A Unified Approach to Proximity Search Through Delone Sets

Ahmed Abdelkader

Department of Computer Science
University of Maryland, College Park

(Joint work with David Mount)

Presented at the Capital Area Theory Seminar 10/06/2017
Proximity Searching: Applications

Proximity searching:
A set of related geometric retrieval problems that involve finding the objects close to a given query object.

- Pattern recognition and classification
- Object recognition in images
- Content-based retrieval:
  - Shape matching
  - Image/Document retrieval
  - Biometric identification (face/fingerprint/voice recognition)
- Clustering and phylogeny
- Data compression (vector quantization)
- Physical simulation (collision detection and response)
- Computer graphics: photon mapping and point-based modeling

...and many more
Nearest Neighbor Searching

Preprocess a point set $P \subset \mathbb{R}^d$, so that given any query point $q \in \mathbb{R}^d$, can efficiently find its closest point in $P$.

Assumptions:

- Real $d$-dimensional space
- Assume Euclidean distance
- Dimension is a constant (e.g., $d \leq 20$)
Nearest Neighbor Searching - Exact?

Ideal: $O(n)$ space and $O(\log n)$ query time

Voronoi Diagrams
- Subdivide space into regions according to which point is closest
- Apply point location to answer queries

- In $\mathbb{R}^2$: $O(n)$ space and $O(\log n)$ time
- No good solutions higher dimensions
- Curse of dimensionality
Nearest Neighbor Searching - Exact?

Ideal: $O(n)$ space and $O(\log n)$ query time

Voronoi Diagrams

- Subdivide space into regions according to which point is closest
- Apply point location to answer queries

- In $\mathbb{R}^2$: $O(n)$ space and $O(\log n)$ time
- No good solutions higher dimensions
- Curse of dimensionality
Nearest Neighbor Searching - Exact?

Ideal: $O(n)$ space and $O(\log n)$ query time

Voronoi Diagrams
- Subdivide space into regions according to which point is closest
- Apply point location to answer queries

- In $\mathbb{R}^2$: $O(n)$ space and $O(\log n)$ time
- No good solutions higher dimensions
- Curse of dimensionality
Nearest Neighbor Searching - Exact?

Ideal: $O(n)$ space and $O(\log n)$ query time

Voronoi Diagrams
- Subdivide space into regions according to which point is closest
- Apply point location to answer queries

- In $\mathbb{R}^2$: $O(n)$ space and $O(\log n)$ time
- No good solutions higher dimensions
- Curse of dimensionality
Approximate Nearest Neighbor (ANN)

Given a query point $q$, whose true nearest neighbor is $p^*$, return any point $p \in P$, such that

$$
\text{dist}(q, p) \leq (1 + \varepsilon) \cdot \text{dist}(q, p^*)
$$
Brief Survey

- Logarithmic query times, exponential dependencies on $d$
  - Trees (e.g., k-d trees, BBD, AVD)
  - Grids (e.g., bucketing, shifted/rotated, DVD)
  - Algebraic (Chebyshev polynomials)
- Sublinear query times, near-linear storage, polynomial dependencies on $d$
  - Locality-sensitive Hashing (LSH)
- And many more
  - Neighborhood graphs
  - Spectral methods (PCA)
  - Dynamic Continuous Indexing (DCI)
  - Offline (e.g., one-shot, batch queries)
  - Other metric spaces (e.g., doubling-dimension, Bregman distances)
  - Other variants: moving points, uncertainty, ...
ANN Search with kd-Trees

ANN Searching with kd-trees

- **Preprocessing:** $O(n \log n)$ time, $O(n)$ space
- **Query Processing:**
  - Locate the cell containing $q$
  - Establish initial search radius
  - Visit cells in increasing order of distance
  - Stop when: cell-dist > $NN$-dist/(1 + $\varepsilon$)

- **Query time:** $O(\log n + (1/\varepsilon)^d)$
- Works well in practice
ANN Searching with kd-trees

- **Preprocessing:** $O(n \log n)$ time, $O(n)$ space
- **Query Processing:**
  - Locate the cell containing $q$
  - Establish initial search radius
  - Visit cells in increasing order of distance
  - Stop when: $\text{cell-dist} > \text{NN-dist}/(1 + \varepsilon)$

- Query time: $O(\log n + (1/\varepsilon)^d)$
- Works well in practice
ANN Searching with kd-trees

- **Preprocessing:** \( O(n \log n) \) time, \( O(n) \) space
- **Query Processing:**
  - Locate the cell containing \( q \)
  - Establish initial search radius
  - Visit cells in increasing order of distance
  - Stop when: \( \text{cell-dist} > \text{NN-dist}/(1 + \varepsilon) \)

- **Query time:** \( O(\log n + (1/\varepsilon)^d) \)
- Works well in practice
### ANN Searching with kd-trees

- **Preprocessing:** $O(n \log n)$ time, $O(n)$ space
- **Query Processing:**
  - Locate the cell containing $q$
  - Establish initial search radius
  - Visit cells in increasing order of distance
  - Stop when: $\text{cell-dist} > \text{NN-dist}/(1 + \varepsilon)$

- **Query time:** $O(\log n + (1/\varepsilon)^d)$
- Works well in practice
ANN Searching with kd-trees

- **Preprocessing:** $O(n \log n)$ time, $O(n)$ space
- **Query Processing:**
  - Locate the cell containing $q$
  - Establish initial search radius
  - Visit cells in increasing order of distance
  - Stop when: $\text{cell-dist} > \text{NN-dist}/(1 + \varepsilon)$

- Query time: $O(\log n + (1/\varepsilon)^d)$
- Works well in practice
ANN Searching with kd-trees

- Preprocessing: $O(n \log n)$ time, $O(n)$ space
- **Query Processing:**
  - Locate the cell containing $q$
  - Establish initial search radius
  - Visit cells in increasing order of distance
  - Stop when: $\text{cell-dist} > \text{NN-dist}/(1 + \varepsilon)$

- Query time: $O(\log n + (1/\varepsilon)^d)$
- Works well in practice
ANN Searching with kd-trees

- **Preprocessing:** $O(n \log n)$ time, $O(n)$ space
- **Query Processing:**
  - Locate the cell containing $q$
  - Establish initial search radius
  - Visit cells in increasing order of distance
  - Stop when: $\text{cell-dist} > \text{NN-dist}/(1 + \varepsilon)$

- **Query time:** $O(\log n + (1/\varepsilon)^d)$
- **Works well in practice**
ANN Searching with kd-trees

- **Preprocessing:** $O(n \log n)$ time, $O(n)$ space
- **Query Processing:**
  - Locate the cell containing $q$
  - Establish initial search radius
  - Visit cells in increasing order of distance
  - Stop when: \( \text{cell-dist} > \text{NN-dist}/(1 + \varepsilon) \)

- **Query time:** $O(\log n + (1/\varepsilon)^d)$
- Works well in practice
ANN Searching with kd-trees

- **Preprocessing:** $O(n \log n)$ time, $O(n)$ space
- **Query Processing:**
  - Locate the cell containing $q$
  - Establish initial search radius
  - Visit cells in increasing order of distance
  - Stop when: $\text{cell-dist} > \text{NN-dist}/(1 + \varepsilon)$

- **Query time:** $O(\log n + (1/\varepsilon)^d)$
- Works well in practice
Approximate Voronoi Diagrams

**Trade-offs:** More space but lower query times?

**Approximate Voronoi Diagram (AVD)**
- Quadtree subdivision into cells
- Each cell stores a representative, \( r \in P \), such that \( r \) is an \( \varepsilon \)-ANN of any point \( q \) in the cell

**Har-Peled (2001):**
Given a set of \( n \) points in \( \mathbb{R}^d \), \( \varepsilon \)-approximate nearest neighbor queries can be answered in space \( \tilde{O}(n/\varepsilon^{d-1}) \) and in time \( O(\log(n/\varepsilon)) \)
Approximate Voronoi Diagrams

**Trade-offs:** More space but lower query times?

**Approximate Voronoi Diagram (AVD)**
- Quadtree subdivision into cells
- Each cell stores a representative, \( r \in P \), such that \( r \) is an \( \varepsilon \)-ANN of any point \( q \) in the cell

Har-Peled (2001):
Given a set of \( n \) points in \( \mathbb{R}^d \), \( \varepsilon \)-approximate nearest neighbor queries can be answered in space \( \tilde{O}(n/\varepsilon^{d-1}) \) and in time \( O(\log(n/\varepsilon)) \)
Space-Time Tradeoffs

Multi-Rep AVDs [Arya, Malamatos (2002)]

- Quadtree subdivision into cells
- Each cell stores up to $t$ representatives, $\{r_1, \ldots, r_t\} \in P$
- Given any point $q$ in the cell, at least one rep is an $\varepsilon$-ANN of $q$

- Increase $t \Rightarrow$ decrease space, increase query time
- Storage can be prohibitive in practice
Lifting and Distances

- Project a point \( p \) vertically to \( p^\uparrow \) on a paraboloid \( \Psi \).
- Let \( h \) be the tangent hyperplane at \( p^\uparrow \).
- For any point \( q \) at distance \( \delta \) from \( p \), the vertical distance between \( \Psi \) and \( h \) is \( \delta^2 \).

