Finding the Most Accessible Locations:

Reverse Path Nearest Neighbour Query in Road Networks

ACM SIGSPATIAL GIS 2011

Shuo Shang, Bo Yuan, Ke Deng,
Kexin Xie and Xiaofang Zhou

School of Information Technology & Electrical Engineering
The University of Queensland
Australia

Outline

- **Introduction**
- Problem definition
- Query processing
- Experiment results
- Conclusion

Motivation

- Massive trajectory data
  - GPS-enabled mobile devices
  - Trajectory sharing and recommendation sites
    - Bikely (http://www.bikely.com/)
    - GPS-Waypoints (http://www.gps-waypoints.net/)
    - Share-My-Routes (http://www.sharemyroutes.com/)
    - Microsoft GeoLife (http://research.microsoft.com/enus/projects/geolife/)

Motivation

- What is the problem?
  - Given
    - a set of trajectories *T*
    - a set of location candidates *O*

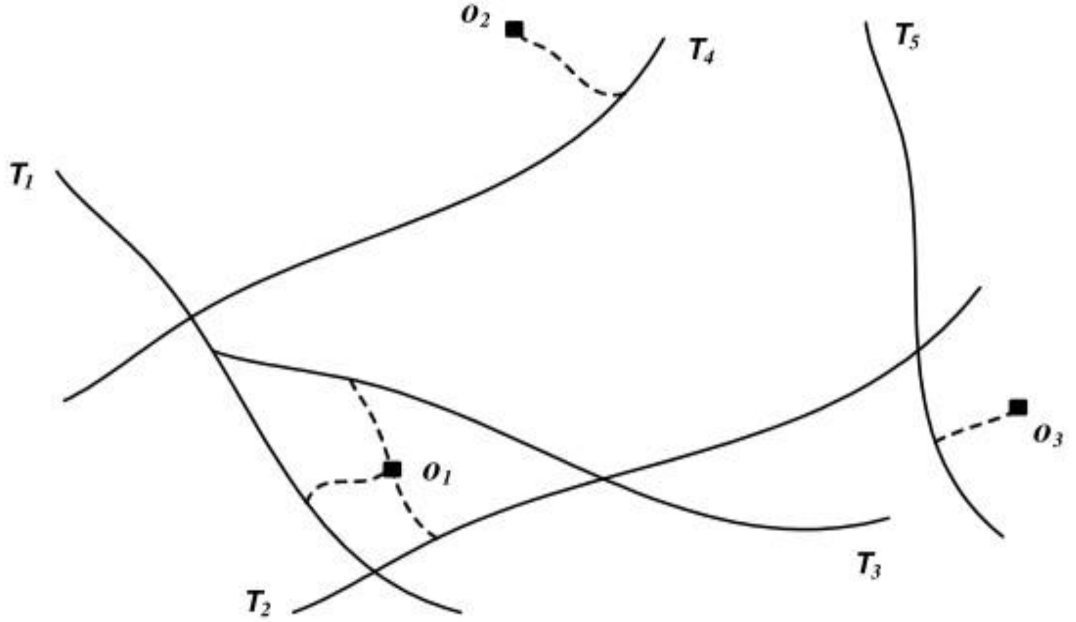
    If a location candidate *o* is the Path Nearest Neighbour (PNN) of *k* trajectories, the *influence-factor* of *o* is defined as *k,* such that *o.if = k*.

    PNN: the closest data object to a specified path, according to a distance metric

  - R-PNN query finds
    - The data object *o* with the maximum *influence-factor,* such that *o.if > o'.if, o' Є O – {o}*

Motivation

- Example



  - t_1, t_2, t_3, t_4 and t_5 are trajectories.
  - o_1, o_2, o_3 are location candidates.
  - $o_1.if = 3$, $o_2.if = 1$, $o_3.if = 1$
  - o_1 is returned.

