density estimates [18].

Let $b(p, r)$ denote a Euclidean ball in $\mathbb{R}^d$ centered at a point $p$ and having radius $r$. Given $\varepsilon > 0$ and the range $b(p, r)$, a set $S' \subseteq S$ is an admissible solution to an *$\varepsilon$-approximate range query* if it contains all the points of a $(1-\varepsilon)$-factor contraction of $b(p, r)$ and does not contain any point that lies outside a $(1+\varepsilon)$-factor expansion of this ball, that is,

$$S \cap b(p, r(1-\varepsilon)) \;\subseteq\; S' \;\subseteq\; S \cap b(p, r(1+\varepsilon)).$$

An $\varepsilon$-approximate range counting query returns the exact number (or semigroup sum) of the points in any such admissible solution. Note that the range is approximated, not the count. Although the error is two-sided, it is an easy matter to modify the values of $r$ and $\varepsilon$ to generate only one-sided errors. Arya and Mount [5] considered this problem and showed that with $O(n \log n)$ preprocessing time and $O(n)$ space, $\varepsilon$-approximate range counting queries can be answered in time $O(\log n + 1/\varepsilon^{d-1})$. (Their approach applied more generally to ranges that are fat, convex sets.)

In range searching it is often desirable to offer a tradeoff between space and query time. Unfortunately, Arya and Mount's results on approximate range searching do not admit any such tradeoffs. In this paper we remedy this situation by offering space-time tradeoffs for approximate spherical range counting queries. Let $S$ be a set of $n$ points in $\mathbb{R}^d$, and let $0 < \varepsilon \leq 1/2$ be the approximation bound. We take $n$ and $\varepsilon$ to be asymptotic quantities and assume that $d$ is a constant. Given a parameter $\gamma$, where $2 \leq \gamma \leq 1/\varepsilon$, we show how to construct a data structure of space $O(n\gamma^d \log(1/\varepsilon))$ that can answer $\varepsilon$-approximate range queries in time $O(\log(n\gamma) + 1/(\varepsilon\gamma)^{d-1})$. The data structure can be built in time $O(n\gamma^d \log(n/\varepsilon)\log(1/\varepsilon))$. Note that the construction time exceeds the space by a relatively modest factor of $O(\log(n/\varepsilon))$.

At one extreme ($\gamma = 2$) this yields a data structure of space $O(n \log(1/\varepsilon))$ that answers queries in time $O(\log n + (1/\varepsilon)^{d-1})$, thus matching the results of Arya and Mount [5] up to a factor of $\log(1/\varepsilon)$ in the space. At the other extreme ($\gamma = 1/\varepsilon$) this yields a data structure of space $O(n/(\varepsilon^d) \log(1/\varepsilon))$ that can answer queries in time $O(\log n + \log 1/\varepsilon)$. To our knowledge, these are the fastest query times for approximate spherical range searching. These results are summarized in Table 1.

Table 1: Summary of results for $\varepsilon$-approximate spherical range counting, with low-space ($\gamma = 2$) and high-space ($\gamma = 1/\varepsilon$). $O(\log 1/\varepsilon)$ factors have been omitted.

| | Query Time | Space |
|---|---|---|
| Low-Space | $\log n + 1/\varepsilon^{d-1}$ | $n$ |
| High-Space | $\log n$ | $n/\varepsilon^d$ |
| Tradeoff | $\log(n\gamma) + 1/(\varepsilon\gamma)^{d-1}$ | $n\gamma^d$ |

Our earlier work on linear space structures for approximate nearest neighbor queries [1] suggested the problem of achieving space-time tradeoffs for spherical range queries. Virtually all range searching structures operate by precomputing a number of *generators*, each of which is a subset of the point set. For counting queries we require that the intersection of any range with the point set can be expressed as a disjoint cover of an (ideally small) set of generators. The number of points (or generally the semigroup sum) for each generator is precomputed along with a data structure for computing the generators needed to answer a query.

The principal challenge in providing space-time tradeoffs in our case is determining a good way of defining generators. A natural approach is to subdivide the range space so that queries that are sufficiently similar (in a metric sense, depending on $\varepsilon$) can take advantage of this by using roughly the same generator sets. Our approach is to subdivide space hierarchically into hypercube cells using a quadtree-like decomposition, and each node is responsible for handling queries whose center lies inside the corresponding cell and whose radius is proportional to the cell size.

The most difficult aspect of this approach is achieving good bounds on the space required, particularly for point sets that are not uniformly distributed and may be highly clustered. Our approach is to adapt recent techniques derived for approximate nearest neighbor searching based on *approximate Voronoi diagram* (*AVDs*). Har-Peled [15] introduced the AVD of a point set $S$ as a quadtree-like partition of space into cells, such that all the points within each leaf cell have the same approximate nearest neighbor. Later results by Arya *et al.* [1, 2] generalized this by allowing each cell to store a small number of representative points and showed how this can lead to significant space improvements. An important element of their work is the notion of subdividing space hierarchically into hypercubes so that certain *separation properties* hold with respect to the point set. Such properties assert that the region surrounding each leaf cell of the decomposition is simple enough that all the information needed for answering queries can be encoded succinctly.

To achieve our results we have generalized a number of elements of the AVD construction. We show how to extend the separation properties for nearest neighbor searching (which is closely related to approximate range emptiness queries) to apply to arbitrary range counting queries. We have developed new arguments to analyze the total space requirements. Another new element is generator construction. Because ranges are spherical, rather than using quadtree cells themselves to define generators, we have developed a more efficient method based on a radial generalization

of a quadtree decomposition of space. This uses a polar representation of the points relative to the center of each cell. One of the appealing features of our overall approach to range searching is that it is based largely on quadtree decompositions and straightforward generalizations thereof. These data structures are easy to implement, and are not subject to numerical issues.

# 2 Preliminaries

Throughout we assume that the dimension $d$ is a fixed constant and treat $n$, $\varepsilon$ and $\gamma$ as asymptotic quantities. We assume that the set $S$ of points has been scaled and translated to lie within a ball of radius $\varepsilon/2$ placed at the center of the unit hypercube $[0,1]^d$. Let $x$ and $y$ denote any two points in $\mathbb{R}^d$. We use $\|xy\|$ to denote the Euclidean distance between $x$ and $y$ and $\overline{xy}$ to denote the segment joining $x$ and $y$. We denote by $b(x,r)$ a closed ball of radius $r$ centered at $x$, i.e, $b(x,r) = \{y : \|xy\| \leq r\}$. For a ball $b$ and any positive real $\gamma$, we use $\gamma b$ to denote the ball with the same center as $b$ and whose radius is $\gamma$ times the radius of $b$, and $\overline{b}$ to denote the set of points that are not in $b$. The concepts of well-separated pair decomposition and box-decomposition trees will play an important role in our constructions and analyses. We present these next.

## 2.1 Well-Separated Pair Decomposition

We say that two sets of points $X$ and $Y$ are *well-separated* if they can be enclosed within two disjoint $d$-dimensional balls of radius $r$, such that the distance between the centers of these balls is at least $\alpha r$, where $\alpha > 2$ is a real parameter called the *separation factor*. If we consider joining the centers of these two balls by a line segment, the resulting shape resembles a *dumbbell*. The balls are the *heads* of the dumbbell. A dumbbell *separates* two points $x$ and $y$ if $x$ is contained in one head and $y$ in the other. A *well-separated pair decomposition* (*WSPD*) of $S$ is a set $\mathcal{P}_{S,\alpha} = \{(X_1, Y_1), \cdots, (X_m, Y_m)\}$ of pairs of subsets of $S$ such that

(i) for $1 \leq i \leq m$, $X_i$ and $Y_i$ are well-separated, and

(ii) for any distinct points $x, y \in S$, there exists a unique pair $(X_i, Y_i)$ such that either $x \in X_i$ and $y \in Y_i$ or $x \in Y_i$ and $y \in X_i$. (We say that the pair $(X_i, Y_i)$ *separates* $x$ and $y$.)

Callahan and Kosaraju [9] have shown that we can construct a WSPD containing $O(\alpha^d n)$ pairs in time $O(n \log n + \alpha^d n)$. For each pair, their construction also provides the $d$-balls enclosing $X_i$ and $Y_i$ satisfying the separation criteria mentioned above.

## 2.2 BD- BBD-trees

A box-decomposition (BD) and balanced box-decomposition (BBD) trees are enhanced forms of the well known quadtree structure and its higher dimensional generalizations. We provide a short introduction here to these structures.

The well known *quadtree* structure (in its multidimensional form) is a hierarchical decomposition of space into hypercubes in $\mathbb{R}^d$. Starting with the unit hypercube $\mathbb{U}^d = [0,1]^d$, a *quadtree box* is any $d$-cube that can be obtained by a recursive splitting process that starts with $\mathbb{U}^d$ and generally splits an existing quadtree box by $d$ axis-orthogonal hyperplanes passing through its center into $2^d$ identical subcubes. Such a decomposition naturally defines a $2^d$-ary tree, such that each node is associated with a cube, called its *cell*. The *size* of a quadtree box is its side length. A nice property

of quadtree boxes is that any two quadtree boxes either have disjoint interiors or one is contained inside the other.

Quadtrees suffer from two shortcomings, which make them inappropriate for worst-case analysis. First, if points are densely clustered in some very small region of space, it is not possible to bound the number of quadtree splits needed to decompose the cluster by a function of $n$ alone. The box-decomposition (BD) tree [6] overcomes this by introducing an additional decomposition operation called shrinking. Second, if the point distribution is not uniform, the tree may not have logarithmic depth. The balanced box-decomposition (BBD) tree [7] extends the BD tree and remedies this problem by employing a balanced shrinking operation.

