Query algorithms

NN search: Branch and Bound

- [Roussopoulos+, sigmod95]:
  - At each node, priority queue, with promising MBRs, and their best and worst-case distance
- Main idea: Every face of an MBR contains at least one point of an actual spatial object.
- Prune using MINDIST and MINMAXDIST

MINDIST

- MINDIST(q, R) is the minimum distance between a point q and a rectangle R
- If the point is inside R, then MINDIST=0
- If q is outside of R, MINDIST is the distance of q to the closest point of R

MINDIST computation

- MINDIST(q,R) is the minimum distance between q and R with corner points l and u.
  - the closest point in R is at least this distance away

\[
MINDIST(q, R) = \sqrt{\sum_{i=1}^{d} (q_i - r_i)^2}
\]

\[
r_i = l_i \text{ if } q_i < l_i
\]
\[
= u_i \text{ if } q_i > u_i
\]
\[
= q_i \text{ otherwise}
\]

\[
\forall p \in R, MINDIST(q, R) \leq \|q - p\|
\]
MINMAXDIST

- MINMAXDIST(q,R): for each dimension i, find the closest face, compute the distance to the farthest point on this face and take the minimum of all these distances
- MINMAXDIST(q,R) is the smallest possible upper bound of distances from q to R
- MINMAXDIST guarantees that there is at least one object in R with a distance to q smaller or equal to it.

\[ \exists p \in R \parallel (q, p) \leq \text{MINMAXDIST}(q, R) \]

MINDIST and MINMAXDIST

- MINDIST(q, R) \leq \text{NN}(q,R) \leq \text{MINMAXDIST}(q,R)

Pruning in NN search

- Downward pruning: An MBR R is discarded if there exists another R’ s.t. MINDIST(q,R) > MINMAXDIST(q,R’)
  - An object p is discarded if there exists an R’ s.t. the dist(q,p) > MINMAXDIST(q,R’)
- Upward pruning: An MBR R is discarded if an object p is found s.t. the MINDIST(q,R) > dist(q,p)

(Downward) Pruning example

- Downward pruning: An MBR R is discarded if there exists another R’ s.t. MINDIST(q,R) > MINMAXDIST(q,R’)

Spring 2013  Searching Big Data  5

Spring 2013  Searching Big Data  6

Spring 2013  Searching Big Data  7

Spring 2013  Searching Big Data  8
(Downward) Pruning example

- Downward pruning: An object $p$ is discarded if there exists an $R$ s.t. $\text{dist}(q,p) > \text{MINMAXDIST}(q,R)$

(Upward) Pruning example

- Upward pruning: An MBR $R$ is discarded if an object $p$ is found s.t. $\text{MINDIST}(q,R) > \text{dist}(q,p)$

Ordering Distance

- $\text{MINDIST}$ is an optimistic distance where $\text{MINMAXDIST}$ is a pessimistic one.

NN-search Algorithm

1. Initialize the nearest distance as infinite
2. Traverse the tree depth-first starting from the root. At each index node, sort all MBRs using an ordering metric and put them in an Active Branch List (ABL).
3. Apply pruning rules to ABL.
4. Visit the MBRs from the ABL following the order until it is empty.
5. If Leaf node, compute actual distances, compare with the best NN so far, update if necessary.
6. At the return from the recursion, use pruning rules.
7. When the ABL is empty, the NN search returns.
K-NN search

- Keep the sorted buffer of at most k current nearest neighbors
- Pruning is done using the k-th distance

Another NN search: Best-First

- Global order [HS99]
  - Maintain distance to all entries in a common Priority Queue
  - Use only MINDIST
  - Repeat
    - Inspect the next MBR in the list
    - Add the children to the list and reorder
  - Until all remaining MBRs can be pruned

Nearest Neighbor Search (NN) with R-Trees

- Best-first (BF) algorithm:

HS algorithm

- Initialize PQ (priority queue)
- InsertQueue(PQ, Root)
- While not IsEmpty(PQ)
  - R = Dequeue(PQ)
    - If R is an object
      - Report R and exit
    - If R is a leaf page node
      - For each O in R, compute the Actual-Dists, InsertQueue(PQ, O)
      - If R is an index node
        - For each MBR C, compute MINDIST, insert into PQ
Best-First vs Branch and Bound

• Best-First is the “optimal” algorithm in the sense that it visits all the necessary nodes and nothing more.
• But needs to store a large priority queue in main memory. If this becomes large, we have thrashing…
• BB uses small lists for each node. Also uses MINMAXDIST to prune some entries.

NN queries in reduced space

• 2-stage algorithm
  – D = “object” distance, d = “feature” distance, d ≤ D
  – Index structure constructed in feature space
  – Find k NNs in reduced feature space using d
  – Compute object distances
    • Find the object distance r to kth distant object
  – Perform a range query using r in feature space
    • Sort the retrieved points using D
    • Return the best k objects

Optimal multi-step algorithm [SK98]

• Find k NNs in reduced feature space.
• Maintain a priority queue Q of these points based on the object distance from query object.
• R := distance to the kth distant object in object space
• r := current range in feature space
• While r ≤ R do
  – Get next NN p in feature space
  – Insert p into priority queue Q
  – Update r and R
• Return the best k objects from Q
**Spatial Join Queries**

**Spatial Join**
- Find all parks in a city
- Find all trails that go through a forest
- Basic operation
  - find all pairs of objects that overlap
- Single-scan queries
  - nearest neighbor queries, range queries
- Multiple-scan queries
  - spatial join

**Algorithms**
- No existing index structures
  - Transform data into 1-d space [O89]
  - \( z \)-transform; sensitive to size of pixel
  - Partition-based spatial-merge join [PW96]
    - partition into tiles that can fit into memory
    - plane sweep algorithm on tiles
  - Spatial hash joins [LR96, KS97]
  - Sort data [BBKK01]
- With index structures [BKS93, HJR97]
  - k-d trees and grid files
  - R-trees
Join1(R,S)

- Tree synchronized traversal algorithm
  Join(R,S)
  Repeat
  Find a pair of intersecting entries E in R and F in S
  If R and S are leaf pages then
    add (E,F) to result-set
  Else Join(E,F)
  Until all pairs are examined

R-tree based Join [BKS93]

CPU Time Tuning

- Two ways to improve CPU time
  - Restricting the search space
  - Spatial sorting and plane sweep

Join2(R,S,IntersectedVol)

- Repeat
  - Find a pair of intersecting entries E in R and F in S that overlap with IntersectedVol
  - If R and S are leaf pages then add (E,F) to result-set
  - Else Join2(E,F,CommonEF)
- Until all pairs are examined
- 15+6 comparisons instead of 56
- In general, number of comparisons equals
  - size(R) + size(S) + relevant(R)*relevant(S)
- Reduce the product term
Reducing CPU bottleneck

Using Plane Sweep

Consider the extents along x-axis sweep a vertical line

Using Plane Sweep

Check if \((r_1,s_1)\) intersect along y-dimension; add \((r_1,s_1)\) to result
Repeat for \((r_2,s_1)\); do not add \((r_2,s_1)\)
Using Plane Sweep

Check if (r1,s2) intersect along y-dimension
Add (r1,s2) to result set

Total of 2(r1) + 1(r2) + 0(s1) + 1(s2) + 0(r3) = 4 comparisons

References

References