Motivation

- Applications
  - Facility allocation
    - finding the most accessible location among all given location candidates, to maximize the commercial value of the new facility
  - Traffic monitoring
    - finding the optimal location to monitor the most vehicles
  - Urban planning
  - Location based services

Motivation

- Existing methodologies
  - On computing top-t most influential spatial sites *[VLDB 2005]*

    Finding the most influential sites by considering the relationship in Euclidean space between data points

    - points to points in Euclidean space
    - trajectories to points in road networks

Challenges

- Trajectories to points
- Multiple query points
- Spatial networks

Contributions

- Define a novel type of queries to find the Reverse Path Nearest Neighbour (R-PNN) in road networks.
- Propose an effective trajectory clustering technique that can further enhance the R-PNN query efficiency.
- Devise a two-phase algorithm to answer the R-PNN query efficiently.
- Conduct extensive experiments to demonstrate the efficiency of our approaches

Outline

- Introduction
- **Problem definition**
- Query processing
- Experiment results
- Conclusion

Problem definition

- Road Networks
  - Road network are modelled as connected and undirected planar graphs *G(V,E)*, where *V* is the set of vertices and *E* is the set of edges.

  - A weight can be assigned to each edge to represent its length, travelling time or some other travelling cost factor.

Problem definition

- Trajectories
  - A trajectory of a moving object *t* is a sequence of sample points, $t = \{p_1,p_2,\ldots,p_n\}$, where p_i is the sample point in *G*, for $i = 1,2,\ldots,n$.

  - We assume that all sample points have already been aligned to the vertices on the road network by some map-matching algorithms*. Between two adjacent sample points *a, b,* the moving objects always follow the shortest path connecting *a and b.*

** SODA 2003, VLDB 2005, SSDBM 2006*

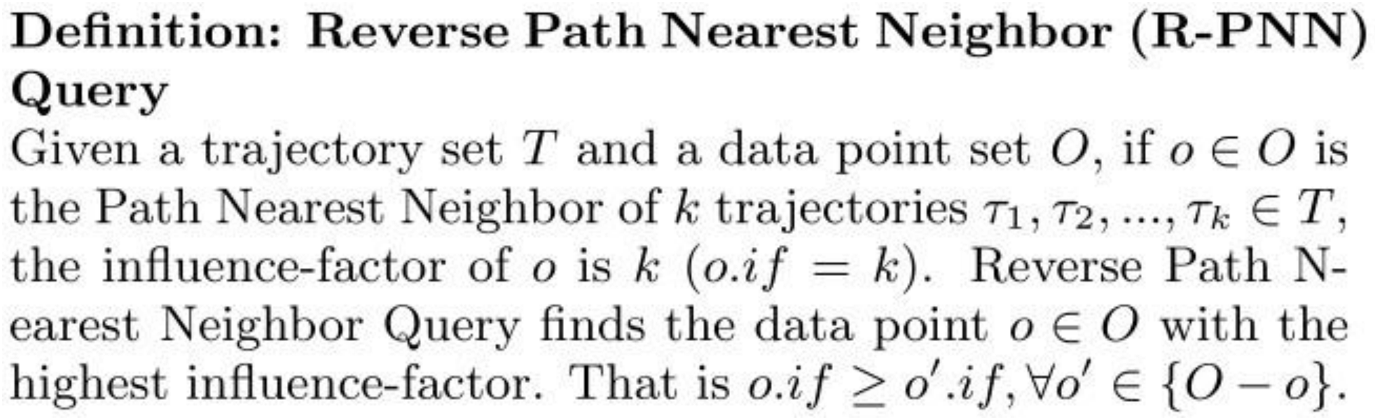
Problem definition

- The distance between a trajectory *t* and a data point *o* is defined as

$$d_M(o,\tau) = \min_{v_i \in \tau}\{sd(o, v_i)\}$$

Definition: Path Nearest Neighbor (PNN)
Given a trajectory τ and a set of data points O, the Path Nearest Neighbor (PNN) of τ is the data point $o \in O$ with the minimum $d_M(o,\tau)$. That is $d_M(o,\tau) \leq d_M(o',\tau), \forall o' \in \{O - o\}$. $\square$

Problem definition



Definition: Reverse Path Nearest Neighbor (R-PNN) Query

Given a trajectory set T and a data point set O, if $o \in O$ is the Path Nearest Neighbor of k trajectories $\tau_1, \tau_2, ..., \tau_k \in T$, the influence-factor of o is k $(o.if = k)$. Reverse Path N-earest Neighbor Query finds the data point $o \in O$ with the highest influence-factor. That is $o.if \geq o'.if, \forall o' \in \{O - o\}$.