More formally, a *box-decomposition* (*BD*) *tree* of a set $S$ of $n$ points is a $2^d$-ary tree[1] that compactly represents a hierarchical decomposition of space [6]. A BD-tree cell is either a quadtree box or the set theoretic difference of two quadtree boxes, an *outer box* and an *inner box*. Cells of the former type are called *box cells* and cells of the latter type are called *doughnut cells*. As with the quadtree, the root node is associated with $\mathbb{U}^d$. When a cell contains at most one point of $S$ it is declared a leaf. There are two ways that an internal node can be decomposed. A *splitting operation* decomposes space into $2^d$ subcubes in exactly the same way as the quadtree. A *shrinking operation* decomposes a quadtree box $u$ into two children. The inner child cell is the smallest quadtree box $u'$ that contains $S \cap u$, and outer child cell is doughnut cell $u - u'$, which contains no points and hence is a leaf. The relevant properties of the BD tree are given below. Properties (i) and (iii) are proved in [6], and property (ii) is a simple generalization of (i).

(i) The BD tree has $O(n)$ nodes and can be constructed in time $O(n \log n)$.

(ii) A collection $\mathcal{C}$ of $n$ quadtree boxes can be stored in a BD tree with $O(n)$ nodes such that the subdivision induced by its leaves is a refinement of the subdivision induced by the quadtree boxes in $\mathcal{C}$. This can be constructed in time $O(n \log n)$.

(iii) The number of cells of the BD tree with pairwise disjoint interiors, each of size at least $s$, that intersect a ball of radius $r$ is at most $O((1 + r/s)^d)$.

The *balanced box-decomposition* (*BBD*) *tree* has many of the properties of a BD tree, but it has $O(\log n)$ depth. As in the BD tree, a cell of a BBD tree is a quadtree box or the difference of two quadtree boxes. Let $S$ be a set of $n$ points in $\mathbb{R}^d$, and let $T$ denote its BBD tree. A quadtree box (not necessarily in $T$) is *nonempty* if it contains at least one point of $S$. We will use the BBD tree for the following construction, called an *annulus cover*. Given two concentric balls $b_1$ and $b_2$, where $b_2$ is contained within $b_1$, and given $s > 0$ find the set $\mathcal{Q}$ of nonempty quadtree boxes of side length $s$ that overlap the annulus $b_1 - b_2$.

The properties of the BBD tree that are relevant to this paper are given below. Properties (i), (iii), and (iv) are proved in [7], property (ii) follows from the analysis in [5], and property (v) follows from the balancing aspect of the BBD tree. (In particular we use the fact that with every constant number of levels of descent of the BBD tree the number of points associated with each node decreases by a fixed constant. The straightforward proof is omitted.)

(i) $T$ has $O(n)$ nodes and $O(\log n)$ depth and can be constructed in time $O(n \log n)$.

(ii) It is possible to compute an annulus cover as described above in time $O(\log n + t)$, where $t$ is the number of nonempty quadtree boxes of size $s$ that intersect the larger annulus $2b_1 - b_2/2$.

[1]The tree defined in [6] is a binary version of this tree, but the assumption that all cells are hypercubes simplifies our presentation.

In the same time we can compute $|S \cap z|$ for each box $z$ in the cover. Since quadtree boxes exist only in discrete sizes (powers of 2) it is understood that if $s$ is not of this form this construction is performed for the next smaller power of 2.

(iii) The number of cells of the BBD tree with pairwise disjoint interiors, each of size at least $s$, that intersect a ball of radius $r$ is at most $O((1 + r/s)^d)$.

(iv) Using the BBD tree for a set $S$ of $n$ points, we can find the 1-approximate nearest neighbor of any query point $q$ in time $O(\log n)$.

(v) Let $1 \leq t \leq n$. Consider the set of nodes of the BBD tree that have at most $t$ points but whose parents have more than $t$ points. The size of such a set is at most $O(n/t)$.

For both BD trees and BBD trees, we define the *size* of a cell to be the size of its outer box. Throughout, for a cell $u$, we will use $s_u$ to denote its size. We let $r_u = s_u d/2$ and let $b_u$ denote the ball of radius $r_u$ whose center coincides with the center of $u$'s outer box (note that $u \subseteq b_u$). For $\gamma > 0$, we will make frequent use of the ball $\gamma b_u$, which we call the $\gamma$-*expansion* of $u$.

# 3 Separation Properties

Let $S$ be a set of $n$ points in $\mathbb{R}^d$ and let $0 < \varepsilon \leq 1/2$ and $2 \leq \gamma \leq 1/\varepsilon$ be two real parameters. In this section we show how to answer $\varepsilon$-approximate spherical range counting queries in time $O(\log(n\gamma) + 1/(\varepsilon\gamma)^{d-1})$ using a data structure of space $O(n\gamma^d \log(1/\varepsilon))$. We will show that the data structure can be constructed in time $O(n\gamma^d \log(n/\varepsilon) \log(1/\varepsilon))$.

Throughout we assume that a query ball $B$ is represented by its center point $q$ and **radius** $R$. Recall that $S$ has been scaled and translated to lie within a ball of radius $\varepsilon/2$ placed at the center of the unit hypercube $\mathbb{U}^d$. It follows that any query ball $B$ that is not contained within $\mathbb{U}^d$ can be answered trivially as follows. If the ball contains the center of $\mathbb{U}^d$, then we can treat all the points of $S$ as lying in the approximate range and return $n$ as the answer; otherwise, we return 0. Henceforth, we assume that the query ball $B$ is contained within $\mathbb{U}^d$.

During preprocessing, we first construct a BD tree based on the following lemma, which states that it is possible to construct a BD tree whose nodes enjoy certain separation properties with respect to the points of $S$. (See Fig. 1.) Intuitively, part (i) says that, for a leaf cell, if there are many points close to it, then they are sufficiently well clustered relative to their distance to the cell. As we will see, (i) is useful in answering approximate range queries when a range is sufficiently small. When a range is large, however, we need to use information associated with nodes higher up in the tree. Part (ii) describes the separation properties that aid in this case. This innovation is needed for range searching (versus nearest neighbor searching). The parameter $\gamma$ controls the separation and is used to control the space/time tradeoff.

**Lemma 1** (Separation Properties) *Let $S$ be a set of $n$ points in $\mathbb{R}^d$, and let $\gamma \geq 16$ and $0 < f \leq 1$ be two real parameters. It is possible to construct a BD tree $T$ with $O(nf\gamma^d \log \gamma)$ nodes satisfying the following properties. Let $u$ be a cell corresponding to a node of $T$.*

*(i) Suppose that $u$ is a leaf cell. Then there exists a ball $b'_u$ such that $|S \cap (\gamma b_u \setminus b'_u)| = O(1/f)$ and the ball $\gamma b'_u$ does not overlap $u$.*

*(ii) Suppose that $u$ is a box cell obtained by a shrink operation. Let $v$ denote the parent cell of $u$. Then either:*

*(a) there exists a ball $b'_v$ such that $|S \cap (\gamma b_v \setminus b'_v)| = O(1/f)$ and $\gamma b'_v$ does not overlap $v$, or*
*(b) $|S \cap (\gamma b_v \setminus 8b_u)| = O(1/f)$.*

*Moreover, in time $O(n\gamma^d \log(n\gamma) \log \gamma)$, we can construct $T$ with the following information stored at the nodes. For each leaf cell $u$, we store the ball $b'_u$ and $S \cap (\gamma b_u \setminus b'_u)$. For each box cell $u$, if it satisfies (ii.a) we store $S \cap (\gamma b_v \setminus 8b_u)$ and $|S \cap 8b_u|$, and if it satisfies (ii.b) we store the ball $b'_u$ and $S \cap (\gamma b_v \setminus b'_u)$.*

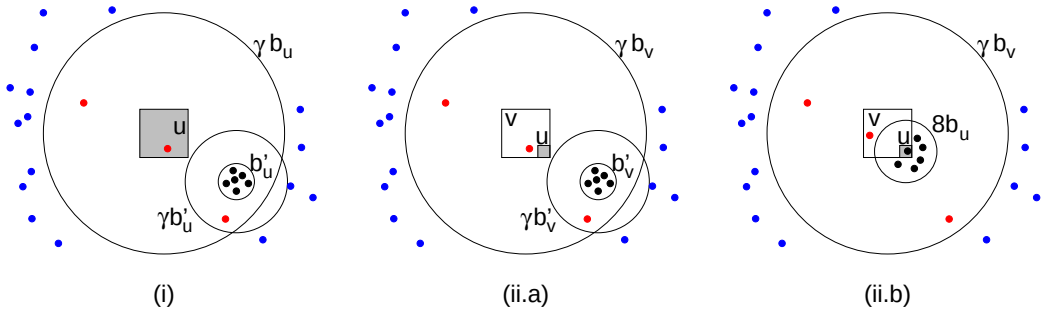


Fig. 1: The separation properties for cells of the BD tree.

Note that the cases in the lemma are neither mutually exclusive nor do they apply to all nodes (internal nodes obtained by splitting, in particular). When both conditions apply to a node, both pieces of auxiliary information are stored. In the lemma, the points of size $O(1/f)$ are called *pollutants*, since they do not satisfy the separation properties. They result as the leftovers of a sampling process and are needed to achieve our strongest bounds. Since they are handled by simple brute force in our query processing, the reader may find it easier to simply ignore them on first reading.

Before proving Lemma 1, we first prove the following lemma. It can be viewed as a "pollutant-free" version of the Lemma 1 with slightly stronger separation values. Part (i) of the lemma is similar to Lemma 4.5 in [1], while part (ii), which deals with separation properties involving the internal nodes, is new.