Lifting and Voronoi Diagrams

- Lift the points of \( P \) vertically to \( \Psi \).
- Intersect their tangent upper halfspaces.
- The projected skeleton of the resulting polytope is the Voronoi diagram of \( P \).
Lifting and Distances

- Project a point $p$ vertically to $p^\uparrow$ on a paraboloid $\Psi$
- Let $h$ be the tangent hyperplane at $p^\uparrow$
- For any point $q$ at distance $\delta$ from $p$, the vertical distance between $\Psi$ and $h$ is $\delta^2$

Lifting and Voronoi Diagrams

- Lift the points of $P$ vertically to $\Psi$
- Intersect their tangent upper halfspaces
- The projected skeleton of the resulting polytope is the Voronoi diagram of $P$
Lifting and Distances

- Project a point $p$ vertically to $p^\uparrow$ on a paraboloid $\Psi$.
- Let $h$ be the tangent hyperplane at $p^\uparrow$.
- For any point $q$ at distance $\delta$ from $p$, the vertical distance between $\Psi$ and $h$ is $\delta^2$.

Lifting and Voronoi Diagrams

- Lift the points of $P$ vertically to $\Psi$.
- Intersect their tangent upper halfspaces.
- The projected skeleton of the resulting polytope is the Voronoi diagram of $P$. 
Lifting and Distances

- Project a point $p$ vertically to $p^{\uparrow}$ on a paraboloid $\Psi$
- Let $h$ be the tangent hyperplane at $p^{\uparrow}$
- For any point $q$ at distance $\delta$ from $p$, the vertical distance between $\Psi$ and $h$ is $\delta^2$

Lifting and Voronoi Diagrams

- Lift the points of $P$ vertically to $\Psi$
- Intersect their tangent upper halfspaces
- The projected skeleton of the resulting polytope is the Voronoi diagram of $P$
Lifting and Voronoi Diagrams

Lift the points of $P$ to $\Psi$, take the upper envelope of the tangent hyperplanes, and project the skeleton back onto the plane. The result is the Voronoi diagram of $P$.

Intuition: Improved representations of polytopes lead to improvements for ANN
Polytope Membership Queries

Given a polytope $K$ in $\mathbb{R}^d$, preprocess $K$ to answer membership queries:

Given a point $q \in \mathbb{R}^d$, is $q \in K$?

Assumptions:
- Dimension $d$ is a constant
- $K$ given as intersection of $n$ halfspaces

Dual: Halfspace emptiness searching [Matoušek (1992)]
- $d \leq 3 \Rightarrow$ Space: $O(n)$, Query time: $O(\log n)$
- $d \geq 4 \Rightarrow$ Space: $O(n^{\lfloor d/2 \rfloor})$, Query time: $O(\log n)$
Approximate Polytope Membership Queries

**ε-APM Queries:**
- Given an approximation parameter $\varepsilon > 0$ (at preprocessing time)
- Assume the polytope scaled to unit diameter
- If the query point's distance from $K$:
  - $0 \Rightarrow$ Inside
  - $> \varepsilon \Rightarrow$ Outside
  - Otherwise: Either answer is acceptable

**Arya, da Fonseca, Mount [SODA 2017]**

Query time: $O(\log \frac{1}{\varepsilon})$ ← optimal
Storage: $O(1/\varepsilon^{(d-1)/2})$ ← optimal
Approximate Polytope Membership Queries

\(\varepsilon\)-APM Queries:

- Given an approximation parameter \(\varepsilon > 0\) (at preprocessing time)
- Assume the polytope scaled to unit diameter
- If the query point’s distance from \(K\):
  - \(0 \Rightarrow \) Inside
  - \(\varepsilon \Rightarrow \) Outside
  - Otherwise: Either answer is acceptable

Arya, da Fonesca, Mount [SODA 2017]

Query time: \(O(\log \frac{1}{\varepsilon})\) ← optimal
Storage: \(O\left(\frac{1}{\varepsilon^{(d-1)/2}}\right)\) ← optimal
Polytope Approximation and Ray Shooting Queries

Ray shooting preliminaries
Data structure for APM based on ray shooting
Polytope Approximation and Ray Shooting Queries (3)

Projective transformation for vertical ray shooting
State-of-the-art in ANN

reps = \{p_1, p_2\}

Nearly two decades of work on this problem
Matt Might, The Illustrated Guide to a Ph.D.
http://matt.might.net/articles/phd-school-in-pictures/
Intuition - Quadtree Search

Quadtree-based query approach:

- **Preprocessing:** Build a quadtree, subdividing each node that cannot be resolved as being inside or outside
- **Stop at diameter** $\varepsilon$
- **Query:** Find the leaf node containing $q$ and return its label

**Analysis:**

Query time: $O(\log \frac{1}{\varepsilon})$ (Quadtree descent)

Storage: $O(1/\varepsilon^{d-1})$ (Number of leaves)
Intuition - Quadtree Search

Quadtree-based query approach:

- **Preprocessing:** Build a quadtree, subdividing each node that cannot be resolved as being inside or outside
- **Stop at diameter** $\varepsilon$
- **Query:** Find the leaf node containing $q$ and return its label

Analysis:

**Query time:** $O(\log \frac{1}{\varepsilon})$ (Quadtree descent)

**Storage:** $O(1/\varepsilon^{d-1})$ (Number of leaves)
Intuition - Quadtree Search

Quadtree-based query approach:

- **Preprocessing**: Build a quadtree, subdividing each node that cannot be resolved as being inside or outside.
- **Stop at diameter** $\varepsilon$.
- **Query**: Find the leaf node containing $q$ and return its label.

**Analysis**:

- **Query time**: $O(\log \frac{1}{\varepsilon})$ (Quadtree descent).
- **Storage**: $O(1/\varepsilon^{d-1})$ (Number of leaves).
Intuition - Quadtree Search

Quadtree-based query approach:

- **Preprocessing:** Build a quadtree, subdividing each node that cannot be resolved as being inside or outside
- **Stop at diameter** \( \varepsilon \)
- **Query:** Find the leaf node containing \( q \) and return its label

**Analysis:**

Query time: \( O(\log \frac{1}{\varepsilon}) \) (Quadtree descent)

Storage: \( O(1/\varepsilon^{d-1}) \) (Number of leaves)
Intuition - Quadtree Search

Quadtree-based query approach:

- Preprocessing: Build a quadtree, subdividing each node that cannot be resolved as being inside or outside
- Stop at diameter $\varepsilon$
- Query: Find the leaf node containing $q$ and return its label

Analysis:

Query time: $O(\log \frac{1}{\varepsilon})$ (Quadtree descent)
Storage: $O(1/\varepsilon^{d-1})$ (Number of leaves)
Intuition - Quadtree Search

Quadtree-based query approach:

- Preprocessing: Build a quadtree, subdividing each node that cannot be resolved as being inside or outside
- Stop at diameter $\varepsilon$
- Query: Find the leaf node containing $q$ and return its label

Analysis:

