CONTINUOUS NEAREST NEIGHBOR SEARCH

Instructor: Cyrus Shahabi
OVERVIEW

• Introduction
• Preliminary & Related Work
• Continuous k-Nearest Neighbor Query (CkNN)
  – Definition
  – Problem Characteristics
  – R-tree algorithm
  – Query analysis
  – Complex CNN extension
• Experiments
• Discussion and Conclusion
INTRODUCTION

• Continuous Nearest Neighbor

• Why called “continuous”?
  – Nearest neighbor of every points in the trajectory
PRELIMINARY -- CONTINUOUS NEAREST NEIGHBOR

- **Data:** A set of points \( P = \{a, b, c, d, f, g, h\} \)
- **Query:** A line segment \( q = [s, e] \)
- **Result:** The nearest neighbor (NN) of every point on \( q \).
- **Result representation:** \( \{<a, [s, s_1]>, <c, [s_1, s_2]>, <f, [s_2, s_3]>, <h, [s_3, e]>, \} \)
RELATED WORK – SAMPLING

• Try to convert the continuous-NN to point-NN
  – Every point on the line -> unlimited points
  – Sampling

• Drawback:
  – Sample Rate: low -> incorrect
  – Sample Rate: high -> overhead (still cannot guarantee accuracy)
RELATED WORK – TP NN (CONT.)

• Step 1: Find the NN of the start point $s$, i.e., point $a$.
• Step 2: Use the TP technique: find the first point on the line segment $(s_1)$ where there is a change in the NN (i.e., point $c$) will become the next NN – result: $<a, [s,s_1), c>$
• Can be thought as conventional NN query, where the goal is to find the point $x$ with the minimum $dist(s, sx)$
• Step 3: Perform another TP NN to find:
• Starting from s1, the smallest distance we need to travel for the current NN (i.e., c) to change
• Repeat this until we finish the entire segment.
• Not only NN, but support k-NN
• Still overhead: \( n (= \text{split points}) \) times NN queries, multiple scans of database
CKNN - DEFINITION

• Goal: Find all split points (as well as the corresponding NN for each segment) with a single traversal.
• Split List (SL): The set of split points (including s and e).
• Each split point $s_i \in SL$ and all points in $[s_i, s_{i+1}]$ have the same NN, denoted as $s_i.NN$ (e.g., $s_1.NN$ is c, which is also the NN for all points in interval $[s_1, s_2]$)
• $s_i.NN$ (e.g., c) covers point $s_i$ ($s_1$) and interval $[s_i, s_{i+1}]$ ($[s_1, s_2]$).
• Vicinity Circle (VC): The circle that centers at split point $s_i$ with radius $\text{dist}(s_i, s_i.NN)$
Lemma 1: Given a split list $SL = \{s_0, s_1, \ldots, s_{|SL| - 1}\}$, and a new data point $p$, then: $p$ covers some point on query segment $q$ if and only if $p$ covers a split point.

Analyzing the first data point “a” (in alphabetical order)

Analyzing “b”: not in VC of $s$ and $e$, hence no point on $[s,e]$ closer to $b$ than $a$

Result: \{<$a$, [s,s1]>, <c, [s1,e]> \}
CKNN - PROBLEM CHARACTERISTICS

• Lemma 2: (Covering Continuity)
  – The split points covered by a point \( p \) are continuous.
  – Namely, if \( p \) covers split point \( s_i \) but not \( s_{i-1} \) (or \( s_{i+1} \)), then \( p \) cannot cover \( s_{i-j} \) (or \( s_{i+j} \)) for any value of \( j > 1 \).
  – Below: \( p \) cover \( S_i, S_{i+1} \) and \( S_{i+2} \) (\( p \) falls in their vicinity circles), but not \( s_{i-1}, s_{i+3} \), so no need to check \( p \) against any other split points

\[ SL = \{ s_{i-1} (\text{NN}=a), s_i (\text{NN}=b), s_{i+1} (\text{NN}=c), s_{i+2} (\text{NN}=d), s_{i+3} (\text{NN}=f) \} \]
CKNN - PROBLEM CHARACTERISTICS