**Lemma 2** *Let $S$ be a set of $n$ points in $\mathbb{R}^d$ and let $\gamma \geq 16$. It is possible to construct a BD tree $T$ with $O(n\gamma^d \log \gamma)$ nodes satisfying the following properties. Let $u$ be a cell corresponding to a node of $T$.*

*(i) Suppose that $u$ is a leaf cell. Then either:*

*(a) $|S \cap u| = 1$ and $S \cap (\gamma b_u \setminus u) = \emptyset$, or*
*(b) there exists a ball $b'_u$ such that $S \cap (\gamma b_u \setminus b'_u) = \emptyset$, and the ball $8\gamma b'_u$ does not overlap $u$.*

*(ii) Suppose that $u$ is a box cell obtained by a shrink operation. Let $v$ denote the parent cell of $u$. Then either:*

*(a) $|S \cap v| = 1$ and $S \cap (\gamma b_v \setminus v) = \emptyset$, or*
*(b) there exists a ball $b'_v$ such that $S \cap (\gamma b_v \setminus b'_v) = \emptyset$, and the ball $\gamma b'_v$ does not overlap $v$, or*
*(c) $S \cap (\gamma b_v \setminus 4b_u) = \emptyset$.*

*Moreover, in time $O(n\gamma^d \log(n\gamma) \log \gamma)$, we can construct $T$ with the following additional information stored at the nodes. For each leaf cell $u$, if it satisfies (i.a) we store the point $S \cap u$, and if it satisfies (i.b) we store the ball $b'_u$. For each box cell $u$ satisfying (ii.b), we store the ball $b'_u$.*

**Proof**: In order to construct $T$, we first compute a well-separated pair decomposition for $S$ using separation factor 8. Let $\mathcal{D}$ be the resulting set of dumbbells. For each dumbbell $P \in \mathcal{D}$, we compute a set of quadtree boxes $\mathcal{C}_P$ as follows. Let $x$ and $y$ denote the centers of the two heads of $P$, $\ell = \|xy\|$, $z$ denote the center of segment $\overline{xy}$, and $\mathcal{B}_P$ denote the set of balls of radius $2^i\ell$ for $0 \leq i \leq \lceil \log \gamma + 5 \rceil$, centered at $z$. For a ball $b \in \mathcal{B}_P$, let $\mathcal{C}_b$ be the set of quadtree boxes overlapping $b$ that have size $r_b/(8\gamma d)$, where $r_b$ denotes the radius of $b$. Note that $|\mathcal{C}_b| = O(\gamma^d)$. Let $\mathcal{C}_P = \cup_{b \in \mathcal{B}_P} C_b$ and $\mathcal{C} = \cup_{P \in \mathcal{D}} \mathcal{C}_P$. Clearly $|\mathcal{C}_P| = O(\gamma^d \log \gamma)$ and $|\mathcal{C}| = O(n\gamma^d \log \gamma)$. Finally, we store all the boxes in $\mathcal{C}$ in a BD tree $T$. By BD property (ii), the number of nodes in $T$ is $O(|\mathcal{C}|) = O(n\gamma^d \log \gamma)$, and it can be constructed in time $O(|\mathcal{C}| \log |\mathcal{C}|) = O(n\gamma^d \log(n\gamma) \log \gamma)$. We will show below that $T$ satisfies properties (i) and (ii) in the statement of the lemma.

In addition to $T$, we also construct the BBD tree $T_b$ for $S$. This takes $O(n \log n)$ time. Let 1-*NN* denote the approximate nearest neighbor for $\varepsilon = 1$. With the help of $T_b$ we compute the 1-NN for the center of each leaf cell in $T$. By BBD property (iv), each 1-NN computation takes $O(\log n)$ time, so the total time for this step is $O((\gamma^d \log \gamma) n \log n)$.

We begin with a proof of assertion (i). Let $x$ be a 1-NN of the center of the leaf cell $u$. We distinguish three cases.

**Case 1:** ($x \notin 2\gamma b_u$) Since $x$ is a 1-NN, it follows that there is no point of $S$ in $\gamma b_u$ and so (i.b) trivially holds (we choose $b'_u$ to be the empty ball).

**Case 2:** ($x \in u$) We claim that (i.a) must hold; that is, besides $x$, there can be no other point of $S$ in $\gamma b_u$. For the sake of contradiction, suppose that there is a point $y \in S \cap \gamma b_u$, $y \neq x$. Consider the dumbbell $P \in \mathcal{D}$ separating points $x$ and $y$. Intuitively, our strategy is based on showing that $P$ should have forced $u$ to split. Let $x'$ and $y'$ denote the centers of the two heads of $P$, $\ell = \|x'y'\|$, and $z$ denote the center of segment $\overline{x'y'}$. Using the definition of well-separatedness and the triangle inequality, we obtain $\|zx\| \leq 5\ell/8$ and $\ell \leq 4\|xy\|/3$. Recall that $b(z, \ell) \in \mathcal{B}_P$. Since $b(z, \ell)$ overlaps $x$, there must be a quadtree box $\hat{u} \in \mathcal{C}_{b(z,\ell)}$ of size $s_{\hat{u}} \leq \ell/(8\gamma d)$ that contains $x$. Since $\ell \leq 4\|xy\|/3$ and $\|xy\| \leq \gamma s_u d$ (because both $x$ and $y$ are contained in $\gamma b_u$), we get $s_{\hat{u}} \leq s_u/6$. Thus $u$ should have been split further, which contradicts our assumption that $u$ is a leaf.

**Case 3:** ($x \in 2\gamma b_u \setminus u$) Define $b'_u$ to be the ball centered at $x$ such that $16\gamma b'_u$ just touches $u$. Let $p \in u$ be the point where $16\gamma b'_u$ touches $u$. For this choice of $b'_u$, we will show that (i.b) holds. Obviously $8\gamma b'_u$ does not overlap $u$, so we only need to show that $S \cap \gamma b_u \subseteq b'_u$. For the sake of contradiction, suppose there is a point $y \in S \cap (\gamma b_u \setminus b'_u)$. Consider the dumbbell $P \in \mathcal{D}$ separating points $x$ and $y$. As in Case 2, we will derive a contradiction by showing that $P$ should have forced $u$ to split.

To this end, let $x'$ and $y'$ denote the centers of the two heads of $P$, $\ell = \|x'y'\|$, $z$ denote the center of segment $\overline{x'y'}$, and $z'$ denote the center of segment $\overline{xy}$. Using the definition of well-separatedness and the triangle inequality, we get $4\|xy\|/5 \leq \ell \leq 4\|xy\|/3$, $\|zx\| \leq 5\ell/8$, and $\|zz'\| \leq \ell/8$.

We claim that the largest ball in $\mathcal{B}_P$ must overlap $p$. By our choice of $b'_u$, $\|xp\| = 16\gamma r$, where $r$ denotes the radius of $b'_u$. Since $y \notin b'_u$, $\|xy\| \geq r$. Thus $\|xp\| \leq 16\gamma\|xy\| \leq 20\gamma\ell$, where we have used the inequality $4\|xy\|/5 \leq \ell$ given above. By the triangle inequality, we get

$$\|zp\| \leq \|zx\| + \|xp\| \ \leq \ 5\ell/8 + 20\gamma\ell.$$

Recalling that the largest ball in $\mathcal{B}_P$ has radius at least $32\gamma\ell$, it follows that this ball must overlap $p$.

Let $b$ the smallest ball in $\mathcal{B}_P$ that overlaps $p$. We now consider two subcases and show that both lead to a contradiction. If $b = b(z, \ell)$ (recall this is the smallest ball in $\mathcal{B}_P$), then there must be a quadtree box $\hat{u} \in \mathcal{C}_b$ of size $s_{\hat{u}} \leq \ell/(8\gamma d)$ that contains $p$. Since $\ell \leq 4\|xy\|/3$ and $\|xy\| \leq 2\gamma s_u d$ (because both $x$ and $y$ are contained in $2\gamma b_u$), we get $s_{\hat{u}} \leq s_u/3$. Thus $u$ should have been split further, a contradiction.

Otherwise if $b \neq b(z, \ell)$, then it is clear that $b$ has radius $r_b$ satisfying $r_b \leq 2\|zp\|$ (because the radii of the balls in $\mathcal{B}_P$ differ by a factor of 2). Since $x$ and $y$ are both contained in $2\gamma b_u$, by convexity, the midpoint $z'$ of segment $\overline{xy}$ also lies in $2\gamma b_u$. By the triangle inequality, $\|zp\| \leq \|zz'\| + \|z'p\|$. Since $\|zz'\| \leq \ell/8, \ell \leq 4\|xy\|/3$, and $\|xy\| \leq 2\gamma s_u d$, we get $\|zz'\| \leq \gamma s_u d/3$. Thus, $\|zp\| \leq \gamma s_u d/3 + 2\gamma s_u d \leq 7\gamma s_u d/3$. By our construction, there must be a quadtree box $\hat{u} \in \mathcal{C}_b$ that overlaps $p$ having size $s_{\hat{u}} \leq r_b/(8\gamma d) \leq 2\|zp\|/(8\gamma d) \leq 7s_u/12$. Thus $u$ should have been split further, a contradiction. This completes the proof of (i).

Now let us prove assertion (ii). Since $u$ is obtained from $v$ by a shrink operation, clearly $w = v \setminus u$ is a doughnut leaf cell. Hence we can apply part (i) of this lemma to $w$. First, suppose that $w$ satisfies (i.a); that is, $|S \cap w| = 1$ and $S \cap (\gamma b_v \setminus w) = \emptyset$ (here we have replaced $b_w$ by $b_v$ as $b_v = b_w$). It follows that $|S \cap v| = 1$ and $S \cap (\gamma b_v \setminus v) = \emptyset$, and so (ii.a) holds.