Query time: $O(\log \frac{1}{\varepsilon})$ (Quadtree descent)
Storage: $O(1/\varepsilon^{d-1})$ (Number of leaves)
Intuition - Quadtree Search

Quadtree-based query approach:

- Preprocessing: Build a quadtree, subdividing each node that cannot be resolved as being inside or outside
- Stop at diameter $\varepsilon$
- Query: Find the leaf node containing $q$ and return its label

Analysis:

Query time: $O(\log \frac{1}{\varepsilon})$ (Quadtree descent)
Storage: $O(1/\varepsilon^{d-1})$ (Number of leaves)
Intuition - Quadtree Search

Quadtree-based query approach:
- Preprocessing: Build a quadtree, subdividing each node that cannot be resolved as being inside or outside
- Stop at diameter $\varepsilon$
- Query: Find the leaf node containing $q$ and return its label

Analysis:
Query time: $O(\log \frac{1}{\varepsilon})$ (Quadtree descent)
Storage: $O(1/\varepsilon^{d-1})$ (Number of leaves)
Intuition - Hierarchy of Covers by Balls

Hierarchy of covering balls:

- **Preprocessing:** Cover $K$ by balls of diameter $1, \frac{1}{2}, \frac{1}{4}, \ldots, \varepsilon$
- **DAG Structure:** Each ball stores pointers to overlapping balls at next level
- **Query:** Find any ball at each level that contains $q$. If none ⇒ “outside”.
- **Need only check $O(1)$ balls that overlap previous**

**Analysis:**

- **Query:** $O\left(\log \frac{1}{\varepsilon}\right)$ (Log depth, constant degree)
- **Storage:** $O\left(\frac{1}{\varepsilon^{d-1}}\right)$ (Number of leaves)
Intuition - Hierarchy of Covers by Balls

Hierarchy of covering balls:

- **Preprocessing:** Cover $K$ by balls of diameter $1, \frac{1}{2}, \frac{1}{4}, \ldots, \varepsilon$
- **DAG Structure:** Each ball stores pointers to overlapping balls at next level
- **Query:** Find any ball at each level that contains $q$. If none $\Rightarrow$ “outside”.
- **Need only check $O(1)$ balls that overlap previous**

**Analysis:**

- **Query:** $O(\log \frac{1}{\varepsilon})$ (Log depth, constant degree)
- **Storage:** $O(1/\varepsilon^{d-1})$ (Number of leaves)
Intuition - Hierarchy of Covers by Balls

Hierarchy of covering balls:

- **Preprocessing:** Cover $K$ by balls of diameter $1, \frac{1}{2}, \frac{1}{4}, \ldots, \varepsilon$
- **DAG Structure:** Each ball stores pointers to overlapping balls at next level
- **Query:** Find any ball at each level that contains $q$. If none $\Rightarrow$ “outside”.
- **Need only check** $O(1)$ balls that overlap previous

**Analysis:**

Query: $O(\log \frac{1}{\varepsilon})$ (Log depth, constant degree)
Storage: $O(1/\varepsilon^{d-1})$ (Number of leaves)
Intuition - Hierarchy of Covers by Balls

Hierarchy of covering balls:

- **Preprocessing**: Cover $K$ by balls of diameter $\frac{1}{\varepsilon}, \frac{1}{2}, \frac{1}{4}, \ldots, \varepsilon$
- **DAG Structure**: Each ball stores pointers to overlapping balls at next level
- **Query**: Find any ball at each level that contains $q$. If none $\Rightarrow$ “outside”.
- **Need only check $O(1)$ balls that overlap previous**

**Analysis:**

**Query**: $O(\log \frac{1}{\varepsilon})$ (Log depth, constant degree)

**Storage**: $O(1/\varepsilon^{d-1})$ (Number of leaves)
Intuition - Hierarchy of Covers by Balls

Hierarchy of covering balls:

- **Preprocessing:** Cover $K$ by balls of diameter $1, \frac{1}{2}, \frac{1}{4}, \ldots, \varepsilon$
- **DAG Structure:** Each ball stores pointers to overlapping balls at next level
- **Query:** Find any ball at each level that contains $q$. If none $\Rightarrow$ “outside”.
- **Need only check $O(1)$ balls that overlap previous**

**Analysis:**

Query: $O(\log \frac{1}{\varepsilon})$ (Log depth, constant degree)
Storage: $O(1/\varepsilon^{d-1})$ (Number of leaves)
Intuition - Hierarchy of Covers by Balls

Hierarchy of covering balls:

- Preprocessing: Cover $K$ by balls of diameter $1, \frac{1}{2}, \frac{1}{4}, \ldots, \varepsilon$
- DAG Structure: Each ball stores pointers to overlapping balls at next level
- Query: Find any ball at each level that contains $q$. If none $\Rightarrow$ “outside”.
- Need only check $O(1)$ balls that overlap previous

Analysis:

Query: $O(\log \frac{1}{\varepsilon})$ (Log depth, constant degree)
Storage: $O(1/\varepsilon^{d-1})$ (Number of leaves)
Intuition - Hierarchy of Covers by Balls

Hierarchy of covering balls:

- Preprocessing: Cover $K$ by balls of diameter $1, \frac{1}{2}, \frac{1}{4}, \ldots, \varepsilon$
- DAG Structure: Each ball stores pointers to overlapping balls at next level
- Query: Find any ball at each level that contains $q$. If none $\Rightarrow$ “outside”.
- Need only check $O(1)$ balls that overlap previous

Analysis:

Query: $O(\log \frac{1}{\varepsilon})$ (Log depth, constant degree)
Storage: $O(1/\varepsilon^{d-1})$ (Number of leaves)
Intuition - Hierarchy of Covers by Balls

Hierarchy of covering balls:

- **Preprocessing**: Cover $K$ by balls of diameter $1, \frac{1}{2}, \frac{1}{4}, \ldots, \varepsilon$
- **DAG Structure**: Each ball stores pointers to overlapping balls at next level
- **Query**: Find any ball at each level that contains $q$. If none $\Rightarrow$ “outside”.
- Need only check $O(1)$ balls that overlap previous

**Analysis:**

- **Query**: $O(\log \frac{1}{\varepsilon})$ (Log depth, constant degree)
- **Storage**: $O(1/\varepsilon^{d-1})$ (Number of leaves)
Hierarchy of covering balls:

- Preprocessing: Cover \( K \) by balls of diameter \( 1, \frac{1}{2}, \frac{1}{4}, \ldots, \varepsilon \)

- DAG Structure: Each ball stores pointers to overlapping balls at next level

- Query: Find any ball at each level that contains \( q \). If none \( \Rightarrow \) “outside”.

- Need only check \( O(1) \) balls that overlap previous

Analysis:

Query: \( O(\log \frac{1}{\varepsilon}) \) (Log depth, constant degree)

Storage: \( O(1/\varepsilon^{d-1}) \) (Number of leaves)
Intuition - Hierarchy of Covers by Balls

Hierarchy of covering balls:

- **Preprocessing:** Cover $K$ by balls of diameter $1, \frac{1}{2}, \frac{1}{4}, \ldots, \varepsilon$
- **DAG Structure:** Each ball stores pointers to overlapping balls at next level
- **Query:** Find any ball at each level that contains $q$. If none $\Rightarrow$ “outside”.
- Need only check $O(1)$ balls that overlap previous

**Analysis:**

**Query:** $O(\log \frac{1}{\varepsilon})$ (Log depth, constant degree)
**Storage:** $O(1/\varepsilon^{d-1})$ (Number of leaves)
Intuition - Hierarchy of Covers by Balls

Hierarchy of covering balls:

- **Preprocessing:** Cover $K$ by balls of diameter $1, \frac{1}{2}, \frac{1}{4}, \ldots, \varepsilon$
- **DAG Structure:** Each ball stores pointers to overlapping balls at next level
- **Query:** Find any ball at each level that contains $q$. If none $\Rightarrow$ “outside”.
- **Need only check** $O(1)$ balls that overlap previous

**Analysis:**

- **Query:** $O(\log \frac{1}{\varepsilon})$ (Log depth, constant degree)
- **Storage:** $O(1/\varepsilon^{d-1})$ (Number of leaves)
Intuition - Hierarchy of Covers by Balls

