Data Structures for Approximate Proximity and Range Searching

David M. Mount  
University of Maryland  
Joint work with:  
Sunil Arya (Hong Kong U. of Sci. and Tech)  
Charis Malamatos (Max Plank Inst.)
Computational Geometry: The study of efficient algorithms and data structures for discrete geometric structures: finite point sets, polyhedra, spatial subdivisions.

Spatial Data Retrieval: Given a finite set of objects (points) preprocess these objects into a data structure that supports efficient processing of some given class of queries.

Efficiency: is measured in terms of the resources used as a function of the number of entities in the structure:
- Query time
- Space (for the data structure)
- Preprocessing time (usually of secondary importance)
- Update time (for dynamic applications)

1-Dimensional Example: Binary search and binary search trees. Improves $O(n)$ brute-force search to $O(\log n)$ time.
Nearest Neighbor: Given a point set $S \subseteq \mathbb{R}^d$ and $q \in \mathbb{R}^d$, find the point $p^* \in S$ that is closest to $q$.

(Spherical) Range Queries: Given a query ball, report all the points that lie within, or the total weight of points within.

Throughout we assume Euclidean distances.
Challenges

**Curse of Dimensionality:** Many methods for geometric retrieval problems have query times that grow exponentially in dimension (assuming a fixed amount of space).

**Lower bounds:** Many retrieval problems have known lower bounds, which imply that more efficient methods cannot exist.

For example, given a parameter $m$, where $n \leq m \leq n^d$, if $O(m)$ space is used, then spherical range queries cannot be answered in substantially less than $O(n/(m^{1/d}))$ time [Brönnimann, et. al, 1993].
Approximate Proximity Search

**Approx Nearest Neighbor:** Given $\varepsilon > 0$ and $q \in \mathbb{R}^d$, a point $p \in S$ is an $\varepsilon$-nearest neighbor of $q$ if,

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

where $p^* \in S$ is the nearest neighbor of $q$. 
Approximate Proximity Search

Approx Range Searching: Let \( B(q,r) \) denote the Euclidean ball of radius \( r \) centered at \( q \). A set \( S' \) is an admissible solution to an \( \varepsilon \)-approximate range query if

\[
S \cap B(p,r(1 - \varepsilon)) \subseteq S' \subseteq S \cap B(p,r(1 + \varepsilon))
\]

Goal: Return the weight of any admissible solution.
Any convex body in $\mathbb{R}^d$ can be $\varepsilon$-approximated by a polyhedron with $(1/\varepsilon)^{(d-1)/2}$ facets [Dudley 74].

<table>
<thead>
<tr>
<th>Method</th>
<th>Query Time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact</td>
<td>$\log n$</td>
<td>$n^{d/2}$</td>
</tr>
<tr>
<td>AMNSW'98</td>
<td>$(1/\varepsilon)^d \log n$</td>
<td>$n$</td>
</tr>
<tr>
<td>Clarkson '97</td>
<td>$(1/\varepsilon)^{\frac{d-1}{2}} \log n$</td>
<td>$(1/\varepsilon)^{\frac{d-1}{2}} n\log n$</td>
</tr>
<tr>
<td>Chan '98</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Our Results on ε-NN Search

**Theorem:** (AMM02) Given a point set $S$ in $\mathbb{R}^d$, and $0 < \varepsilon \leq \frac{1}{2}$, $2 \leq \gamma \leq \frac{1}{\varepsilon}$, it is possible to build a data structure of space $O(n\gamma^{d-1}\log\gamma)$ that can answer $\varepsilon$-NN queries in $O(\log (n\gamma) + 1/(\varepsilon\gamma^{(d-1)/2}))$ time.

<table>
<thead>
<tr>
<th>$\gamma$</th>
<th>Query Time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1/\varepsilon$</td>
<td>$\log n + \log (1/\varepsilon)$</td>
<td>$n/\varepsilon^d$</td>
</tr>
<tr>
<td>$2$</td>
<td>$\log n + 1/\varepsilon^{(d-1)/2}$</td>
<td>$n$</td>
</tr>
</tbody>
</table>

**Note:** For low-space version, space is independent of $\varepsilon$, and query time is additive, not multiplicative.
Best exact approaches are based on cuttings, but results scale poorly with dimension.