Outline

- Introduction
- Problem definition
- **Query processing**
- Experiment results
- Conclusion

Trajectory data pre-processing

- Trajectory clustering
  - *k*-medoids trajectory clustering method

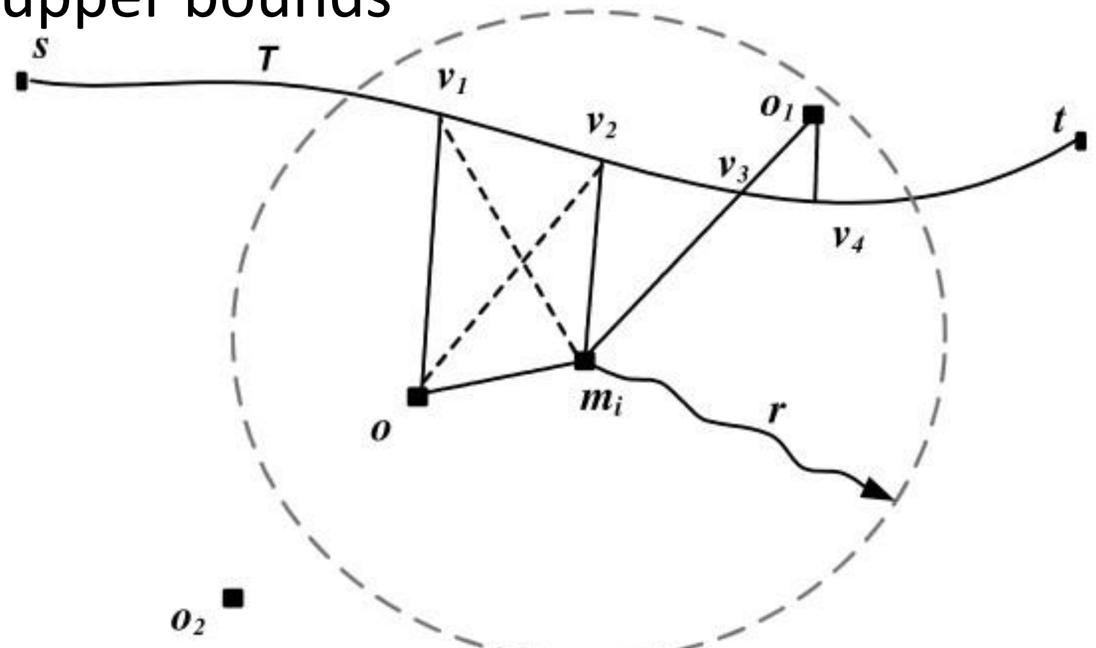
$$R((C_i, m_i) : i \in [1, k]) = \sum_{i=1}^{k} \sum_{\tau \in C_i} d_M(m_i, \tau)$$

m_i is the medoid of cluster C_i for $i \in [1, k]$.

  - Benefits
    - Tighten the searching range during the query processing
    - Allow the use of a divide-and-conquer strategy to further enhance the query efficiency

Identifying candidates

- A pair of lower and upper bounds



  - upper bound
    $$d_M(o,\tau).ub = d_M(m_i,\tau) + sd(o,m_i)$$
  - lower bound
    $$d_M(o,\tau).lb = sd(o,m_i) - d_M(m_i,\tau) - \max\{d(s,v_2),d(v_2,t)\}$$
  - the gap
    $$d_M(o,\tau).ub - d_M(o,\tau).lb = 2d_M(m_i,\tau) + \max\{d(s,v_2),d(v_2,t)\}$$