Hierarchy of covering balls:

- Preprocessing: Cover $K$ by balls of diameter $1, \frac{1}{2}, \frac{1}{4}, \ldots, \varepsilon$
- DAG Structure: Each ball stores pointers to overlapping balls at next level
- Query: Find any ball at each level that contains $q$. If none $\Rightarrow$ “outside”.
- Need only check $O(1)$ balls that overlap previous

Analysis:

Query: $O(\log \frac{1}{\varepsilon})$ (Log depth, constant degree)
Storage: $O(1/\varepsilon^{d-1})$ (Number of leaves)
Macbeath Regions

Want cells that conform to $K$’s shape

Macbeath Region [Macbeath (1952)]

Given convex body $K$, $x \in K$, and $\lambda > 0$:

- $M^\lambda_K(x) = x + \lambda((K - x) \cap (x - K))$
- $M_K(x) = M^1_K(x)$: Intersection of $K$ and $K$’s reflection around $x$
- $M^\lambda_K(x)$: Scaling of $M_K(x)$ by factor $\lambda$

Will omit $K$ when clear
Macbeath Regions

Want cells that conform to $K$’s shape

**Macbeath Region [Macbeath (1952)]**

Given convex body $K$, $x \in K$, and $\lambda > 0$:

- $M^\lambda_K(x) = x + \lambda((K - x) \cap (x - K))$
- $M_K(x) = M^1_K(x)$: Intersection of $K$ and $K$’s reflection around $x$
- $M^\lambda_K(x)$: Scaling of $M_K(x)$ by factor $\lambda$

Will omit $K$ when clear
Macbeath Regions

Want cells that conform to $K$’s shape

**Macbeath Region [Macbeath (1952)]**

Given convex body $K$, $x \in K$, and $\lambda > 0$:

- $M^\lambda_K(x) = x + \lambda((K - x) \cap (x - K))$
- $M_K(x) = M^1_K(x)$: Intersection of $K$ and $K$’s reflection around $x$
- $M^\lambda_K(x)$: Scaling of $M_K(x)$ by factor $\lambda$

Will omit $K$ when clear
Macbeath Regions

Want cells that conform to $K$’s shape

**Macbeath Region [Macbeath (1952)]**

Given convex body $K$, $x \in K$, and $\lambda > 0$:

- $M^\lambda_K(x) = x + \lambda((K - x) \cap (x - K))$
- $M_K(x) = M^1_K(x)$: Intersection of $K$ and $K$’s reflection around $x$
- $M^\lambda_K(x)$: Scaling of $M_K(x)$ by factor $\lambda$

Will omit $K$ when clear
Macbeath Regions

Want cells that conform to $K$’s shape

**Macbeath Region [Macbeath (1952)]**

Given convex body $K$, $x \in K$, and $\lambda > 0$:

- $M^\lambda_K(x) = x + \lambda((K - x) \cap (x - K))$
- $M_K(x) = M^{1/2}_K(x)$: Intersection of $K$ and $K$’s reflection around $x$
- $M^\lambda_K(x)$: Scaling of $M_K(x)$ by factor $\lambda$

Will omit $K$ when clear
Want cells that conform to $K$’s shape

**Macbeath Region [Macbeath (1952)]**

Given convex body $K$, $x \in K$, and $\lambda > 0$:

- $M_K^\lambda(x) = x + \lambda((K - x) \cap (x - K))$
- $M_K(x) = M_K^{1/2}(x)$: Intersection of $K$ and $K$’s reflection around $x$
- $M_K^\lambda(x)$: Scaling of $M_K(x)$ by factor $\lambda$

Will omit $K$ when clear
### Properties of Macbeath Regions

**Properties:**

- **Symmetry:** $M^\lambda(x)$ is convex and centrally symmetric about $x$

- **Expansion-Containment:** [Ewald et al (1970)]
  
  If for $\lambda < 1$, $M^\lambda(x)$ and $M^\lambda(y)$ intersect, then

  $$M^\lambda(y) \subseteq M^{c\lambda}(x), \ \text{where} \ c = \frac{3 + \lambda}{1 - \lambda}.$$  

**Upshot:** By expansion-containment, shrunken Macbeath regions behave “like” Euclidean balls, but they conform locally to $K$’s boundary... metric balls?
Properties of Macbeath Regions

Properties:

- **Symmetry**: $M^\lambda(x)$ is convex and centrally symmetric about $x$
- **Expansion-Containment**: [Ewald et al (1970)]
  If for $\lambda < 1$, $M^\lambda(x)$ and $M^\lambda(y)$ intersect, then
  \[
  M^\lambda(y) \subseteq M^{c\lambda}(x), \quad \text{where } c = \frac{3 + \lambda}{1 - \lambda}.
  \]

**Upshot**: By expansion-containment, shrunken Macbeath regions behave “like” Euclidean balls, but they conform locally to $K$’s boundary... metric balls?
Properties of Macbeath Regions

Properties:

- **Symmetry:** $M^\lambda(x)$ is convex and centrally symmetric about $x$

- **Expansion-Containment:** [Ewald et al (1970)]
  If for $\lambda < 1$, $M^\lambda(x)$ and $M^\lambda(y)$ intersect, then

  $$M^\lambda(y) \subseteq M^{c\lambda}(x), \quad \text{where } c = \frac{3 + \lambda}{1 - \lambda}.$$

Upshot: By expansion-containment, shrunken Macbeath regions behave “like” Euclidean balls, but they conform locally to $K$’s boundary... metric balls?
Properties of Macbeath Regions

Properties:

- **Symmetry:** $M^\lambda(x)$ is convex and centrally symmetric about $x$

- **Expansion-Containment:** [Ewald et al (1970)]
  If for $\lambda < 1$, $M^\lambda(x)$ and $M^\lambda(y)$ intersect, then

  $$M^\lambda(y) \subseteq M^{c\lambda}(x), \quad \text{where } c = \frac{3 + \lambda}{1 - \lambda}.$$

**Upshot:** By expansion-containment, shrunken Macbeath regions behave “like” Euclidean balls, but they conform locally to $K$’s boundary ...

... metric balls?
Metric Spaces

Metric Space: A set $X$ and distance measure $f : X \times X \rightarrow \mathbb{R}$ that satisfies:

- **Nonnegativity:** $f(x, y) \geq 0$, and $f(x, y) = 0$ if and only if $x = y$
- **Symmetry:** $f(x, y) = f(y, x)$
- **Triangle Inequality:** $f(x, z) \leq f(x, y) + f(y, z)$
Macbeath Regions and the Hilbert Geometry

- **Hilbert Metric**: Given $x, y \in K$, let $x'$ and $y'$ be the intersection of $\overrightarrow{xy}$ with $\partial K$. Define

  $$f_K(x, y) = \frac{1}{2} \ln \left( \frac{\|x' - y\| \|x - y'\|}{\|x' - x\| \|y - y'\|} \right)$$

- **Hilbert Ball**: $B_H(x, \delta) = \{ y \in K : f_K(x, y) \leq \delta \}$

  [Vernicos and Walsh (2016)]

For all $x \in K$ and $0 \leq \lambda < 1$:

$$B_H(x, \frac{1}{2} \ln (1 + \lambda)) \subseteq M^\lambda(x) \subseteq B_H \left( x, \frac{1}{2} \ln \frac{1 + \lambda}{1 - \lambda} \right)$$

e.g. $B_H(x, 0.091) \subseteq M^{0.2}(x) \subseteq B_H(x, 0.203), \forall x \in K$. 
Macbeath Regions and the Hilbert Geometry

- **Hilbert Metric**: Given $x, y \in K$, let $x'$ and $y'$ be the intersection of $\overline{xy}$ with $\partial K$. Define

$$f_K(x, y) = \frac{1}{2} \ln \left( \frac{\|x' - y\| \|x - y'\|}{\|x' - x\| \|y - y'\|} \right)$$

- **Hilbert Ball**: $B_H(x, \delta) = \{y \in K : f_K(x, y) \leq \delta\}$

[Vernicos and Walsh (2016)]