<table>
<thead>
<tr>
<th></th>
<th>Query Time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exact</strong></td>
<td>log n</td>
<td>$n^d / \log^d n$</td>
</tr>
<tr>
<td>$n^{(1 - 1/d)}$</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>$\log n + n/m^{1/d}$</td>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td><strong>Approx [AM’00]</strong></td>
<td>$\log n + (1/\varepsilon)^{d-1}$</td>
<td>n</td>
</tr>
</tbody>
</table>

The approximate solution of AM’00 is based on kd-trees.
Our Results on $\varepsilon$-Range Search

**Theorem:** (AMM04) Given a point set $S$ in $\mathbb{R}^d$, and $0 < \varepsilon \leq \frac{1}{2}$, $2 \leq \gamma \leq \frac{1}{\varepsilon}$, it is possible to build a data structure of space $O(n^d \log (1/\varepsilon))$ that can answer $\varepsilon$-range queries in $O(\log (n/\varepsilon) + 1/(\varepsilon^2)^{d-1})$ time.

<table>
<thead>
<tr>
<th>$\gamma$</th>
<th>Query Time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1/\varepsilon$</td>
<td>$\log n + \log (1/\varepsilon)$</td>
<td>$(n/\varepsilon^d) \log (1/\varepsilon)$</td>
</tr>
<tr>
<td>$2$</td>
<td>$\log n + 1/\varepsilon^{(d-1)}$</td>
<td>$n \log (1/\varepsilon)$</td>
</tr>
</tbody>
</table>

**Note:** Can be used for answering approximate $k$-th nearest neighbor queries in similar time bounds.
Remainder of the Talk

- General techniques used in efficient approximate retrieval.
- Approximate Voronoi Diagram (AVD)
- Applying AVDs to range searching
- Implementation (time permitting)
Voronoi Diagrams

Given a set $S$ of $n$ point sites in $\mathbb{R}^d$. **Voronoi diagram** is a subdivision of space into regions according to which site is closest.

Use **point location** to answer NN queries.
Voronoi Diagrams: Difficulties

**High Complexity:** In dimension $d$, it may be as high as $\Theta(n^{\lceil d/2 \rceil})$.

**Computational Issues:** Geometric degeneracies and topological consistency.

**Point Location:** Optimal solutions only in 2-d.

**Question:** Are there simpler/faster methods if we are willing to approximate?
Approx Voronoi Diagrams

\(\varepsilon\text{-AVD}: \) (Har-Peled ’01)
Quadtree-like subdivision of space. Each cell stores a representative site, \(r \in S\), such that \(r\) is an \(\varepsilon\)-NN of any point \(q\) in the cell.

\(\varepsilon\text{-NN} \rightarrow \text{pt location}\)
Approx Voronoi Diagrams

Har-Peled '01: Size:

\[ O\left(\frac{n}{\varepsilon^d} \log n \left(\log \frac{n}{\varepsilon}\right)\right). \]

\(\varepsilon\)-NN Queries: Point location in a compressed quadtree in time

\[ O\left(\log \frac{n}{\varepsilon}\right). \]

Arya, Malamatos '02: Multiple representatives
Multiple Representatives

Multi-representatives:
Each cell is allowed up to $t \geq 1$ representatives.

Tradeoff: cells vs. representatives.

NN-Query: Pt. Loc. and distance comp.

$t=2$
Basic Tools: WSPDs

Separation factor: $s > 2$.

Two sets $A$ and $B$ are well-separated if they can be enclosed in spheres of radius $r$, whose centers are at distance least $sr$. 
Well-Separated Pair Decomposition (WSPD):
Given a set of $n$ points and separation factor $s$, it is possible to represent all $O(n^2)$ pairs as $O(s^d n)$ well-separated pairs. (Callahan, Kosaraju '95)
**Basic Tools: BBD Trees**

**Quadtree Box:** A box that can be obtained by repeatedly splitting the unit hypercube into \(2^d\) identical boxes.
Basic Tools: BBD Trees

BBD Tree: Given a set of $m$ quadtree boxes, we can build a BBD-tree of size $O(m)$ and height $O(\log m)$ whose induced subdivision is a refinement of the box subdivision. (AMN+98)
Separation: Intuition

The greater the separation from a set of points, the fewer representatives are needed to guarantee that one is an $\varepsilon$-NN.