Identifying candidates

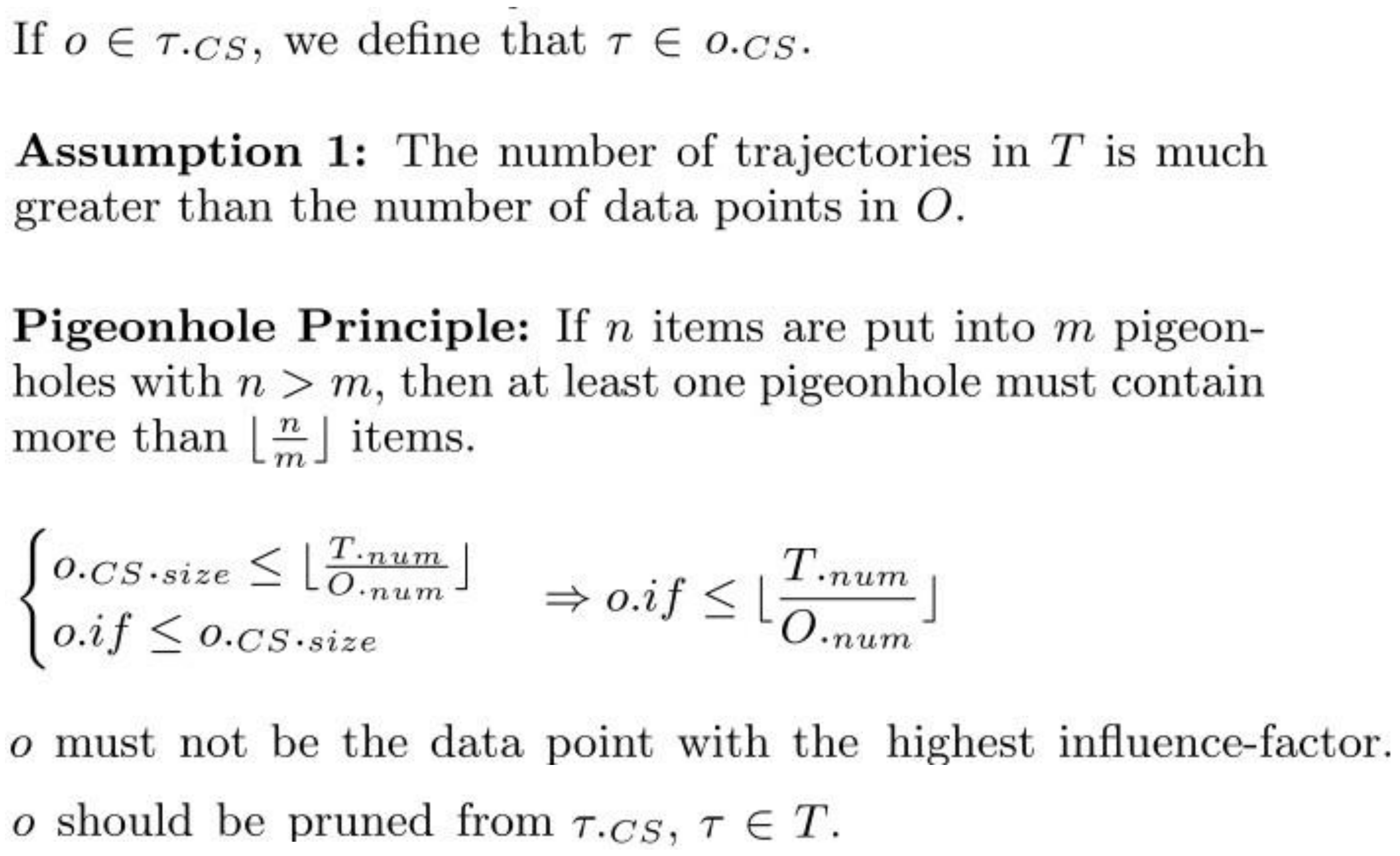
- Data object candidates

$$\tau.ub = \min_{\forall o \in O_s(i)} \{d_M(o, \tau).ub\}$$

$$\tau._{CS} = \{o | sd(o, m_i) - d_M(m_i, \tau) - \max\{d(s, v_2), d(v_2, t)\} \leq \tau.ub\}$$

Identifying candidates

- Filter



If $o \in \tau._{CS}$, we define that $\tau \in o._{CS}$.

Assumption 1: The number of trajectories in T is much greater than the number of data points in O.