For all $x \in K$ and $0 \leq \lambda < 1$:

$$B_H(x, \frac{1}{2} \ln (1 + \lambda)) \subseteq M^\lambda(x) \subseteq B_H \left( x, \frac{1}{2} \ln \frac{1 + \lambda}{1 - \lambda} \right)$$

e.g. $B_H(x, 0.091) \subseteq M^{0.2}(x) \subseteq B_H(x, 0.203), \forall x \in K.$
Macbeath Regions and the Hilbert Geometry

- **Hilbert Metric**: Given $x, y \in K$, let $x'$ and $y'$ be the intersection of $\overrightarrow{xy}$ with $\partial K$. Define

  $$f_K(x, y) = \frac{1}{2} \ln \left(\frac{\|x' - y\| \|x - y'\|}{\|x' - x\| \|y - y'\|}\right)$$

- **Hilbert Ball**: $B_H(x, \delta) = \{y \in K : f_K(x, y) \leq \delta\}$

[Vernicos and Walsh (2016)]

For all $x \in K$ and $0 \leq \lambda < 1$:

$$B_H(x, \frac{1}{2} \ln (1 + \lambda)) \subseteq M^\lambda(x) \subseteq B_H \left(x, \frac{1}{2} \ln \frac{1 + \lambda}{1 - \lambda}\right)$$

e.g. $B_H(x, 0.091) \subseteq M^{0.2}(x) \subseteq B_H(x, 0.203), \forall x \in K$. 
Macbeath Regions and the Hilbert Geometry

- **Hilbert Metric**: Given \( x, y \in K \), let \( x' \) and \( y' \) be the intersection of \( \overrightarrow{xy} \) with \( \partial K \). Define

\[
f_K(x, y) = \frac{1}{2} \ln \left( \frac{\|x' - y\| \|x - y'\|}{\|x' - x\| \|y - y'\|} \right)
\]

- **Hilbert Ball**: \( B_H(x, \delta) = \{y \in K : f_K(x, y) \leq \delta\} \)

[Vernicos and Walsh (2016)]

For all \( x \in K \) and \( 0 \leq \lambda < 1 \):

\[
B_H(x, \frac{1}{2} \ln (1 + \lambda)) \subseteq M^\lambda(x) \subseteq B_H \left( x, \frac{1}{2} \ln \frac{1 + \lambda}{1 - \lambda} \right)
\]

e.g. \( B_H(x, 0.091) \subseteq M^{0.2}(x) \subseteq B_H(x, 0.203), \forall x \in K \).
Macbeath Ellipsoids

Macbeath regions can be combinatorially complex. Want a coarse approximation of low-complexity.

John ellipsoid [John (1948)]

Given a centrally symmetric convex body $M$ in $\mathbb{R}^d$, there exist ellipsoids $E_1, E_2$ such that $E_1 \subseteq M \subseteq E_2$ and $E_2$ is a $\sqrt{d}$-scaling of $E_1$.

Macbeath ellipsoid:

- $E(x)$: maximum volume ellipsoid in $M(x)$
- $E^\lambda(x)$: scaling by factor $\lambda$
- $E^\lambda(x) \subseteq M^\lambda(x) \subseteq E^\lambda\sqrt{d}(x)$
Macbeath regions can be combinatorially complex. Want a coarse approximation of low-complexity.

**John ellipsoid [John (1948)]**

Given a centrally symmetric convex body $M$ in $\mathbb{R}^d$, there exist ellipsoids $E_1, E_2$ such that $E_1 \subseteq M \subseteq E_2$ and $E_2$ is a $\sqrt{d}$-scaling of $E_1$.

**Macbeath ellipsoid:**

- $E(x)$: maximum volume ellipsoid in $M(x)$
- $E^\lambda(x)$: scaling by factor $\lambda$
- $E^\lambda(x) \subseteq M^\lambda(x) \subseteq E^{\lambda\sqrt{d}}(x)$
Macbeath Ellipsoids

Macbeath regions can be combinatorially complex. Want a coarse approximation of low-complexity.

John ellipsoid [John (1948)]

Given a centrally symmetric convex body $M$ in $\mathbb{R}^d$, there exist ellipsoids $E_1, E_2$ such that $E_1 \subseteq M \subseteq E_2$ and $E_2$ is a $\sqrt{d}$-scaling of $E_1$

Macbeath ellipsoid:

- $E(x)$: maximum volume ellipsoid in $M(x)$
- $E^\lambda(x)$: scaling by factor $\lambda$
- $E^\lambda(x) \subseteq M^\lambda(x) \subseteq E^{\lambda\sqrt{d}}(x)$
Macbeath Ellipsoids

Macbeath regions can be combinatorially complex. Want a coarse approximation of low-complexity.

**John ellipsoid [John (1948)]**

Given a centrally symmetric convex body $M$ in $\mathbb{R}^d$, there exist ellipsoids $E_1, E_2$ such that $E_1 \subseteq M \subseteq E_2$ and $E_2$ is a $\sqrt{d}$-scaling of $E_1$

**Macbeath ellipsoid:**

- $E(x)$: maximum volume ellipsoid in $M(x)$
- $E^\lambda(x)$: scaling by factor $\lambda$
- $E^\lambda(x) \subseteq M^\lambda(x) \subseteq E^{\lambda \sqrt{d}}(x)$
Delone Sets

A subset $X \subseteq \mathbb{X}$ is an:

- $\varepsilon$-packing: If the balls of radius $\varepsilon/2$ centered at every point of $X$ are disjoint
- $\varepsilon$-covering: If every point of $X$ is within distance $\varepsilon$ of some point of $X$
- $(\varepsilon_p, \varepsilon_c)$-Delone Set: If $X$ is an $\varepsilon_p$-packing and an $\varepsilon_c$-covering

We seek economical Delone sets for $K$, that fit within $K$'s $\delta$-expansion for $\delta = 1, \frac{1}{2}, \frac{1}{4}, \ldots, \varepsilon$
A subset \( X \subseteq \mathbb{X} \) is an:

- \( \varepsilon \)-packing: If the balls of radius \( \varepsilon/2 \) centered at every point of \( X \) are disjoint
- \( \varepsilon \)-covering: If every point of \( X \) is within distance \( \varepsilon \) of some point of \( X \)
- \((\varepsilon_p, \varepsilon_c)\)-Delone Set: If \( X \) is an \( \varepsilon_p \)-packing and an \( \varepsilon_c \)-covering

We seek economical Delone sets for \( K \), that fit within \( K \)'s \( \delta \)-expansion for \( \delta = 1, \frac{1}{2}, \frac{1}{4}, \ldots, \varepsilon \)
Delone Sets

A subset \( X \subseteq X \) is an:

- \( \varepsilon \)-packing: If the balls of radius \( \varepsilon/2 \) centered at every point of \( X \) are disjoint
- \( \varepsilon \)-covering: If every point of \( X \) is within distance \( \varepsilon \) of some point of \( X \)
- \((\varepsilon_p, \varepsilon_c)\)-Delone Set: If \( X \) is an \( \varepsilon_p \)-packing and an \( \varepsilon_c \)-covering

We seek economical Delone sets for \( K \), that fit within \( K \)'s \( \delta \)-expansion for \( \delta = 1, \frac{1}{2}, \frac{1}{4}, \ldots, \varepsilon \)
A subset $X \subseteq \mathbb{X}$ is an:

- **ε-packing**: If the balls of radius $\epsilon/2$ centered at every point of $X$ are disjoint
- **ε-covering**: If every point of $X$ is within distance $\epsilon$ of some point of $X$
- **($\epsilon_p, \epsilon_c$)-Delone Set**: If $X$ is an $\epsilon_p$-packing and an $\epsilon_c$-covering

We seek economical Delone sets for $K$, that fit within $K$’s $\delta$-expansion for $\delta = 1, \frac{1}{2}, \frac{1}{4}, \ldots, \epsilon$
A subset $X \subseteq X$ is an:

- $\varepsilon$-packing: If the balls of radius $\varepsilon/2$ centered at every point of $X$ are disjoint
- $\varepsilon$-covering: If every point of $X$ is within distance $\varepsilon$ of some point of $X$
- $(\varepsilon_p, \varepsilon_c)$-Delone Set: If $X$ is an $\varepsilon_p$-packing and an $\varepsilon_c$-covering