Next suppose that $w$ satisfies (i.b); that is, there exists a ball $b'_w$ such that $S \cap \gamma b_v \subseteq b'_w$ and the ball $8\gamma b'_w$ does not overlap $w$. It follows that there must be a point $p$ on the boundary of $u$ such that $p \notin 8\gamma b'_w$. Let $x$ denote the center of $b'_w$ and let $r$ be its radius. Then $\|xp\| \geq 8\gamma r$.

We distinguish two cases, depending on whether $x \in (7/2)b_u$ or not. If $x \in (7/2)b_u$ then we claim that $b'_w \subseteq 4b_u$ (and so (ii.c) holds). To prove this claim, let $o$ denote the center of $u$. By the triangle inequality, $\|xp\| \leq \|xo\| + \|op\| \leq 7s_u d/4 + s_u d/2 = 9s_u d/4$. Since $\|xp\| \geq 8\gamma r$ and $\gamma \geq 16$, it follows that $r \leq \|xp\|/128 \leq 9s_u d/512$. Thus, the distance of a point in $b'_w$ from $o$ is at most $\|xo\| + r \leq 7s_u d/4 + 9s_u d/512 \leq 2s_u d$. Hence $b'_w \subseteq 4b_u$ as desired.

Otherwise, if $x \notin (7/2)b_u$ then we set $b'_v = b'_w$ and claim that (ii.b) holds for this choice of $b'_v$. In other words, we need to show that $\gamma b'_v$ does not overlap $v$. Recall that $8\gamma b'_v$ does not overlap $w$, so it suffices to show that $\gamma b'_v$ does not overlap $u$. Let $p'$ be any point in $u$. By the triangle inequality, $\|xp\| \geq \|xo\| - \|op\| \geq 7s_u d/4 - s_u d/2 = 5s_u d/4$. Also, $\|pp'\| \leq s_u d$. Again, by the triangle inequality, $\|xp'\| \geq \|xp\| - \|pp'\|$. Thus $\|xp'\|/\|xp\| \geq 1 - \|pp'\|/\|xp\|$. Using the above inequalities on $\|xp\|$ and $\|pp'\|$, we obtain $\|xp'\|/\|xp\| \geq 1/5$. Recalling that the radius $r$ of $b'_v$ is at most $\|xp\|/(8\gamma)$, it is clear that $\gamma b'_v$ does not contain $p'$. This completes the proof of (ii). □

With the use of Lemma 2 we can now provide the proof of Lemma 1. The lemma follows easily proved by combining Lemma 2 with a sampling procedure based on the BBD tree as described in [2]. This technique yields a BD tree with much fewer cells enjoying only slightly weaker separation properties. The proof is similar to Lemma 4 in [2]. We briefly sketch it here focusing on the construction time.

**Proof**: (Of Lemma 1) First we construct the BBD tree $T_b$ for the set $S$ of $n$ points. This takes $O(n \log n)$ time. Let $\mathcal{N}$ be the set of nodes of $T_b$ that contain at least one and at most $1/f$ pollutant points (of $S$), but whose parents contain more than $1/f$ pollutants. By BBD property (v), $|\mathcal{N}|$ is $O(nf)$. Let $T'_b$ be the truncated BBD tree whose leaves are the nodes in $\mathcal{N}$. Let $\mathcal{X}$ be the cells corresponding to the nodes in $\mathcal{N}$. Let $S' \subseteq S$ be the set of points obtained by sampling one point arbitrarily from each cell in $\mathcal{X}$. We construct the BD tree $T$ described in Lemma 2 for $S'$ but using the value $2\gamma$ in place of $\gamma$ in the lemma. We also store the information with the nodes as described in the lemma. Since $|S'| = O(nf)$, the number of nodes in $T$ is $O(nf\gamma^d \log \gamma)$ and the time to construct it is $O(nf\gamma^d \log(n\gamma) \log \gamma)$.

We claim that the nodes of $T$ satisfy properties (i) and (ii). Suppose first $u$ is a leaf cell satisfying Lemma 2(i.b). That is, there exists a ball $b''_u$ such that $S' \cap 2\gamma b_u \subseteq b''_u$ and the ball $16\gamma b''_u$ does not overlap $u$. Define $b'_u = 2b''_u$. It follows that $8\gamma b'_u$ does not overlap $u$. By using standard BBD tree searching techniques on $T'_b$, we can find the set $\mathcal{X}' \subseteq \mathcal{X}$ of cells that overlap $\gamma b_u \setminus b'_u$ in time $O(|\mathcal{X}'| \log n)$ time [5]. Recall that $S'$ includes one point of $S$ from each cell in $\mathcal{X}'$ and there is no point of $S'$ in the region $2\gamma b_u \setminus b'_u/2$. It follows that each cell in $\mathcal{X}'$ overlaps the boundaries of both $2\gamma b_u$ and $\gamma b_u$ or overlaps the boundaries of both $b'_u$ and $b'_u/2$. By BBD property (iii), $|\mathcal{X}'| = O(1)$ and so the time to find $\mathcal{X}'$ is $O(\log n)$. By scanning the points of $S$ in each cell in $\mathcal{X}'$, we can identify those points that are contained in $\gamma b_u \setminus b'_u$. Since each cell in $\mathcal{X}'$ has at most $O(1/f)$ pollutants, it follows that the number of such points is $O(1/f)$. Thus $u$ satisfies (i). Similarly, we can establish (i) for leaves satisfying Lemma 2(i.a).

Our discussion shows that in $O(\log n + 1/f)$ time we can compute $S \cap (\gamma b_u \setminus b'_u)$ and $b'_u$ for each leaf cell $u$. Summed over all leaf cells, the time for this computation is therefore $O(n\gamma^d \log n \log \gamma)$. We can prove (ii) in essentially the same way and show that the processing time for a box cell is the same as for a leaf cell. (We omit the straightforward details.) Finally, combining the time for computing $T$ and $T_b$, with the time for computing the required information for the nodes of $T$, we obtain a total construction time as stated in the lemma. This completes the proof. □

It will be convenient to view the BD tree described in Lemma 1 as a collection of three types of cells as described below. (This is not a pure classification, since cells may be of more than one type.) Cells of type-2 and type-3 satisfy certain separation properties with respect to the points of $S$, while cells of type-1 generally do not. Letting $u$ denote the cell under consideration, $u$ has the following properties depending on its type.

**Type-1:** $u$ enjoys no separation property in general.

**Type-2:** There exists a ball $b'_u$ such that $|S \cap (\gamma b_u \setminus b'_u)| = O(1/f)$, and the ball $\gamma b'_u$ does not overlap $u$.

**Type-3:** $u$ is a quadtree box. There is an associated quadtree box $v$ such that $u \subseteq v$ and $|S \cap (\gamma b_v \setminus 8b_u)| = O(1/f)$.

Our query processing will be presented with a pair $(q, R)$ where $q$ is the center and $R$ is the radius of the query ball $B$.

**Lemma 3** *Let $S$ be a set of $n$ points in $\mathbb{R}^d$. Let $\gamma \geq 16$ and $0 < f \leq 1$ be two real parameters. In $O(n\gamma^d \log(n\gamma) \log \gamma)$ time, it is possible to construct a data structure with $O(nf\gamma^d)$ cells of type-1, type-2, and type-3, respectively, such that the following holds. For any query $(q, R)$, where $q$ is a point in $\mathbb{U}^d$ and $0 \leq R \leq \sqrt{d}$ is a real number, in $O(\log(n\gamma))$ time, we can find a cell $u$ such that $q \in u$ and $u$ satisfies one of the following properties.*

(*i*) *$u$ is of type-1 and $\gamma r_u/4 < R \leq \gamma r_u/2$.*

(*ii*) *$u$ is of type-2 and $R \leq \gamma r_u/4$.*

(*iii*) *$u$ is of type-3 and $\gamma r_u/4 < R \leq \gamma r_v/4$, where $v$ denotes the quadtree box associated with $u$. (see properties of type-3 cells given above.)*

**Proof**: Let $T$ be the BD tree described in Lemma 1. Let $x$ be the leaf cell that contains $q$. If $R \leq \gamma r_x/4$, then we set $u$ to $x$. Observe that property (ii) holds in this case. Otherwise, let $v$ be the first node on the path from leaf $x$ to the root, such that $R \leq \gamma r_v/4$, and let $w$ be $v$'s child on this path. Note that both $v$ and $w$ are box cells, and $\gamma r_w/4 < R \leq \gamma r_v/4$. There are two cases to consider. If $w$ is obtained from $v$ by splitting, then $r_v = 2r_w$ and so we have $\gamma r_w/4 < R \leq \gamma r_w/2$. In this case, property (i) holds with $u$ set to $w$. Otherwise, $w$ is obtained from $v$ by shrinking. In this case, if property (ii.a) of Lemma 1 holds, then clearly property (ii) holds with $u$ set to $v$. Finally, if property (ii.b) of Lemma 1 holds, then property (iii) holds with $u$ set to $w$.

To complete the proof, we show that it possible to determine the cell $u$ efficiently. We could do this by transforming the (unbalanced) BD tree into a (balanced) BBD tree and applying an appropriate search procedure there. Justifying this, however, this would involve a deeper understanding of the internal structure of the BBD tree [7] than we care to get into at this point. Instead, we will show how this can be done using an alternate balanced tree structure.

