Open Problem: Dynamic Planar Nearest Neighbors

CSCE 620

Problem 63 from the Open Problems Project
http://maven.smith.edu/~orourke/TOPP/P63.html#Problem.63

Aditya Mahadevan
http://parasol.tamu.edu
Nearest Neighbor Search

- Problem of finding closest points in a metric space
- Given a point set $S$, and a query $q$, find the point $s \in S$ that is closest to $q$
- Planar nearest neighbors - where $S$ is a set of points on a 2D plane
- Naïve case: $O(n)$ complexity
Applications

- Classification
  - Statistical classification, e.g. online clustering
  - Pattern recognition (through classification)
- Robotics
- Relations to other problems
  - dynamic convex hull
  - Nearest neighbors for moving points? (delete a point from its previous position, re-insert at new position)
Nearest Neighbors, Voronoi Diagram

- Voronoi diagram of $S$, $Vor(S)$ – partition of the plane into $|S|$ partitions such that each partition contains a point $s$ ("site"), and all points on the plane closer to $s$ than any other point in $S$
- Nearest neighbor of query point $q$ – site representing the Voronoi partition containing $q$
Voronoi Diagram and Convex Hull

- Duality between 2D Voronoi Diagram and 3D Convex Hull
- Can get Voronoi diagram of a set of points using convex hull algorithms
  - Map each 2D point to 3D using the function \(-z = 2p_x x + 2p_y y - (p_x^2 + p_y^2)\) for each point \(p\) (plane in 3D)
  - Project the plane intersections in 2D – gives Voronoi diagram
  - Thus Voronoi Diagram problem in 2D is reducible to Convex Hull problem in 3D

Nearest Neighbors and Convex Hull

- Lower envelope of H has 1-to-1 correspondence with upper convex hull of P
- Nearest neighbor (from set P) of a point $q$ can be found by intersecting the vertical line through $q$ with H – the plane that intersects will correspond to the nearest neighbor
- Hence nearest neighbor search in 2D is reducible to convex hull in 3D
The Open Problem

- Is there a data structure maintaining a set of $n$ points in the plane subject to insertions, deletions and nearest-neighbor queries in $O(\log n)$ time?
- Reduces to maintaining the convex hull of a set of $n$ points in 3D subject to insertions, deletions and extreme-point queries.
Approaches – Bentley and Saxe

- **Bentley and Saxe ’80** – Decomposable Searching Problems: Static-to-Dynamic Transformation
  - Nearest neighbors as a decomposable search problem
- **Result**: $O(\log^2 n)$ amortized bound for updates and queries when insertion is allowed, but not deletion
- Considered general transformations for converting static structures to dynamic structures (i.e. those supporting insertions, deletions)
- General approach to decomposable search problems: **dynamic structure that maintains a set of static structures**, such that the cardinality of each is a power of 2
  - Concept is borrowed in subsequent approaches
Approaches – Bentley and Saxe

- Introduced a dynamic data structure ("DNN"), which used a static structure from Lipton and Tarjan ("LT")
- Allowing insertion is relatively easy using their methods, but efficient deletion is impossible in the general case
- When a new point is inserted, existing static structures may periodically be deleted and replaced by new ones
  - E.g. when there are 7 elements represented by 3 structures (red) and an 8\textsuperscript{th} one is inserted, structures are replaced by a single larger structure (blue)
  - this amortizes the cost of insertions to $O(\log^2 n)$ (perhaps since $O(\log i)$ structures are affected once every $O(\log i)$ inserts)

History of LT structures for repeated insertions
Source: Bentley&Saxe ’80
Approaches – Agarwal & Matousek

- **Agarwal and Matousek ’95** – Dynamic Half-Space Range Reporting and Its Applications
  - Two structures
    - Updates in $O(\log^2 n)$ time, queries in $O(n^\varepsilon)$ time
    - Updates in $O(n^\varepsilon)$ amortized time, queries in $O(\log n)$ w.c. time
  - Adapted efficient static range query structures to support insertions and deletions
    - Clarkson ’88
    - Matousek ’92
Approaches - Agarwal & Matousek

- **Structure for logarithmic-time updates**
  - Use a partition tree such that at every level of the tree, each point is in exactly one node
  - Deletion of a point occurs by removing the point from each node that contains it, starting from root
  - Amortize cost of delete to $O(\log^2 n)$ by periodically reconstructing the entire data structure after some deletions
  - For insertion, use dynamization techniques from Bentley and Saxe
Approaches - Agarwal & Matousek

- Structure for logarithmic-time *queries*
  - Intersection of simplices with hyperplanes
  - Use deterministic counterpart of Clarkson’s randomized tree-like structure
  - Query by recursively searching for simplex that contains the query point
  - Prevent bad performance by partitioning hyperplanes into subsets of “good” hyperplanes (i.e. those that reduce the number of intersections with simplices)
  - Periodically reconstruct structure after some deletions of hyperplanes
Approaches – Timothy Chan

- **Chan 2006** – A Dynamic Data Structure for 3D Convex Hulls and 2D Nearest Neighbor Queries
- **Results**
  - Insertions in $O(\log^3 n)$ expected amortized time
  - Deletions in $O(\log^6 n)$ expected amortized time
  - Queries in $O(\log^2 n)$ worst case time
- **First method to guarantee poly-logarithmic updates and queries for arbitrary sequences of insertions and deletions**
Approaches – Timothy Chan

- Use cuttings
  - $(1/r)$-cutting of a range $\gamma$ given set of planes $H$ – collection of disjoint open cells such that union of closure of cells contains $\gamma$ and each cell intersects at most $n/r$ planes of $H$
  - Conflict list of a cell – subset of all planes of $H$ intersecting the cell

- Construct layered data structure with logarithmically many layers

- At each layer, make cutting of certain size and remove “bad” planes i.e. those that cause many conflicts

- Dynamic data structure that stores many such static structures
Possible Future Directions

- De-amortize Chan’s time bounds by known tricks
- Improve complexity of cuttings
  - E.g. take multiple independent samples to lower error probability
- Replace $n^\varepsilon$ factors of Agarwal & Matousek by poly-logarithmic factors
- Find a deterministic poly-logarithmic counterpart to Chan’s randomized method
References