We seek economical Delone sets for $K$, that fit within $K$’s $\delta$-expansion for $\delta = 1, \frac{1}{2}, \frac{1}{4}, \ldots, \varepsilon$
Delone Sets

A subset $X \subseteq X$ is an:

- **$\varepsilon$-packing**: If the balls of radius $\varepsilon/2$ centered at every point of $X$ are disjoint
- **$\varepsilon$-covering**: If every point of $X$ is within distance $\varepsilon$ of some point of $X$
- **$(\varepsilon_p, \varepsilon_c)$-Delone Set**: If $X$ is an $\varepsilon_p$-packing and an $\varepsilon_c$-covering

We seek economical Delone sets for $K$, that fit within $K$'s $\delta$-expansion for $\delta = 1, \frac{1}{2}, \frac{1}{4}, \ldots, \varepsilon$
Delone sets from Macbeath ellipsoids:

- For $\delta > 0$, let $K_\delta$ be an expansion of $K$ by distance $\delta$.
- Let $\lambda_0$ be a small constant $(1/(4\sqrt{d} + 1))$.
- Let $X_\delta \subset K$ be a maximal set of points such that $E^{\lambda_0}(x)$ are disjoint for all $x \in X_\delta$.
- Exp-containment $\Rightarrow \bigcup_{x \in X_\delta} E^{1/2}(x)$ cover $K$.

**Macbeath-Based Delone Set**

$X_\delta$ is essentially a $(\frac{1}{2}, 2\lambda_0)$-Delone set for $K$. 
Delone sets from Macbeath ellipsoids:

- For $\delta > 0$, let $K_\delta$ be an expansion of $K$ by distance $\delta$
- Let $\lambda_0$ be a small constant ($1/(4\sqrt{d} + 1)$)
- Let $X_\delta \subset K$ be a maximal set of points such that $E^{\lambda_0}(x)$ are disjoint for all $x \in X_\delta$
- Exp-containment $\Rightarrow \bigcup_{x \in X_\delta} E^{\frac{1}{2}}(x)$ cover $K$

Macbeath-Based Delone Set

$X_\delta$ is essentially a $(\frac{1}{2}, 2\lambda_0)$-Delone set for $K$
Delone sets from Macbeath ellipsoids:

- For $\delta > 0$, let $K_\delta$ be an expansion of $K$ by distance $\delta$
- Let $\lambda_0$ be a small constant $(1/(4\sqrt{d} + 1))$
- Let $X_\delta \subset K$ be a maximal set of points such that $E^{\lambda_0}(x)$ are disjoint for all $x \in X_\delta$
- Exp-containment $\Rightarrow \bigcup_{x \in X_\delta} E^{\frac{1}{2}}(x)$ cover $K$

Macbeath-Based Delone Set

$X_\delta$ is essentially a $(\frac{1}{2}, 2\lambda_0)$-Delone set for $K$
Delone sets from Macbeath ellipsoids:

- For $\delta > 0$, let $K_\delta$ be an expansion of $K$ by distance $\delta$
- Let $\lambda_0$ be a small constant $(1/(4\sqrt{d} + 1))$
- Let $X_\delta \subset K$ be a maximal set of points such that $E^{\lambda_0}(x)$ are disjoint for all $x \in X_\delta$
- Exp-containment $\Rightarrow \bigcup_{x \in X_\delta} E^{1/2}(x)$ cover $K$

Macbeath-Based Delone Set

$X_\delta$ is essentially a $(\frac{1}{2}, 2\lambda_0)$-Delone set for $K$
Delone sets from Macbeath ellipsoids:

- For $\delta > 0$, let $K_\delta$ be an expansion of $K$ by distance $\delta$.
- Let $\lambda_0$ be a small constant ($1/(4\sqrt{d} + 1)$).
- Let $X_\delta \subset K$ be a maximal set of points such that $E^{\lambda_0}(x)$ are disjoint for all $x \in X_\delta$.
- Exp-containment $\Rightarrow \bigcup_{x \in X_\delta} E^{1/2}(x)$ cover $K$.

**Macbeath-Based Delone Set**

$X_\delta$ is essentially a $(\frac{1}{2}, 2\lambda_0)$-Delone set for $K$. 

(Ellipsoids not drawn to scale)
APM Data Structure

Preprocessing:

- Input: \( K \) and \( \varepsilon > 0 \)
- For \( i = 0, 1, \ldots \)
  - \( \delta_i \leftarrow 2^i \varepsilon \)
  - \( X_i \leftarrow \) Macbeath Delone set for \( K_{\delta_i} \)
  - Create a node at level \( i \) for each \( x \in X_i \)
  - Create child links to nodes at level \( i - 1 \)
    whose \( \frac{1}{2} \)-scale Macbeath ellipsoids overlap
- Stop when \( |E_\ell| = 1 \) (at \( \delta_\ell = O(1) \))

Query Processing:

- Descend the DAG from root (level \( \ell \)) until:
  - \( q \notin \frac{1}{2} \)-scaled child ellipsoids \( \Rightarrow \) “outside”
  - Reach leaf \( u \) \( \Rightarrow \) “inside”
APM Data Structure

Preprocessing:
- Input: $K$ and $\varepsilon > 0$
- For $i = 0, 1, \ldots$
  - $\delta_i \leftarrow 2^i \varepsilon$
  - $X_i \leftarrow$ Macbeath Delone set for $K_{\delta_i}$
  - Create a node at level $i$ for each $x \in X_i$
  - Create child links to nodes at level $i - 1$ whose $\frac{1}{2}$-scale Macbeath ellipsoids overlap
- Stop when $|E_\ell| = 1$ (at $\delta_\ell = O(1)$)

Query Processing:
- Descend the DAG from root (level $\ell$) until:
  - $q \notin \frac{1}{2}$-scaled child ellipsoids $\Rightarrow$ “outside”
  - Reach leaf $u \Rightarrow$ “inside”
APM Data Structure

Preprocessing:

- Input: $K$ and $\varepsilon > 0$
- For $i = 0, 1, \ldots$
  - $\delta_i \leftarrow 2^i \varepsilon$
  - $X_i \leftarrow$ Macbeath Delone set for $K_{\delta_i}$
  - Create a node at level $i$ for each $x \in X_i$
  - Create child links to nodes at level $i - 1$ whose $\frac{1}{2}$-scale Macbeath ellipsoids overlap
- Stop when $|E_\ell| = 1$ (at $\delta_\ell = O(1)$)

Query Processing:

- Descend the DAG from root (level $\ell$) until:
  - $q \notin \frac{1}{2}$-scaled child ellipsoids $\Rightarrow$ “outside”
  - Reach leaf $u \Rightarrow$ “inside”
APM Data Structure

Preprocessing:
- **Input:** $K$ and $\varepsilon > 0$
- For $i = 0, 1, \ldots$
  - $\delta_i \leftarrow 2^i \varepsilon$
  - $X_i \leftarrow$ Macbeath Delone set for $K_{\delta_i}$
  - Create a node at level $i$ for each $x \in X_i$
  - Create child links to nodes at level $i - 1$ whose $\frac{1}{2}$-scale Macbeath ellipsoids overlap
- Stop when $|E_\ell| = 1$ (at $\delta_\ell = O(1)$)

Query Processing:
- Descend the DAG from root (level $\ell$) until:
  - $q \notin \frac{1}{2}$-scaled child ellipsoids $\Rightarrow$ “outside”
  - Reach leaf $u$ $\Rightarrow$ “inside”
APM Data Structure

Preprocessing:
- Input: \( K \) and \( \varepsilon > 0 \)
- For \( i = 0, 1, \ldots \)
  - \( \delta_i \leftarrow 2^i \varepsilon \)
  - \( X_i \leftarrow \text{Macbeath Delone set for } K_{\delta_i} \)
  - Create a node at level \( i \) for each \( x \in X_i \)
  - Create child links to nodes at level \( i - 1 \)
    whose \( \frac{1}{2} \)-scale Macbeath ellipsoids overlap
- Stop when \( |E_\ell| = 1 \) (at \( \delta_\ell = O(1) \))