Let us focus on the fundamental operations that are required. Given the BD tree and a query $(q, R)$, there is a unique path from the root of the tree to the leaf whose cell contains $q$. The sizes of the cells associated with the nodes of the tree decrease monotonically as we descend the tree. Our objective is to compute the first node $v$ along this path such that $R \leq \gamma r_v/4$, or failing this, to return the last node of the path. To do this, we convert the BD tree $T$ into a balanced tree structure, such as a link-cut tree [20] or topology tree [12]. Either data structure can be modified to support such searches in $O(\log m)$ time, where $m$ is the number of nodes in the tree. The resulting tree structure has the same space requirements as the BD tree and the resulting search time will be $O(\log(n\gamma))$, as desired. □

Henceforth, we refer to this data structure described in the preceding proof as the *augmented BD tree*. It will be convenient to rephrase properties (i) to (iii) slightly in terms of the query ball itself.

**Lemma 4** *Assume the same conditions as in Lemma 3. For any query ball $B = b(q, R) \subseteq \mathbb{U}^d$, in $O(\log(n\gamma))$ time, we can find a cell $u$ such that $q \in u$ and $u$ satisfies one of the following properties.*

(*i*) *$u$ is of type-1 and $(\gamma/8)b_u \subseteq B \subseteq \gamma b_u$.*

(*ii*) *$u$ is of type-2 and $B \subseteq \gamma b_u$.*

(*iii*) *$u$ is of type-3 and $(\gamma/8)b_u \subseteq B \subseteq \gamma b_v$, where $v$ denotes the quadtree box associated with $u$. (see properties of type-3 cells given above.)*

**Proof**: Since $b(q, R) \subseteq \mathbb{U}^d$, it follows that $q \in \mathbb{U}^d$ and $0 \leq R \leq \sqrt{d}$. By Lemma 3, for the query $(q, R)$, in $O(\log(n\gamma))$ time, we can find a cell $u$ such that $q \in u$ and $u$ satisfies one of the three properties listed therein.

Suppose that property (i) of Lemma 3 holds. That is, $u$ is of type 1 and $\gamma r_u/4 < R \leq \gamma r_u/2$. By the triangle inequality, the distance of any point in the range $B$ from the center of $u$ is at most $R + r_u \leq (\gamma r_u/2) + r_u$. Since $\gamma \geq 16$, this is at most $\gamma r_u$. Thus $B \subseteq \gamma b_u$. Again, by the

triangle inequality, the distance of any point on the boundary of $B$ from the center of $u$ is at least $R - r_u \geq \gamma r_u/4 - r_u$. Since $\gamma \geq 16$, this is at least $\gamma r_u/8$. Thus $(\gamma/8)b_u \subseteq B$. Thus, property (i) of Lemma 3 implies that $(\gamma/8)b_u \subseteq B \subseteq \gamma b_u$. Similarly, we can establish properties (ii) and (iii) as well. □

# 4 Approximate Range Counting

In this section we describe how to apply the results of the previous section to construct a data structure for answering approximate range counting queries, and how queries are answered. First, we set $f = (\varepsilon\gamma)^{d-1}$ and construct the augmented BD tree $T$ described in Lemma 3. (If $\gamma < 64$, we set $\gamma$ to 64 before using the lemma.) This takes time $O(n\gamma^d \log(n\gamma) \log \gamma)$. Let $b(q, R) \subseteq \mathbb{U}^d$ denote the query ball. Applying Lemma 4, it follows that, in $O(\log(n\gamma))$ time, we can find a cell $u$ such that $q \in u$ and $u$ satisfies one of the three properties listed therein.

In the next three subsections, we consider each of the three types of cells in turn (first type-2, then type-3, and finally type-1). In each case we describe the information stored with each type of cell and explain how this information helps to answer queries efficiently. To speed up the preprocessing, we assume that we have also constructed the BBD tree $T_b$ for the set $S$ of points.

## 4.1 Type-2 cells

By property (ii) of Lemma 4, each type-2 cell is responsible for handling queries centered in $u$ that lie entirely within its $\gamma$-expansion. We handle the points in the cluster ball $b'_u$, by subdividing them into a grid of small disjoint generator subsets, and we handle the relatively small number of pollutants by brute force. During the preprocessing phase, for each type-2 cell $u$, we compute a set $\mathcal{Q}(u)$ of weighted quadtree boxes as follows. Let $b'_u$ be the ball described in Lemma 1(i). If $b'_u$ is the empty ball or does not overlap $\gamma b_u$, then we set $\mathcal{Q}(u) = \emptyset$. Otherwise, we expand the ball $b'_u$ such that $\gamma b'_u$ just touches $u$. Henceforth, we will use $b'_u$ to refer to this expanded ball and $r'_u$ to denote its radius. (See Fig. 2.) We then find the set $\mathcal{Q}(u)$ of nonempty quadtree boxes of diameter $2\varepsilon(\gamma - 1)r'_u/3$ that overlap $b'_u$. By a straightforward packing argument, $|\mathcal{Q}(u)| = O(1/(\varepsilon\gamma)^d)$. For each box $z \in \mathcal{Q}(u)$, we assign it a *weight* equal to $|S \cap z|$. By BBD property (ii), we can compute $\mathcal{Q}(u)$ and assign weights to the boxes in it in time $O(\log n + t_u)$, where $t_u$ is the number of nonempty quadtree boxes of diameter $2\varepsilon(\gamma - 1)r'_u/3$ that overlap the ball $2b'_u$.

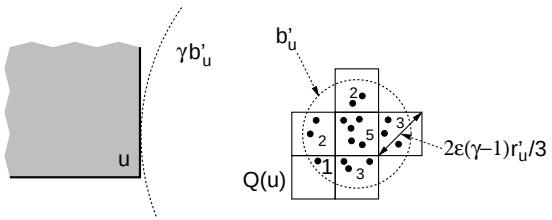


Fig. 2: Processing of type-2 cells.

Next, we scan the list of $O(1/f) = O(1/(\varepsilon\gamma)^{d-1})$ pollutants in $S \cap (\gamma b_u \setminus b'_u)$ and eliminate those points that are contained in a box in $\mathcal{Q}(u)$; let $\mathcal{P}(u)$ denote the set of points that remain. Assuming the floor function, $\mathcal{P}(u)$ can be easily computed in time $O(1/(\varepsilon\gamma)^{d-1})$.

By storing $\mathcal{P}(u)$ and $\mathcal{Q}(u)$ with each type-2 cell $u$, we can answer queries as follows. Recall that $u$ is responsible for answering the query for a ball $B$ centered in $u$ such that $B \subseteq \gamma b_u$. To answer this query, we do a linear scan of the boxes in $\mathcal{Q}(u)$ and compute the total weight of the boxes that overlap $B$. To this we add the number of points of $\mathcal{P}(u)$ that lie inside $B$ and return it as the answer. The correctness of this approach is obvious from Lemma 1(i) and the fact that if $B$ overlaps a box $z$ in $\mathcal{Q}(u)$ then the diameter of $z$ is at most $\varepsilon$ times the radius of $B$. The query time is $O(|\mathcal{Q}(u)| + |\mathcal{P}(u)|) = O(1/(\varepsilon\gamma)^d)$.

By using a slightly more sophisticated method, we can reduce the query time by a factor of $O(1/(\varepsilon\gamma))$. The idea is to organize the quadtree boxes in $\mathcal{Q}(u)$ into a BD tree during the preprocessing phase. By BD property (ii), this can be done in time $O(|\mathcal{Q}(u)| \log |\mathcal{Q}(u)|)$. With the help of this BD tree, using standard techniques [5], we can find the total weight of the boxes $z \in \mathcal{Q}(u)$ that overlap the query range $B$ in time proportional to the number of boxes of $\mathcal{Q}(u)$ that overlap $\partial B$. A straightforward packing argument [5] shows that this quantity is $O(1/(\varepsilon\gamma)^{d-1})$. Since the time to scan $\mathcal{P}(u)$ is also $O(1/(\varepsilon\gamma)^{d-1})$, this quantity bounds the overall query time.

We now estimate the space requirements for all the leaves. By the bounds on $|\mathcal{P}(u)|$ and $|\mathcal{Q}(u)|$ given above, the space used for a leaf $u$ is $O(1/(\varepsilon\gamma)^d)$. Recall that the number of leaves is $O(nf\gamma^d \log\gamma)$, where $f = (\varepsilon\gamma)^{d-1}$. Thus, the total space used by all the leaves together is $O((n\gamma^{d-1} \log\gamma)/\varepsilon)$. However, this simple bound is based on the assumption that, for every type-2 cell, $|\mathcal{Q}(u)|$ achieves its worst case space bound. Lemma 5 shows that this cannot happen and improves this bound by a factor of nearly $O(1/(\varepsilon\gamma))$ using a charging argument (similar to that used in [2]). Let $\mathcal{B}_T$ denote the set of type-2 cells of $T$.

**Lemma 5** *Let $T$ be the BD tree described in Lemma 4 for any value of $f$, $0 < f \leq 1$. For any cell $u \in \mathcal{B}_T$, let $\mathcal{Q}(u)$ be as defined above. Then $\sum_{u\in\mathcal{B}_T} |\mathcal{Q}(u)| = O(n\gamma^d \log(1/\varepsilon))$.*

**Proof**: Recall that $\mathcal{B}_T$ includes all the leaf cells of $T$. In addition, if $u$ is a box cell obtained by a shrink operation that satisfies property (ii.b) of Lemma 1, then its parent cell $v$ belongs to $\mathcal{B}_T$. We will establish this lemma only for the set of leaf cells; the proof for the other case is similar.

