CSE6488: Mobile Computing Systems

An Efficient and Scalable Approach to CNN Queries in a Road Network

Sungwon Jung

Dept. of Computer Science and Engineering
Sogang University
Seoul, Korea
Tel: +82-2-705-8930
Email : jungsung@sogang.ac.kr

Introduction

- Example Queries on Figure 1
  - NN Query : Retrieve the 3 closest gas stations from a query point n_1.
  - CNN Query : Find the 2 closest gas stations from all points on the path P from n_1 to n_6 (i.e., $P = \{n_1, n_2, n_3, n_4, n_5, n_6\}$).

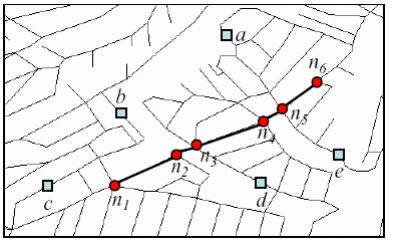


Fig 1. Example continuous search (*a*,*b*,*c*,*d*,*e* : gas stations)

Related Work

- NN Queries in a Road Network
  - INE (Incremental Network Expansion)
    - Based on Dijkstra's algorithm
    - INE suffers from highly computational cost when objects of interest are sparsely distributed.
  - VN^3 (Voronoi-based Network NN Search)
    - VN^3 evaluates *k* NN queries using the Voronoi diagram
    - If the density of objects increases, VN^3 suffers from computational overhead of precalculating network Voronoi polygons

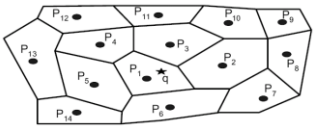
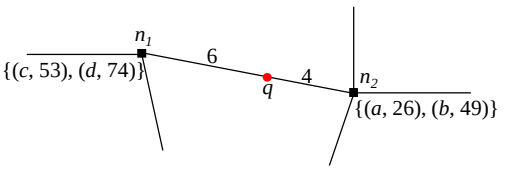


Figure 2: Sample network Voronoi diagram

- Unfortunately, the performance of existing approaches depends largely on the densities of the objects.

Nearest Neighbor Search

- UNICONS (UNIque CONtinuous Search algorithms) for NN queries
  - It incorporates the precomputed NN lists in using Dijkstra's algorithm for NN queries
  - A node where three or more edges meet is called *an intersection point* and an intersection point which maintains precomputed NNs is called *a condensing point*.



Q : Which objects are the two NNs of *q*?
A : The two NNs of *q* are *a* (4+26) and *b* (4+49).

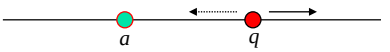
Fig 3. Example NN query processing (n_1,n_2 : condensing points)

Two Basic Ideas of CNN Search

- The 1st basic idea
  - To perform a continuous search along a path $P = \{n_i, n_{i+1}, \ldots, n_j\}$, it is sufficient to retrieve objects on the query path and to run a static query at each node n_k ($i \le k \le j$).

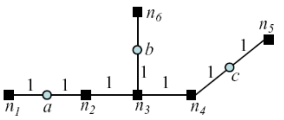
$$\mathcal{R}_{path} = \mathcal{O}_{path} \cup \mathcal{R}_{n_i} \cup \mathcal{R}_{n_{i+1}} \cdots \cup \mathcal{R}_{n_j}$$

- The 2nd basic idea
  - The change in the network distance between a moving query point and a static object of interest can be expressed as *a piecewise linear equation*.

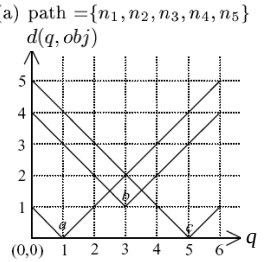


$d(q,a) = |q-a|$

Example of the two Basic Ideas



(a) path $=\{n_1, n_2, n_3, n_4, n_5\}$



$$R_{path} = O_{path} \cup R_{n_1} \cup R_{n_2} \cup R_{n_3} \cup R_{n_4} \cup R_{n_5}$$

(b) $d(q,a) = |q-1|$, $d(q,b) = |q-3|+1$, $d(q,c) = |q-5|$

Fig 4. d(q,a), d(q,b), and d(q,c)