Query Processing:
- Descend the DAG from root (level \( \ell \)) until:
  - \( q \notin \frac{1}{2} \)-scaled child ellipsoids \( \Rightarrow \) “outside”
  - Reach leaf \( u \) \( \Rightarrow \) “inside”
Preprocessing:

- Input: \( K \) and \( \epsilon > 0 \)
- For \( i = 0, 1, \ldots \)
  - \( \delta_i \leftarrow 2^i \epsilon \)
  - \( X_i \leftarrow \) Macbeath Delone set for \( K_{\delta_i} \)
  - Create a node at level \( i \) for each \( x \in X_i \)
  - Create child links to nodes at level \( i - 1 \) whose \( \frac{1}{2} \)-scale Macbeath ellipsoids overlap
- Stop when \( |E_\ell| = 1 \) (at \( \delta_\ell = O(1) \))

Query Processing:

- Descend the DAG from root (level \( \ell \)) until:
  - \( q \notin \frac{1}{2} \)-scaled child ellipsoids \( \Rightarrow \) “outside”
  - Reach leaf \( u \) \( \Rightarrow \) “inside”
APM Data Structure

Preprocessing:

- Input: $K$ and $\varepsilon > 0$
- For $i = 0, 1, \ldots$
  - $\delta_i \leftarrow 2^i \varepsilon$
  - $X_i \leftarrow$ Macbeath Delone set for $K_{\delta_i}$
  - Create a node at level $i$ for each $x \in X_i$
  - Create child links to nodes at level $i - 1$ whose $\frac{1}{2}$-scale Macbeath ellipsoids overlap
- Stop when $|E_\ell| = 1$ (at $\delta_\ell = O(1)$)

Query Processing:

- Descend the DAG from root (level $\ell$) until:
  - $q \notin \frac{1}{2}$-scaled child ellipsoids $\Rightarrow$ “outside”
  - Reach leaf $u$ $\Rightarrow$ “inside”
APM Data Structure

Preprocessing:

- **Input:** $K$ and $\varepsilon > 0$
- **For** $i = 0, 1, \ldots$
  - $\delta_i \leftarrow 2^i \varepsilon$
  - $X_i \leftarrow$ Macbeath Delone set for $K_{\delta_i}$
  - Create a node at level $i$ for each $x \in X_i$
  - Create child links to nodes at level $i - 1$ whose $\frac{1}{2}$-scale Macbeath ellipsoids overlap
- **Stop when** $|E_\ell| = 1$ (at $\delta_\ell = O(1)$)

Query Processing:

- Descend the DAG from root (level $\ell$) until:
  - $q \notin \frac{1}{2}$-scaled child ellipsoids $\Rightarrow$ “outside”
  - Reach leaf $u \Rightarrow$ “inside”
APM Data Structure

Preprocessing:

- Input: $K$ and $\epsilon > 0$
- For $i = 0, 1, \ldots$:
  - $\delta_i \leftarrow 2^i \epsilon$
  - $X_i \leftarrow$ Macbeath Delone set for $K_{\delta_i}$
  - Create a node at level $i$ for each $x \in X_i$
  - Create child links to nodes at level $i - 1$ whose $\frac{1}{2}$-scale Macbeath ellipsoids overlap
- Stop when $|E_\ell| = 1$ (at $\delta_\ell = O(1)$)

Query Processing:

- Descend the DAG from root (level $\ell$) until:
  - $q \notin \frac{1}{2}$-scaled child ellipsoids $\Rightarrow$ “outside”
  - Reach leaf $u \Rightarrow$ “inside”
APM Data Structure

Preprocessing:

- Input: \( K \) and \( \varepsilon > 0 \)
- For \( i = 0, 1, \ldots \)
  - \( \delta_i \leftarrow 2^i \varepsilon \)
  - \( X_i \leftarrow \) Macbeath Delone set for \( K_{\delta_i} \)
  - Create a node at level \( i \) for each \( x \in X_i \)
  - Create child links to nodes at level \( i - 1 \)
    whose \( \frac{1}{2} \)-scale Macbeath ellipsoids overlap
- Stop when \( |E_\ell| = 1 \) (at \( \delta_\ell = O(1) \))

Query Processing:

- Descend the DAG from root (level \( \ell \)) until:
  - \( q \notin \frac{1}{2} \)-scaled child ellipsoids \( \Rightarrow \) “outside”
  - Reach leaf \( u \Rightarrow \) “inside”
APM Data Structure

Preprocessing:

- Input: $K$ and $\varepsilon > 0$
- For $i = 0, 1, \ldots$
  - $\delta_i \leftarrow 2^i \varepsilon$
  - $X_i \leftarrow$ Macbeath Delone set for $K_{\delta_i}$
  - Create a node at level $i$ for each $x \in X_i$
  - Create child links to nodes at level $i - 1$ whose $\frac{1}{2}$-scale Macbeath ellipsoids overlap
- Stop when $|E_\ell| = 1$ (at $\delta_\ell = O(1)$)

Query Processing:

- Descend the DAG from root (level $\ell$) until:
  - $q \notin \frac{1}{2}$-scaled child ellipsoids $\Rightarrow$ “outside”
  - Reach leaf $u \Rightarrow$ “inside”
APM Data Structure

Preprocessing:

- Input: \( K \) and \( \varepsilon > 0 \)
- For \( i = 0, 1, \ldots \)
  - \( \delta_i \leftarrow 2^i \varepsilon \)
  - \( X_i \leftarrow \) Macbeath Delone set for \( K_{\delta_i} \)
  - Create a node at level \( i \) for each \( x \in X_i \)
  - Create child links to nodes at level \( i - 1 \) whose \( \frac{1}{2} \)-scale Macbeath ellipsoids overlap
- Stop when \( |E_\ell| = 1 \) (at \( \delta_\ell = O(1) \))

Query Processing:

- Descend the DAG from root (level \( \ell \)) until:
  - \( q \not\in \frac{1}{2} \)-scaled child ellipsoids \( \Rightarrow \) “outside”
  - Reach leaf \( u \Rightarrow \) “inside”
APM Data Structure

Preprocessing:

- Input: \( K \) and \( \varepsilon > 0 \)
- For \( i = 0, 1, \ldots \)
  - \( \delta_i \leftarrow 2^i \varepsilon \)
  - \( X_i \leftarrow \) Macbeath Delone set for \( K_{\delta_i} \)
  - Create a node at level \( i \) for each \( x \in X_i \)
  - Create child links to nodes at level \( i - 1 \) whose \( \frac{1}{2} \)-scale Macbeath ellipsoids overlap
- Stop when \( |E_\ell| = 1 \) (at \( \delta_\ell = O(1) \))

Query Processing:

- Descend the DAG from root (level \( \ell \)) until:
  - \( q \notin \frac{1}{2} \)-scaled child ellipsoids \( \Rightarrow \) “outside”
  - Reach leaf \( u \Rightarrow \) “inside”
Analysis

- **Total Query time:** $O(\log \frac{1}{\varepsilon})$
  - Out-degree: $O(1)$ (By expansion-containment)
  - Query time per level: $O(1)$
  - Number of levels: $O(\log \frac{1}{\varepsilon})$ (From $\varepsilon$ to $O(1)$)

- **Total storage:** $O(1/\varepsilon^{(d-1)/2})$
  - Economical cap cover [AFM (2016)]: Number of Macbeath regions needed to cover $K_{\delta_i}$ is $O(1/\delta^{(d-1)/2})$
  - Storage for bottom level: $O(1/\varepsilon^{(d-1)/2})$
  - Geometric progression shows that leaf level dominates
Analysis

- **Total Query time:** $O(\log \frac{1}{\varepsilon})$
  - Out-degree: $O(1)$ (By expansion-containment)
  - Query time per level: $O(1)$
  - Number of levels: $O(\log \frac{1}{\varepsilon})$ (From $\varepsilon$ to $O(1)$)

- **Total storage:** $O(1/\varepsilon^{(d-1)/2})$
  - Economical cap cover [AFM (2016)]: Number of Macbeath regions needed to cover $K_{\delta_i}$ is $O(1/\delta^{(d-1)/2})$
  - Storage for bottom level: $O(1/\varepsilon^{(d-1)/2})$
  - Geometric progression shows that leaf level dominates
Bypassing the Lifting Transform