Let $\mathcal{D}$ be the set of dumbbells corresponding to the well-separated pair decomposition of $S$, using separation factor 8. Each dumbbell $P \in \mathcal{D}$ allocates a unit *charge* to the leaf cells in $T$ satisfying certain conditions. Let $a$ and $b$ denote the centers of two heads of $P$, $\ell = \|ab\|$, and $o$ denote the center of the segment $\overline{ab}$. Let $\mathcal{B}_P$ denote the set of balls, centered at $o$, having radius $2^i\ell$, where $\log(c_1\gamma) \leq i \leq \log(c_2/\varepsilon)$ for suitable constants $c_1$ and $c_2$. For a ball $b \in \mathcal{B}_P$, let $\mathcal{C}_b$ be the set of leaf cells in $T$ overlapping $b$ that have size at least $c_3 r_b/\gamma$, where $r_b$ denotes the radius of $b$ and $c_3$ is a suitable constant. The dumbbell $P$ allocates a unit charge to each leaf cell in $\mathcal{C}_P = \cup_{b\in\mathcal{B}_P}\mathcal{C}_b$. By BD property (iii), $|\mathcal{C}_b| = O(\gamma^d)$. Since the number of balls in $\mathcal{B}_P$ is $O(\log(1/(\varepsilon\gamma)))$, it follows that $P$ allocates a unit charge to $O(\gamma^d \log(1/(\varepsilon\gamma)))$ leaf cells. Thus, the total charge allocated by all the dumbbells together is $O(n\gamma^d \log(1/(\varepsilon\gamma)))$.

Next we show that each leaf cell $u$ receives a charge from at least $\Omega(|\mathcal{Q}(u)| - 1)$ dumbbells. Recall that $r'_u$ is the radius of the ball $b'_u$ such that the ball $\gamma b'_u$ just touches $u$. We now show that there exists a subset $\mathcal{Q}'(u) \subseteq \mathcal{Q}(u)$ such that $|\mathcal{Q}'(u)| = \Omega(|\mathcal{Q}(u)|)$ and the distance between any pair of boxes in $\mathcal{Q}'(u)$ is at least $\Omega(\varepsilon\gamma r'_u)$. We can find $\mathcal{Q}'(u)$ as follows. Initially, we set $\mathcal{Q}'(u) = \emptyset$. We then consider the boxes in $\mathcal{Q}(u)$ one by one. Two boxes in $\mathcal{Q}(u)$ are said to be *neighbors* if they share a $(d-1)$-facet. At each step, we add a box in $\mathcal{Q}(u)$ to $\mathcal{Q}'(u)$ and then eliminate it and all its neighbors in $\mathcal{Q}(u)$ from further consideration. We continue in this manner until all the boxes in $\mathcal{Q}(u)$ have been pruned. Clearly this process finds a set $\mathcal{Q}'(u)$ with the desired properties.

Let $\mathcal{X}(u)$ be the set of points obtained by picking one point of $S$ from each box in $\mathcal{Q}'(u)$. The distance between any two points in $\mathcal{X}(u)$ is $\Omega(\varepsilon\gamma r'_u)$ and at most $O(r'_u)$. It is easy to show that there exist $|\mathcal{X}(u)| - 1$ distinct dumbbells in $\mathcal{D}$ that separate pairs of points in $\mathcal{X}(u)$ (we omit the straightforward details). Noting that $s_u = \Omega(r'_u)$, one can now verify that each of these dumbbells allocates a charge to cell $u$, for a suitable choice of the constants $c_1, c_2$ and $c_3$. Thus we have shown that each leaf cell $u$ receives a charge of $\Omega(|\mathcal{Q}(u)| - 1)$.

Let $\mathcal{L}_T$ denote the set of leaf cells of $T$. Since the total charge allocated by all the dumbbells together is $O(n\gamma^d \log(1/(\varepsilon\gamma)))$ and the number of leaf cells is $O(nf\gamma^d \log\gamma)$, it follows that $\sum_{u\in\mathcal{L}_T} |\mathcal{Q}(u)| = O(n\gamma^d \log(1/(\varepsilon\gamma)) + nf\gamma^d \log\gamma) = O(n\gamma^d \log(1/\varepsilon)$. This completes the proof. □

Using this lemma and the bound on $|\mathcal{P}(u)|$ given above, it follows that the space used by all the type-2 cells is $O(n\gamma^d \log(1/\varepsilon))$.

We now estimate the preprocessing time for all the type-2 cells. Recall that for each type-2 cell $u$, $\mathcal{P}(u)$ can be computed in time $O(1/(\varepsilon\gamma)^{d-1})$ and $\mathcal{Q}(u)$ can be computed in time $O(t_u + \log n)$. Here $t_u$ is the number of nonempty quadtree boxes of diameter $2\varepsilon(\gamma - 1)r'_u/3$ that overlap the ball $2b'_u$. Applying a similar charging argument as used above to bound $\sum_{u\in\mathcal{B}_T} |\mathcal{Q}(u)|$, it follows that $\sum_{u\in\mathcal{B}_T} t_u = O(n\gamma^d \log(1/\varepsilon))$. Thus, in time $O((f\gamma^d \log\gamma)n \log n + (\gamma^d \log(1/\varepsilon))n)$, we can compute $\mathcal{P}(u)$ and $\mathcal{Q}(u)$ for all type-2 cells $u$. Also, recall that it takes $O(|\mathcal{Q}(u)| \log |\mathcal{Q}(u)|)$ time to organize the boxes in $\mathcal{Q}(u)$ into a BD tree. Since $|\mathcal{Q}(u)| = O((1/\varepsilon\gamma)^d)$ and $\sum_{u\in\mathcal{B}_T} |\mathcal{Q}(u)| = O(n\gamma^d \log(1/\varepsilon))$, it follows that

$$\sum_{u\in\mathcal{B}_T} |\mathcal{Q}(u)| \log |\mathcal{Q}(u)| \;=\; O(n\gamma^d \log^2(1/\varepsilon)).$$

Thus, the total preprocessing time for all the type-2 cells is $O((f\gamma^d \log\gamma)n \log n + (\gamma^d \log^2(1/\varepsilon))n)$ $= O(n\gamma^d \log(n/\varepsilon) \log(1/\varepsilon))$.

### 4.2 Type-3 cells

By property (iii) of Lemma 4, each type-3 cell $u$ is responsible for handling query ranges $B$ that are centered in $u$ and satisfy $(\gamma/8)b_u \subseteq B \subseteq \gamma b_v$ where $v$ is $u$'s parent cell. Since $\gamma \geq 64$, it follows that $B \supseteq 8b_u$. Thus the exact answer to such a query can be computed as the sum of $|S \cap 8b_u|$ and the number of points of $S \cap (\gamma b_v \setminus 8b_u)$ that lie within $B$. Both $|S \cap 8b_u|$ and $S \cap (\gamma b_v \setminus 8b_u)$ are precomputed and stored with $u$, so the query can be answered in time $O(1/(\varepsilon\gamma)^{d-1})$. The total space and preprocessing time for all the type-3 cells is the same as for all the type-2 cells.

### 4.3 Type-1 cells

By property (i) of Lemma 4, each type-1 cell $u$ is responsible for handling query ranges $B$ that are centered in $u$ such that $\gamma' b_u \subseteq B \subseteq \gamma'' b_u$, where $\gamma' = \gamma/8$ and $\gamma'' = \gamma$. Such a cell does not generally enjoy any separation properties with respect to the point set $S$. Our approach is to compute a cover of the annulus $\gamma'' b_u \setminus \gamma' b_u$ by a set of disjoint regions such that the subset of points lying within these regions can be used as generators for the query. We describe two approaches for constructing these regions. The first is a simple method based on a grid decomposition of the annulus. The second achieves better query performance and can be thought of as a radial version of a quadtree decomposition taking place in the space of polar coordinates. We provide both because the space analysis of the second method depends on the space analysis of the first.

During the preprocessing phase, we compute an annulus cover $\mathcal{Q}(u)$ by boxes of size $\varepsilon\gamma' s_u/4$ for the annulus $\gamma'' b_u \setminus \gamma' b_u$. For each box $z \in \mathcal{Q}(u)$, we assign it a *weight* equal to $|S \cap z|$. (See Fig. 3(a).) By BBD property (ii) this can be done in time $O(\log n + t_u)$, where $t_u$ is the number of nonempty quadtree boxes of size $\varepsilon\gamma' s_u/4$ overlapping the larger annulus $2\gamma'' b_u \setminus (\gamma'/2) b_u$. In the same time, using standard techniques [5], we compute the number of points in $S \cap \gamma' b_u$ that are not contained in any box of $\mathcal{Q}(u)$, and store this information with $u$.

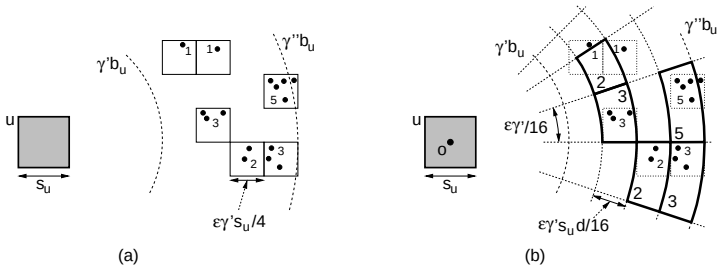


Fig. 3: Processing of type-1 cells: (a) the annulus cover $\mathcal{Q}(u)$ and (b) the nonempty radial fragments $\mathcal{F}'(u)$.

By a straightforward packing argument, $|\mathcal{Q}(u)| = O(1/\varepsilon^d)$. Let $\mathcal{A}_T$ denote the set of type-1 cells. Since $|\mathcal{A}_T| = O(nf\gamma^d \log \gamma)$, where $f = (\varepsilon\gamma)^{d-1}$, it follows that $\sum_{u \in \mathcal{A}_T} |\mathcal{Q}(u)| = O((n\gamma^{2d-1} \log \gamma)/\varepsilon)$. This bound can be significantly improved by using a charging argument similar to that used earlier for type-2 cells. Applying this argument yields $\sum_{u \in \mathcal{A}_T} |\mathcal{Q}(u)| = O(n\gamma^d \log(1/\varepsilon))$. (We omit the details.) Similarly, we obtain $\sum_{u \in \mathcal{A}_T} t_u = O(n\gamma^d \log(1/\varepsilon))$. Thus, the time to compute $\mathcal{Q}(u)$ for all the cells $u \in \mathcal{A}_T$ is $O((f\gamma^d \log \gamma) n \log n + (\gamma^d \log(1/\varepsilon))n)$.