1 rep

4 reps
Disjoint & Concentric Balls

**Disjoint Ball Lemma:** Given disjoint balls of radii $r_1$ and $r_2$ separated by $L$, the number of representatives needed is

$$\left( \frac{r_1 r_2}{(\varepsilon L^2)} \right)^{\frac{d-1}{2}}$$

**Concentric Ball Lemma:** Given concentric balls of radii $r$ and $\gamma r$, the number of representatives needed is

$$\frac{1}{(\varepsilon \gamma)^{\frac{d-1}{2}}}$$
Separation: Goal

Low-$\gamma$: Assume $\gamma=2$.

Goal: Subdivide space into $O(n)$ cells. For each cell of size $s$, all sites within distance $4s$ can be enclosed within a ball whose factor-2 expansion does not intersect the cell.
Construction

Create a **WSPD** with separation 4.

For each WSP, create a set of **quadtree boxes** whose sizes depend on the dist from this WSP.

Build a **BBD tree** for these boxes.
Achieving Separation

Why does this work?
Suppose that the points within the 4s expansion are not contained within a separated ball. Then there would be a well-separated pair, which would force the cell to be split.
Selecting Representatives

Two-Step Approach:

- Construct a set of $1/\varepsilon^{(d-1)/2}$ points uniform on an intermediate sphere $B$.
- Reps are the nearest neighbors of these points.
Space Reduction: Sampling

Recall that representatives come from two sources:
- From outside large ball
- From inner cluster
- No points exist in the remaining “no-man’s land”

Idea: Allow more points into no-man’s land, and make them all reps.
Space Reduction: Sampling

**Intuition:** Use a sample $S'$ of $n \varepsilon^{(d-1)/2}$ points in the basic AVD construction. We expect $O(1/\varepsilon^{(d-1)/2})$ points of $S$ to lie in no-man’s land.

**Representatives:** From outer, inner cluster, and no-man’s land.
Bisector-Sensitivity

Recall that the basic AVD construction creates quadtree boxes uniformly around each WSP.

Idea: Concentrate boxes along bisector.
Bisector-Sensitivity

Bisector Sensitive Construction: For each WSP \((A,B)\), create quadtree boxes as before, but only for those that intersect the \(A-B\) bisector.
AVDs and Range Searching

**Adaptation:** AVDs can be adapted to perform range searching. Rather than using just the leaf nodes, **internal nodes** are used as well for answering queries, where the query size is roughly $\gamma$ times the size of the associated cell.

**Auxiliary information:** Each internal node stores information about surrounding region in order to answer queries.
Polar kd-trees

With spherical ranges there are two sources of approximation error:
- radial distance from center
- angular error

It is possible to tolerate greater angular error than radial error.
**Polar kd-trees**

**Polar kd-tree:** Build a collection of hierarchical spatial subdivisions, based on a polar representation of points relative to some local center.
Conclusions

\( \varepsilon \)-AVD: A spatial subdivision in which \( \varepsilon \)-NN queries reduce to point location.

**Space Efficiency:** Through deterministic sampling and bisector sensitivity.
- \( O(\log n + 1/\varepsilon^{(d-1)/2}) \) time
- \( O(n) \) space

**AVDs for Other Problems:** AVDs for other objects? AVD-like structures for interpolation?

**Approximating Voronoi Cells:** Some initial results by Arya and Vigneron.
The End

Thank you!
The WSPD construction is not very practical:
- Large constants.
- Bottom-up construction (must know all the AVDs to construct any part).
- Further study is warranted.

**Partial construction:** Given the size of the AVD, it is useful to build/rebuild portions of the structure. Need for top-down construction.
Top-Down Construction

**Input:** Point set $S$. Error factor $\varepsilon$, and number of representatives $t$.

**Basis:** Start with bounding hypercube as cell and all points of $S$ as candidate nearest neighbors.

**Recursive step:** Given a quadtree cell $C$, and a collection of candidate nearest neighbors $U$.
- Prune from $U$ all points that cannot be an $\varepsilon$-NN to any point of $C$.
- If $|U| \leq t$, then done. Otherwise, split $C$ and recurse.
How to Prune?

Let $p$ in $U$ be the closest candidate to the center of the cell. Let $r$ be some other candidate.

If for all $x$ in $C$, if $\text{dist}(x, p) \leq (1+\varepsilon)\text{dist}(x, r)$ then prune $r$.

This can be solved numerically.
Results: Cells vs n,t
Results: cells vs $\varepsilon$