Implications of approximating the upper envelope

δ-expanded Voronoi Cell

\[ V_\delta(p) = \{ x \in \mathbb{R}^d : \| p - x \|^2 \leq \| p' - x \|^2 + \delta^2, \forall p' \in P \} \]
Bypassing the Lifting Transform

Implications of approximating the upper envelope

\( V_\delta(p) = \{ x \in \mathbb{R}^d : \| p - x \|^2 \leq \| p' - x \|^2 + \delta^2, \forall p' \in P \} \)
Implications of approximating the upper envelope

**δ-expanded Voronoi Cell**

\[ V_\delta(p) = \{ x \in \mathbb{R}^d : \| p - x \|^2 \leq \| p' - x \|^2 + \delta^2, \forall p' \in P \} \]

**Lemma**

\[ V_\delta(p) \] can be expressed as \( \bigcap_{p' \in P \setminus \{p\}} H_{p',\delta} \) where

\[ H_{p',\delta} = \{ x \in \mathbb{R}^d : \langle x, v_{p'} \rangle \leq a_{p'} + \delta^2 \}, \]

with \( v_{p'} = 2(p' - p) \) and \( a_{p'} = \| p' \|^2 - \| p \|^2 \).
Bypassing the Lifting Transform

Implications of approximating the upper envelope

**δ**-expanded Voronoi Cell

\[ V_\delta(p) = \{ x \in \mathbb{R}^d : \|p - x\|^2 \leq \|p' - x\|^2 + \delta^2, \forall p' \in P \} \]

**Lemma**

\[ \forall q \in V_\delta(p), \text{ if } \delta^2 \leq \|p - q\|^2 \cdot \min(\varepsilon, 1/2) \text{ then } p \text{ is an } \varepsilon\text{-ANN of } q. \]
Working with Expanded Voronoi Cells

- Macbeath regions w.r.t. expanded Voronoi cells
- Points from different cells ...

**Lemma - Expansion-Containment**

If $x, y \in \mathbb{R}^d$ such that $M_\delta^\lambda(x) \cap M_\delta^\lambda(y) \neq \emptyset$, then for any $\alpha \geq 0$ and $\beta = \frac{2+\alpha(1+\lambda)}{1-\lambda}$, $M^{\alpha\lambda}(y) \subseteq M^{2\beta\lambda}(x)$. 
Working with Expanded Voronoi Cells

- Macbeath regions w.r.t. expanded Voronoi cells
- Points from different cells ...

**Lemma - Expansion-Containment**

If \( x, y \in \mathbb{R}^d \) such that \( M_\delta^\lambda(x) \cap M_\delta^\lambda(y) \neq \emptyset \), then for any \( \alpha \geq 0 \) and \( \beta = \frac{2+\alpha(1+\lambda)}{1-\lambda} \), \( M_\alpha^\lambda(y) \subseteq M_\beta^\lambda(x) \).
Working with Expanded Voronoi Cells

- Macbeath regions w.r.t. expanded Voronoi cells
- Points from different cells ...

Lemma - Expansion-Containment

If \( x, y \in \mathbb{R}^d \) such that \( M^\lambda_\delta(x) \cap M^\lambda_\delta(y) \neq \emptyset \), then for any \( \alpha \geq 0 \) and \( \beta = \frac{2+\alpha(1+\lambda)}{1-\lambda} \), \( M^{\alpha\lambda}(y) \subseteq M^{2\beta\lambda}(x) \).
New Data Structure for ANN

**(r_{min}, r_{max})\text{-restricted} \varepsilon\text{-ANN queries:}**

- If \(d(q, P) > r_{max}\) \(\Rightarrow\) Outside
- If \(d(q, P) \leq r_{min}\) \(\Rightarrow\) Any \(p'\) with \(d(p', q) < r_{min}\)
- Otherwise: return an \(\varepsilon\text{-ANN for } q\)

**Layers**

Setting \(\gamma_0 = r_{min}, \gamma_i = 2^i \gamma_0\) and \(\hat{\gamma}_i = \min(\gamma_i, r_{max})\)

\[L_i(P) = \{x \in \mathbb{R}^d : \text{dist}(x, P) \leq \hat{\gamma}_{i+1}\}\]
(\(r_{\text{min}}, r_{\text{max}}\))-restricted \(\varepsilon\)-ANN queries:

- If \(d(q, P) > r_{\text{max}}\) \(\Rightarrow\) Outside
- If \(d(q, P) \leq r_{\text{min}}\) \(\Rightarrow\) Any \(p'\) with \(d(p', q) < r_{\text{min}}\)
- Otherwise: return an \(\varepsilon\)-ANN for \(q\)

**Layers**

Setting \(\gamma_0 = r_{\text{min}}, \gamma_i = 2^i \gamma_0\) and \(\hat{\gamma}_i = \min(\gamma_i, r_{\text{max}})\)

\[
L_i(P) = \{x \in \mathbb{R}^d : \text{dist}(x, P) \leq \hat{\gamma}_{i+1}\}
\]
Set $r_{\text{min}} = \delta_{\text{min}}/2$, $r_{\text{max}} = \delta_{\text{max}}/\varepsilon$ and

$\Phi(P) = \delta_{\text{max}}/\delta_{\text{min}}$.

**Theorem**

Given an $n$-element point set $P \subset \mathbb{R}^d$ and $\varepsilon > 0$, there exists a DAG structure of height $\ell = O(\log \frac{\Phi(P)}{\varepsilon})$ that can answer $\varepsilon$-ANN queries in time $O(\ell)$ space $O(\ell n/\varepsilon^{(d-1)/2})$.

Remove $\Phi(P)$ using ideas from [Har-Peled (2001)].
New Data Structure for ANN

Set $r_{\text{min}} = \delta_{\text{min}}/2$, $r_{\text{max}} = \delta_{\text{max}}/\varepsilon$ and $
abla(P) = \delta_{\text{max}}/\delta_{\text{min}}$.

**Theorem**

Given an $n$-element point set $P \subset \mathbb{R}^d$ and $\varepsilon > 0$, there exists a DAG structure of height $\ell = O\left(\log \frac{\nabla(P)}{\varepsilon}\right)$ that can answer $\varepsilon$-ANN queries in time $O(\ell)$ space $O(\ell n/\varepsilon^{(d-1)/2})$.

Remove $\nabla(P)$ using ideas from [Har-Peled (2001)].
Set \( r_{\text{min}} = \frac{\delta_{\text{min}}}{2}, \ r_{\text{max}} = \frac{\delta_{\text{max}}}{\varepsilon} \) and 
\( \Phi(P) = \frac{\delta_{\text{max}}}{\delta_{\text{min}}} \).

**Theorem**

Given an \( n \)-element point set \( P \subset \mathbb{R}^d \) and \( \varepsilon > 0 \), there exists a DAG structure of height 
\( \ell = O\left( \log \frac{\Phi(P)}{\varepsilon} \right) \) that can answer \( \varepsilon \)-ANN queries in 
time \( O(\ell) \) space \( O(\ell n/\varepsilon^{(d-1)/2}) \).

Remove \( \Phi(P) \) using ideas from [Har-Peled (2001)].
Concluding Remarks

- Much simpler and optimal solution to $\varepsilon$-APM queries:
  - Query time: $O(\log \frac{1}{\varepsilon})$
  - Storage: $O(\frac{1}{\varepsilon^{(d-1)/2}})$

- Much simpler data structure for $\varepsilon$-ANN queries
  - Extra log factor ..

- Goals
  - Match or improve upon state-of-the-art
  - Other metrics
Concluding Remarks

- Much simpler and optimal solution to $\varepsilon$-APM queries:
  - Query time: $O(\log \frac{1}{\varepsilon})$
  - Storage: $O(\frac{1}{\varepsilon^{(d-1)/2}})$

- Much simpler data structure for $\varepsilon$-ANN queries
  - Extra log factor ..

- Goals
  - Match or improve upon state-of-the-art
  - Other metrics

Thank you for your attention!