It is possible to answer a query by computing the total weight of the boxes in $\mathcal{Q}(u)$ that overlap the range $B$. However, since $|\mathcal{Q}(u)|$ can be as large as $\Omega(1/\varepsilon^d)$, this would lead to a high query time. We now discuss how this can be reduced to $O(1/(\varepsilon\gamma)^{d-1})$ by using a more efficient decomposition of the annulus based on a radial decomposition of space.

First we need to introduce some notation. Let $o$ denote the center of the quadtree box $u$. Let $H$ denote the axis-parallel hypercube of size $\gamma' s_u d$ centered at $o$. Consider a regular grid of side length $\varepsilon(\gamma')^2 s_u/(16\pi)$ on each of the $2d$ faces of dimension $(d-1)$ of $\partial H$. Let $\mathcal{C}$ be the set of cones that have their apex at $o$ and whose base is a cell in the grid placed on the faces of $\partial H$. Extend each cone through this base to infinity. It is easy to see that the angular diameter of each cone in $\mathcal{C}$ is at most $\varepsilon\gamma'/16$ and $|\mathcal{C}| = O(1/(\varepsilon\gamma)^{d-1})$. Let $\mathcal{B}$ be a set of $O(1/\varepsilon)$ spheres between $\gamma' b_u$ and $\gamma'' b_u$, such that the radii of any two successive spheres differs by at most $\varepsilon\gamma' s_u d/16$. The boundaries of both $\gamma' b_u$ and $\gamma'' b_u$ are included in $\mathcal{B}$. (See Fig. 3(b).) We refer to a cell in the arrangement of $\mathcal{C} \cup \mathcal{B}$ as a *fragment*. Note that $\gamma'' b_u \setminus \gamma' b_u$ is partitioned into $O(1/(\varepsilon\gamma)^{d-1}(1/\varepsilon))$ fragments; let $\mathcal{F}(u)$ denote this set of fragments.

During the preprocessing phase, for each quadtree box $z \in \mathcal{Q}(u)$, we determine a fragment in $\mathcal{F}(u)$ that it overlaps (assuming the floor function, this takes constant time), and add the weight of $z$ to the weight of the fragment. Initially we assume that the weight of all the fragments is zero. (The reason for assigning weights from quadtree boxes of the annulus cover, rather than from the point set $S$ directly, is that quadtree boxes share a common coordinate system, which allows us to construct and analyze them globally for all nodes. In contrast, the radial fragments are based on coordinate systems that are local to each node.) At the end of this process, the weight of a fragment

is the sum of the weights of all the quadtree boxes in $\mathcal{Q}(u)$ that transfer their weight to it. Let $\mathcal{F}'(u)$ denote the set of fragments that finally have a non-zero weight. (See Fig. 3(b).) Since $\mathcal{F}'(u)$ typically has fewer fragments than $\mathcal{F}(u)$, we use hashing [13, 19] to compute only the fragments of $\mathcal{F}'(u)$ along with their associated weights. This can be done in time $O(|\mathcal{Q}(u)|)$. The total space used to store $\mathcal{F}'(u)$ over all type-1 cells is $\sum_{u\in\mathcal{A}_T}|\mathcal{F}'(u)| \le \sum_{u\in\mathcal{A}_T}|\mathcal{Q}(u)| = O(n\gamma^d \log(1/\varepsilon))$.

We answer a query by scanning the fragments of $\mathcal{F}'(u)$ and determining the total weight of the fragments that overlap the range $B$. To this we add the precomputed number of the points of $S \cap \gamma' b_u$ that are not included in any box of $\mathcal{Q}(u)$. The correctness of this method follows from Lemma 7. Intuitively, the lemma shows that the algorithm makes errors only with respect to points of $S$ that lie close to the boundary of $B$. An important ingredient in its proof is Lemma 6, which implies that all points in a fragment are at nearly the same distance from any point in cell $u$. Define the *absolute ratio* between two positive numbers $x$ and $y$ to be the maximum of $x/y$ and $y/x$.

**Lemma 6** *Let $0 < \varepsilon \le 1/2$ and $\hat{\gamma} \ge 2$. Let $u$ be a hypercube of side length $s$ centered at $o$. Let $x_1$ and $x_2$ be two points such that $\|ox_1\| \ge \|ox_2\| \ge \hat{\gamma}sd$, $\|ox_1\| - \|ox_2\| \le \varepsilon\hat{\gamma}sd/8$, and $\angle x_1ox_2 \le \varepsilon\hat{\gamma}/8$. Then for any point $o'$ in $u$, the absolute ratio of $\|o'x_1\|$ and $\|o'x_2\|$ is at most $1 + \varepsilon/3$.*

**Proof**: Let $x_1'$ be the point on the ray $ox_1$ such that $\|ox_1'\| = \|ox_2\|$. Let $\theta = \angle o'ox_1'$ and $\phi = \angle o'ox_2$. Applying the law of cosines to $\triangle o'ox_1'$ and $\triangle o'ox_2$, we obtain

$$\|o'x_1'\|^2 = \|o'o\|^2 + \|ox_1'\|^2 - 2\|o'o\| \cdot \|ox_1'\| \cdot \cos\theta, \tag{1}$$

and

$$\|o'x_2\|^2 = \|o'o\|^2 + \|ox_2\|^2 - 2\|o'o\| \cdot \|ox_2\| \cdot \cos\phi. \tag{2}$$

Subtracting Eq. (2) from Eq. (1), we get

$$\begin{aligned}\|o'x_1'\|^2 - \|o'x_2\|^2 &= 2\|o'o\| \cdot \|ox_2\| \cdot (\cos\phi - \cos\theta)\\ &= 2\|o'o\| \cdot \|ox_2\| \cdot 2\sin((\theta+\phi)/2)\sin((\theta-\phi)/2).\end{aligned}$$

Note that $\sin((\theta+\phi)/2) \le 1$, $\sin((\theta-\phi)/2) \le |\theta-\phi|/2$ and, by the triangle inequality on angles, $|\theta - \phi| \le \angle x_1ox_2 \le \varepsilon\hat{\gamma}/8$. Thus

$$\|o'x_1'\|^2 - \|o'x_2\|^2 \le \|o'o\| \cdot \|ox_2\| \cdot \varepsilon\hat{\gamma}/4.$$

We can rewrite this as

$$\begin{aligned}\frac{\|o'x_1'\|^2}{\|o'x_2\|^2} - 1 &\le \frac{\|o'o\|}{\|o'x_2\|} \cdot \frac{\|ox_2\|}{\|o'x_2\|} \cdot \frac{\varepsilon\hat{\gamma}}{4}\\ &\le \frac{\|o'o\|}{\|ox_2\| - \|o'o\|} \cdot \frac{1}{1 - \|o'o\|/\|ox_2\|} \cdot \frac{\varepsilon\hat{\gamma}}{4},\end{aligned}$$

where we have used the triangle inequality $\|o'x_2\| \ge \|ox_2\| - \|o'o\|$. Since $\|o'o\| \le sd/2$ and $\|ox_2\| \ge \hat{\gamma}sd \ge 2sd$, we get $\|o'o\|/(\|ox_2\| - \|o'o\|) \le 1/(2\hat{\gamma} - 1)$ and $1/(1 - \|o'o\|/\|ox_2\|) \le 4/3$. Thus

$$\frac{\|o'x_1'\|^2}{\|o'x_2\|^2} - 1 \le \frac{4}{3} \cdot \frac{1}{2\hat{\gamma} - 1} \cdot \frac{\varepsilon\hat{\gamma}}{4} \le \frac{2\varepsilon}{9},$$

which simplifies to

$$\|o'x_1'\|/\|o'x_2\| \le 1 + \varepsilon/9. \tag{3}$$

By a symmetric argument, it follows that

$$\|o'x_2\|/\|o'x_1'\| \leq 1 + \varepsilon/9. \tag{4}$$

Also, by the triangle inequality,

$$\frac{\|o'x_1\|}{\|o'x_1'\|} \;\leq\; \frac{\|o'x_1'\| + \|x_1x_1'\|}{\|o'x_1'\|} \;\leq\; 1 + \frac{\|x_1x_1'\|}{\|ox_1'\| - \|o'o\|}.$$

Since $\|ox_1'\| \geq \hat{\gamma}sd$, $\|o'o\| \leq sd/2$ and, by the statement of the lemma, $\|x_1x_1'\| \leq \varepsilon\hat{\gamma}sd/8$, we get

$$\frac{\|o'x_1\|}{\|o'x_1'\|} \;\leq\; 1 + \frac{\varepsilon\hat{\gamma}/8}{\hat{\gamma} - 1/2} \;\leq\; 1 + \frac{\varepsilon}{6}. \tag{5}$$

Similarly, we can show that

$$\|o'x_1'\|/\|o'x_1\| \;\leq\; 1 + \varepsilon/6. \tag{6}$$

Combining Eqs. (3) and (5), we get

$$\|o'x_1\|/\|o'x_2\| \;\leq\; (1 + \varepsilon/9)(1 + \varepsilon/6) \;\leq\; 1 + \varepsilon/3,$$

since $\varepsilon \leq 1/2$. Similarly, combining Eqs. (4) and 6, we get $\|o'x_2\|/\|o'x_1\| \leq 1 + \varepsilon/3$. This completes the proof. □

**Lemma 7** *Let $\gamma \geq 64$. Let $u$ be a type-1 cell. Let $B$ be a query ball centered at a point in $u$ such that $\gamma' b_u \subseteq B \subseteq \gamma'' b_u$, where $\gamma' = \gamma/8$ and $\gamma'' = \gamma$. Let $r$ denote the radius of $B$. Let $p$ be a point in $S$ whose distance from $\partial B$ is at least $r\varepsilon$, and let $z \in \mathcal{Q}(u)$ be the quadtree box that contains $p$.*

*(i) If $p \in B$, then any fragment in $\mathcal{F}'(u)$ that overlaps $z$ must also overlap $B$.*
*(ii) If $p \notin B$, then no fragment in $\mathcal{F}'(u)$ overlaps both $z$ and $B$.*

**Proof**: Let $o$ denote the center of cell $u$ and let $o' \in u$ denote the center of ball $B$. To prove (i), observe that the diameter of each box in $\mathcal{Q}(u)$ is at most $\varepsilon\gamma' s_u d/4 \leq r\varepsilon/2$, since $r \geq \gamma' s_u d/2$. By the triangle inequality, the distance of any point in $z$ from $o'$ is at most $r(1-\varepsilon) + r\varepsilon/2 < r$. It follows that $z \subseteq B$, which implies (i).