Pigeonhole Principle: If n items are put into m pigeonholes with $n > m$, then at least one pigeonhole must contain more than $\lfloor \frac{n}{m} \rfloor$ items.

$$\begin{cases} o._{CS \cdot size} \leq \lfloor \frac{T._{num}}{O._{num}} \rfloor \\ o.if \leq o._{CS \cdot size} \end{cases} \Rightarrow o.if \leq \lfloor \frac{T._{num}}{O._{num}} \rfloor$$

o must not be the data point with the highest influence-factor.

o should be pruned from $\tau._{CS}$, $\tau \in T$.

Identifying candidates

- Candidate sets
  - Trajectory candidate set $T.{cs}$
    - $\tau.{cs}$ is not empty.
  - Data object candidate set $\tau.{cs}$

Searching the most accessible location

- PNN search

Given a trajectory $\tau \in T_{.CS}$ and a data point set $\tau_{.CS}$, the Path Nearest Neighbor query processing takes two steps.

1. Further tighten the data point candidate set $\tau_{.CS}$ according to the proposed lower/upper bound.
2. Compute the minimum network distance between every candidate $o \in \tau_{.CS}$ and τ, then combine the results to find the exact PNN to τ.

- PNN computation results combination

Finally, we combine the PNN query results for trajectories in $T_{.CS}$ to retrieve the data point with the highest influence-factor. It is returned as the most accessible location to users.

Outline

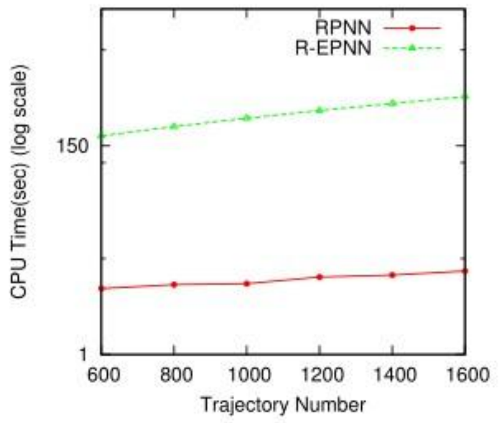
- Introduction
- Problem definition
- Query processing
- **Experiment results**
- Conclusion

Experiment setup

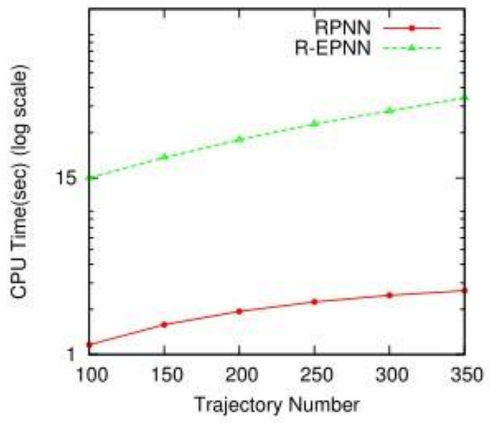
- Road networks
  - Beijing road network (28,342 vertices)
  - Oldenburge road network (6,105 vertices)

- Trajectories
  - In Beijing road network
    - Real trajectory data collected by the MORI project *[VLDB 09]*
  - In Oldenburge road network
    - Synthetic trajectory data