- Finding new split points for b:
  - b covers $S_{i-1}$ and f covers $S_{i+2}$; so we need to find the space that NN changes from b to p and then to g

When new data point “p” arrives…

Processing point “p” …

\[
SL = \{ s_{i-1} (.NN=a), s_i (.NN=b), s_{i+1} (.NN=p), s_{i+2} (.NN=f) \}
\]

\[
SL = \{ s_{i-1} (.NN=a), s_i (.NN=b), s_{i+1} (.NN=c), s_{i+2} (.NN=d), s_{i+3} (.NN=f) \}
\]
• How about the k-NN?
• Lemma 1: Fit  ||  Lemma 2: Cannot Fit
• Eg:
  – K=3

\[ SL = \{ s_i, s_{i+1}, s_{i+2}, s_{i+3} \} \]
CKNN – R-TREE ALGORITHM

• General key notes:
  – Use branch-and-bound techniques to prune the search space.
  – R-tree traverse principle:
    • When a leaf entry (i.e., a data point) \( p \) is encountered, SL is updated if \( p \) covers any split point (i.e., \( p \) is a qualifying entry) – By Lemma 1.
    • For an intermediate entry, We visit its subtree only if it may contain any qualifying data point – Use heuristics.
  – Avoid accessing non qualified nodes
R-TREE ALGORITHM – HEURISTIC 1

- Given an intermediate entry $E$ and query segment $q$, the sub-tree of $E$ may contain qualifying points only if $\text{mindist}(E, q) < S_{\text{MAXD}}$, where $S_{\text{MAXD}}$ is the maximum distance between a split point and its NN.

$SL = \{s (\text{NN}=a), s_1 (\text{NN}=b), e (\text{NN}=b)\}$

Compute $\text{Mindist}(E, q)$
R-TREE ALGORITHM – HEURISTIC 2
(AFTER 1)

• Given an intermediate entry $E$ and query segment $q$, the subtree of $E$ must be searched if and only if there exists a split point $s_i \in SL$ such that $\text{dist}(s_i, s_i.NN) > \text{mindist}(s_i, E)$. 

\[ SL = \{ s (.NN=a), s_1 (.NN=b), e (.NN=b) \} \]
R-TREE ALGORITHM – HEURISTIC 3 (ORDER)

- Entries (satisfying heuristics 1 and 2) are accessed in increasing order of their minimum distances to the query segment $q$.

\[ SL = \{ s \text{(NN)} = a, e \text{(NN)} = a \} \]

Before processing $E_1$

\[ SL = \{ s \text{(NN)} = a, s_1 \text{(NN)} = e, e \text{(NN)} = e \} \]

After processing $E_1$
R-TREE ALGORITHM – LEAF ENTRY

- **Input:** New entry $p$, SL =$\{s_1,...s_{10}\}$
  - 1) retrieve the split points covered by $p$
  - 2) update SL

- **Binary search:** 1) $[s_0,s_{10}] \rightarrow s_5$ 2) $[s_0,s_5] \rightarrow s_2$
  - Using bisector to judge the direction
CKNN – R-TREE ALGORITHM (EXAMPLE)

- Depth First (query segment: se)
OTHER CNN QUERY

• kCNN query (k=2)

• Trajectory NN query (TNN)
  – q1 = [s,u]
  – q2 = [u,v]
  – q3 = [v,e]
  – Each segment has a SL
  – Treated one by one
EXP: PERFORMANCE VS QUERY LENGTH

- Node accesses
- CPU cost (sec)
- Total cost (sec)

Query length: 1%, 5%, 10%, 15%, 20%, 25%

Comparison between CNN and TP.
DISCUSSION AND CONCLUSION

• A fast algorithm for C-$kNN$ query.
• Future work:
  – Rectangle data
  – Moving data points
  – Application to road networks (i.e., travel instead of Euclidean distance)
References


• A presentation by Penny Pan in csci587 Fall’2010
Sample question