Continuous Nearest Neighbor Monitoring in Road Networks
K. Mouratidis1 M.L. Yiu2, D. Papadias3, N. Mamoulis1

Afsin Akdogan
University of Southern California
Computer Science Department

Introduction

Existing methods are designed for Euclidean spaces. Consider a road network (where edge weights correspond to their length, or travel time). Queries and objects move in the network.

Network distance: the length (i.e., sum of weights) of the shortest path connecting them. (Example: taxi – pedestrians)

Sample Query

pedestrian: query and taxis: data objects.
- show me 2 closest taxis"

Related Work

Euclidean NN monitoring: Yu et al. ICDE’05, Xiong et al. ICDE’05, Mouratidis et al. SIGMOD’05
YPK-CNN, SEA-CNN and CPM algorithms
- Search in the cells around query
- Grid index: cannot capture network-imposed constraints
- Circles/rectangles: no mapping to network distance space
- Do not deal with edge updates

Snapshot NN in road networks: e.g., Papadias et al. VLDB’03, Kolahdouzan and Shahabi VLDB’04
- Static data objects, One-time results
**Incremental Monitoring (IMA) and Group Monitoring (GMA) Algorithms**

Two methods (IMA, GMA) for: monitoring NNs according to network distance, with low CPU cost.

Edges: indexed with a quad-tree.

- Store each edge with
  - (i) the objects in it
  - (ii) an influence list

Queries: For each query we store its current NNs, and its expansion tree. (Memory consumption)

**Types of Object Updates**

Only updates affecting the expansion tree can alter the result! (p5 not)

(i) Current NNs moving within distance $q.kNN\_dist$ from $q$ (e.g., p3)
(ii) **Incoming object**: used to lie further than $q.kNN\_dist$ but their new location is closer to $q$ than $q.kNN\_dist$ (e.g., p4)
(iii) **Outgoing object**: current NNs moving further away than $q.kNN\_dist$ from $q$ (e.g., p1)

**IMA: Initial NN computation**

Initial result ($k=3$): expansion tree, infl. intervals, and marks

**IMA: Object updates (Case 1)**

Outgoing no more than incoming NNs:

- At least $k$ objects within distance $q.kNN\_dist$
- Remove outgoing NNs (p1)
- Calculate union of remaining NNs and incoming objects ($(p3',p2) U p4'$)
- Report best $k$ among them

**IMA: Object updates (Case 2)**

More outgoing than incoming:

- Fewer $k$ objects within distance $q.kNN$
- Notice: $q.Tree$ grows according to the new $q.kNN\_dist$!

In brief: update result and shrink expansion tree

An edge $e$ affects $q$, if it contains an interval where the network dist is less than $q.kNN\_dist$

Pare and marks are valid.
IMA: Object updates (Case 2)

New (grown) expansion tree

IMA: Query updates

Re-compute starting from valid tree marks

IMA: Edge updates - Weight increase

There might exist shorter alternatives paths to objects in sub-tree

IMA: Edge updates - Weight decrease

GMA: Main idea

Intersection node: degree above 2 (e.g., n1, n2, n5)
Terminal node: degree 1 (e.g., n8, n9, n4, n3)
Sequence: path between consecutive intersection or terminal nodes
{n1,n2}, {n2,n3}, n3, n4, {n4,n5}, {n1,n6}...

Lemma 1: The \(k\)-NN set of any query in sequence \(s\) is in the union of (i) the objects in \(s\), (ii) the \(k\)-NNs of its intersection nodes (endpoints).

GMA: Main idea (example)

Objects on sequence between \(n_1\) and \(n_2\) = \{\(p_2, p_3\)\}
2-NNs of intersection \(n_1\) = \{\(p_1, p_3\)\}
2-NNs of intersection \(n_2\) = \{\(p_3, p_2\)\}
2-NNs of \(q_1\) or \(q_2\) ∈ \{\(p_2, p_3\)\} \cup \{\(p_1, p_3\)\} \cup \{\(p_3, p_2\)\}
\(n.k\) = the max number of NNs required by any query in \(n.Q\)
GMA: Active nodes

*Active node*: a node n is active if n is the endpoint on any sequence that has at least 1 query (e.g., n1, n5)

GMA monitors the k-NNs of active nodes (using IMA), and uses them to compute the NNs of the actual user queries.

GMA reduces CPU time by
(i) shared execution among queries in the same sequence
(ii) reduction from NN monitoring of moving queries to NN monitoring of static active nodes.

GMA: Initial Result (2NN of qj)

1. First consider n1n5 and add {p5} to qj NN list
2. Among the 2 reached nodes (n1 and n5), n1 is closer so get NNs of n1 {p1, p5}
3. Search continues towards n5, next node on the path is n7
4. Currently qj.kNN_dist = dist(p1, qj) and dist(n7, qj) < qj.kNN_dist
5. Search continues. Consider edge n7n6
6. Terminate at this point with NNs {p1, p5} since the next node n6 has dist(n6, qj) > qj.kNN_dist

Notice that as opposed to IMA, GMA does not store expansion tree for queries.

GMA: Update processing

Initial Result: utilizing active node NNs

NN Maintenance: In every processing cycle do:
1. Update NNs of active nodes with IMA.
2. If NNs of active node n change, re-compute affected queries in sequences adjacent to n
3. If object/edge updates occur in sequence s, re-compute affected queries within sequence s
4. Re-compute moving queries

IMA vs. GMA

GMA outperforms IMA when
• (i) the number of queries is large with respect to the number of query nodes.
  Note: IMA stores an expansion tree for each query
• (ii) When the queries are concentrated in a small part of the network.

Sample experimental results

No previous work.
OVH: re-computes from scratch.

Sample experimental results

Space (KByte) vs. Number of NNs

CPU time (sec) vs. Number of queries

Sample experimental results

OVH vs. IMA vs. GMA

Number of queries vs. CPU time (sec)
Summary
First work about Continuous NN monitoring in road networks.
- No advance information about query/object moving patterns
- Edge weights fluctuate

Two methods:
IMA: processes each query individually. Stores an expansion tree for each q.
GMA: groups queries falling in between 2 intersection.

GMA is faster and requires less space.

Discussion
- IMA Edge update – Increase Weight
  - Inefficient if edges close to root issue update
- IMA Object update which is out of expansion tree
  - No change on expansion tree but still some computation: quad-tree might be traversed to find if updated object is a part of any edge that falls into some expansion tree