Experiment results



(a) BRN

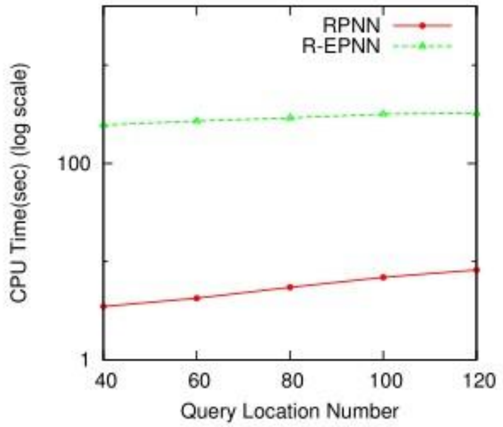


(b) ORN

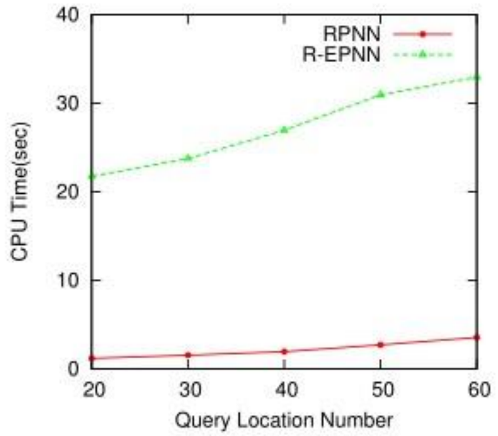
Effect of trajectory number

Location number
BRN: 80
ORN: 40

Trajectory number
BRN: 1000
ORN: 200



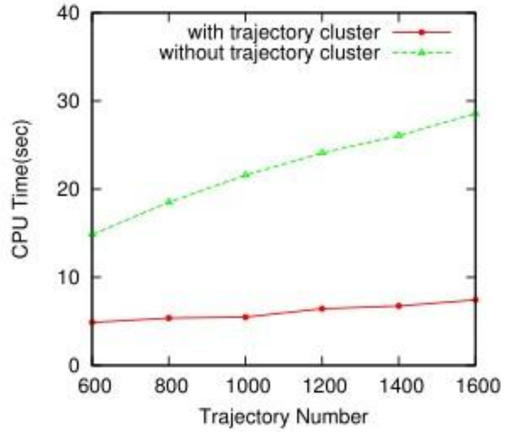
(a) BRN



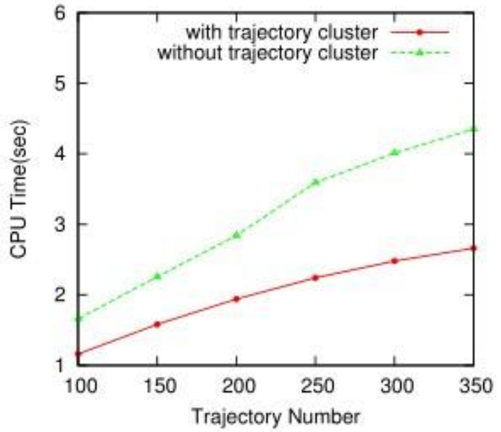
(b) ORN

Effect of query location number

Experiment results

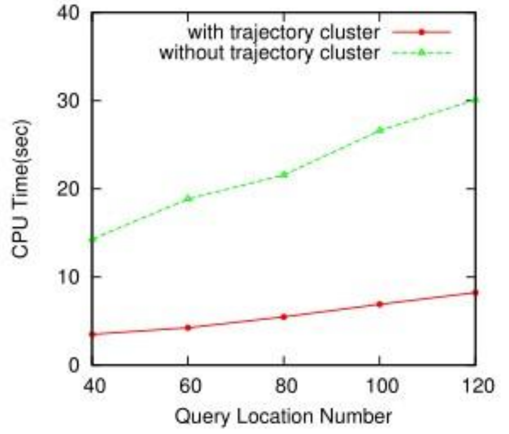


(a) BRN

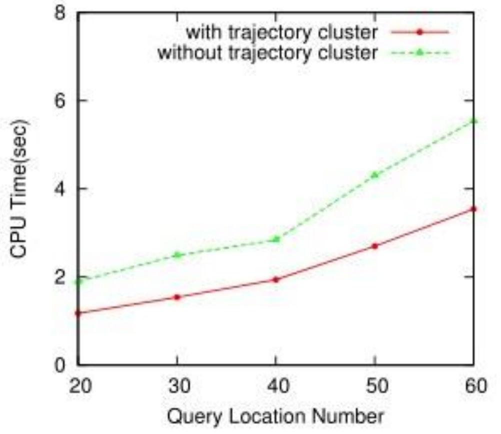


(b) ORN

Location number
BRN: 80
ORN: 40



(c) BRN



(d) ORN

Trajectory number
BRN: 1000
ORN: 200

Effect of trajectory cluster

Outline

- Introduction
- Problem definition
- Query processing
- Experiment results
- **Conclusion**

Conclusion

- Propose and investigate Reverse Path Nearest Neighbour query in road networks
- Devise an effective trajectory clustering approach to enhance the R-PNN query efficiency
- Design a two-phase searching algorithm to address R-PNN query efficiently

Thank you!