To prove (ii), let $t \in \mathcal{F}'(u)$ be any fragment that overlaps $B$. Let $x_2$ be any point in $t \cap B$ and $x_1$ be the point in $t$ that is farthest from $o$. The definition of fragments implies that $\angle x_1 o x_2 \leq \varepsilon\gamma'/16$, $\|ox_1\| \geq \|ox_2\| \geq \gamma' s_u d/2$ and $\|ox_1\| - \|ox_2\| \leq \varepsilon\gamma' s_u d/16$. Applying Lemma 6, it follows that $\|o'x_1\|/\|o'x_2\| \leq 1 + \varepsilon/3$. Since the diameter of box $z$ is at most $r\varepsilon/2$, it follows that the distance of any point in $z$ from $o'$ is at least $r(1+\varepsilon) - r\varepsilon/2 > r(1+\varepsilon/3)$. Thus $t$ cannot overlap $z$. This completes the proof. □

The query time of this method is $O((1/\varepsilon\gamma)^{d-1}(1/\varepsilon))$, since it is proportional to the number of fragments in $\mathcal{F}'(u)$. While this is already a significant improvement over the query time obtained by searching all the boxes in $\mathcal{Q}(u)$, we can speed it up still further through the use of BD trees. The idea is to organize the fragments of $\mathcal{F}'(u)$ into $2d$ BD trees, one for each of the faces of $H$. Since the fragments of $\mathcal{F}(u)$ are pieces of a $d$-dimensional annuli, they are certainly not quadtree boxes. Nonetheless, by applying an appropriate transformation to polar coordinates, it is possible to map

the fragments to a $d$-dimensional grid of hypercubes, from which we can then apply any standard algorithm for constructing a BD tree from a set of quadtree boxes [5]. To see how a point in this "fragment space" can be described as $d$ polar coordinates, the first $d-1$ polar coordinates arise by shooting a ray from $o$ through this point until it hits a $(d-1)$-dimensional face of the hypercube $H$, and then representing the resulting point using a local coordinate system for this face. Based on these coordinates alone, the fragments form a $(d-1)$-dimensional grid of hypercubes of side length $h = \varepsilon(\gamma')^2 s_u/(16\pi)$. The last coordinate of the polar representation arises from the distance of the point from the center of $H$. Along this axis, each fragment has a length of $r = \varepsilon\gamma' s_u d/16$. By scaling this last coordinate by the amount $h/r$, in polar space the fragments are now mapped to a grid of $d$-dimensional hypercubes, as desired.

Now, by applying standard techniques to the resulting BD tree [5], we can determine the total weight of the fragments of $\mathcal{F}'(u)$ that overlap $B$ in time proportional to the number of fragments that intersect $\partial B$ (transformed into polar space). Applying Lemma 6, it is not hard to show that $\partial B$ intersects $O(1)$ fragments in each cone of $\mathcal{C}$, which implies that the query time is $O(1/(\varepsilon\gamma)^{d-1})$.

It takes $O(|\mathcal{F}'(u)| \log |\mathcal{F}'(u)|)$ time to construct the BD trees for the fragments in $\mathcal{F}'(u)$. Noting that $\sum_{u\in\mathcal{A}_T} |\mathcal{F}'(u)| = O(n\gamma^d \log(1/\varepsilon))$ and $|\mathcal{F}'(u)| = O((1/\varepsilon\gamma)^{d-1}(1/\varepsilon))$, it follows that

$$\sum_{u\in\mathcal{A}_T} |\mathcal{F}'(u)| \log |\mathcal{F}'(u)| \;=\; O(n\gamma^d \log^2(1/\varepsilon)).$$

Thus, the preprocessing time for all the type-1 cells is $O((f\gamma^d \log\gamma)n \log n + (\gamma^d \log^2(1/\varepsilon))n) = O(n\gamma^d \log(n/\varepsilon)\log(1/\varepsilon))$.

Putting it all together, we have shown the following theorem.

**Theorem 1** *Let $S$ be a set of $n$ points in $\mathbb{R}^d$, and let $0 < \varepsilon \le 1/2$ and $2 \le \gamma \le 1/\varepsilon$ be two real parameters. Then we can construct a data structure of $O(n\gamma^d \log(1/\varepsilon))$ space that allows us to answer $\varepsilon$-approximate range queries in time $O(\log(n\gamma) + 1/(\varepsilon\gamma)^{d-1})$. The time to construct the data structure is $O(n\gamma^d \log(n/\varepsilon)\log(1/\varepsilon))$.*

## 5 Concluding Remarks

We have presented an algorithm and data structure that provides space-time tradeoffs for approximate spherical range counting queries in fixed dimensions. At one extreme (low space), our results yield a data structure of space $O(n\log(1/\varepsilon))$ that can answer approximate range queries in time $O(\log n + (1/\varepsilon)^{d-1})$ which, up to a factor of $O(\log 1/\varepsilon)$ in space, matches the best known result [5]. At the other extreme (high space), it yields a data structure of space $O((n/\varepsilon^d)\log(1/\varepsilon))$ that can answer queries in time $O(\log n + \log 1/\varepsilon)$. This is the fastest known query time for this problem.

Although we have not shown it here, our results can be adapted to answering approximate $k$th nearest neighbor queries, where $1 \le k \le n$. In range searching the query ball's radius is given. In the $k$th nearest neighbor problem, the problem in essence is to determine the range radius of the ball, centered at the query point, that contains $k$ points. It is possible to generalize the search algorithm used in range searching to determine this radius. This additional generality results in an increase by a factor of $O(1/(\varepsilon\gamma))$ in the query time. The space bounds are the same.

The results of this paper apply to the general case of range queries in which the points have been assigned weights that have been drawn from any faithful semigroup. (The case of range counting is the simplest variant of this in which the semigroup is the natural numbers under addition.) A

natural question is whether the computational complexity of approximate range searching is lower for semigroups and range shapes that satisfy certain additional properties. A semigroup $(S, +)$ is *idempotent* if $x+x = x$ for all $x \in S$, and it is *integral* if for all $k \geq 2$, the $k$-fold sum $x+\cdots+x$ is not equal to $x$. In [4] we show that the complexity of approximate range searching for Euclidean balls is significantly lower for idempotent semigroups than for integral semigroups. In [3] we showed that the benefits offered by idempotence do not apply if the range shape is allowed to have sharp corners. We also show there that if ranges are not Euclidean balls, but are sufficiently smooth convex shapes, then the assumption that the semigroup is idempotent can offer significant improvements in query time.

## References


[1] S. Arya and T. Malamatos. Linear-size approximate Voronoi diagrams. In *Proc. 13th Annu. 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, T. Malamatos, and D. M. Mount. The effect of corners on the complexity of approximate range searching. In *Proc. 22nd Annu. ACM Sympos. Comput. Geom.*, pages 11–20, 2006.

[4] S. Arya, T. Malamatos, and D. M. Mount. On the importance of idempotence. In *Proc. 38th Annu. ACM Sympos. Theory Comput.*, pages 564–573, 2006.

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

[6] S. Arya, D. M. Mount, N. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. In *Proc. 5th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 573–582, 1994.

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

[8] H. Brönnimann, B. Chazelle, and J. Pach. How hard is halfspace range searching. *Discrete Comput. Geom.*, 10:143–155, 1993.

[9] 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.

[10] B. Chazelle. Lower bounds on the complexity of polytope range searching. *J. Amer. Math. Soc.*, 2:637–666, 1989.

[11] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, Germany, 2nd edition, 2000.

[12] G. N. Frederickson. A data structure for dynamically maintaining rooted trees. In *Proc. 4th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 175–194, 1993.

[13] M. L. Fredman, J. Komlos, and E. Szemeredi. Storing a sparse table with $O(1)$ worst case access time. *J. ACM*, 31(3):538–544, 1984.

[14] S. Funke and E. A. Ramos. Smooth-surface reconstruction in near-linear time. In *Proc. 13th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 781–790, 2002.

[15] 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.

[16] J. Majhi, R. Janardan, M. Smid, and P. Gupta. On some geometric optimization problems in layered manufacturing. *Comput. Geom. Theory Appl.*, 12:219–239, 1999.

[17] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete Comput. Geom.*, 10(2):157–182, 1993.

[18] J. Matoušek. On approximate geometric $k$-clustering. *Discrete and Comput. Geometry*, 24:61–84, 2000.

[19] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, 1995.

[20] D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. *J. Comput. Sys. Sci.*, 26:362–391, 1983.