Sketch of CNN Search Algorithm

- Our CNN search algorithm based on the divide and conquer method proceeds with the following three steps:
  - [Step 1] Divide a query path into subpaths based on intersection points
  - [Step 2] Determine valid intervals for each subpath
  - [Step 3] Merge valid intervals of adjacent subpaths

7

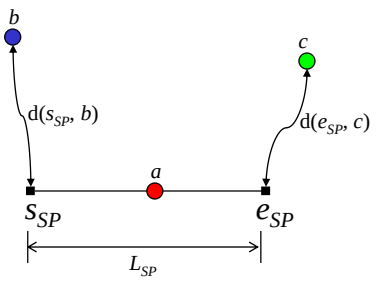
Sketch of CNN Search Algorithm (cont'd)

- [Step 2] consists of the four sub-steps as follows:
  - [Step 2.1] Retrieve objects on the subpath
  - [Step 2.2] Issue two NN queries from the start and end points of the subpath
  - [Step 2.3] Remove some tuples obtained from Steps 2.1 and 2.2 using the *cover* relationship
  - [Step 2.4] Divide the subpath into valid intervals

8

CNN Search Algorithm for the subpath

- The result *R* of Steps 2.1 and 2.2 is the set of (*obj*, *x*, *y*) tuples
  - If *obj* (e.g, *a*) is located on the subpath, $x = d(s_{SP}, obj)$ and $y = 0$.
  - If *obj* (e.g, *b*) is an object satisfying the query predicate at s_{SP}, $x = 0$ and $y = d(s_{SP}, obj)$.
  - If *obj* (e.g, *c*) is an object satisfying the query predicate at e_{SP}, $x = L_{SP}$ and $y = d(e_{SP}, obj)$



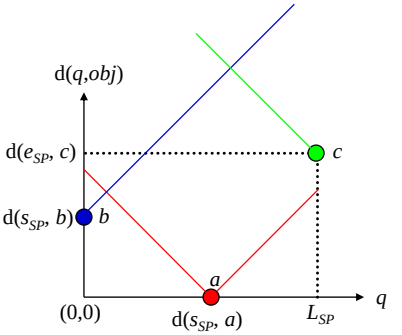


Fig 5. Plotting tuples on the chart
(*q* : location of the query point)

The *Cover* Relationship

- The *cover* relationship
  - It is applied to tuples with the same object

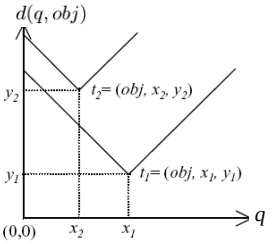


Fig 6. t_1 covers t_2 iff $y_2 \geq |x_2 - x_1| + y_1$

  - A tuple (*obj*, *x*, *y*) on the chart gives the network distance from *q* to *obj* as follows: $d(q, obj) = |q - x| + y$

Example of CNN Query Processing

- For static objects *a* to *e* on the road network of Figure 7, continuously display the two closest objects from any position on the query path $P = \{n_3, n_5, n_7, n_8\}$.

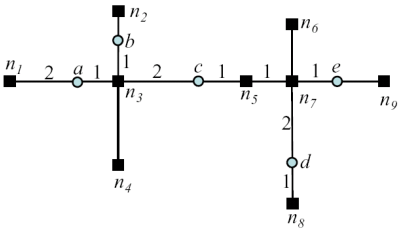


Fig 7. Example Road Network

[Step1] Divide a query path into subpaths : $SP_1 = \{n_3,n_5,n_7\}$ and $SP_2 = \{n_7,n_8\}$

[Step 2] Determine valid intervals for each subpath : Detailed explanation is provided in the next slide

[Step 3] Merge valid intervals of adjacent subpaths

- [Step 2.1] Retrieve the objects on $SP_1 = \{n_3,n_5,n_7\}$
  $O_{SP_1} = \{(c,2,0)\}$
- [Step 2.2] Retrieve the qualifying objects on two end points s_{SP_1} and e_{SP_1}
  $S_{SP_1} = \{(a,0,1),(b,0,1)\}$ and $E_{SP_1} = \{(e,4,1),(c,4,2)\}$
  $\therefore R = O_{SP_1} \cup S_{SP_1} \cup E_{SP_1} = \{(c,2,0),(a,0,1),(b,0,1),(e,4,1),(c,4,2)\}$
- [Step 2.3] Remove redundant tuples using the *cover* relationship
  $R = R - \{(c,4,2)\} \quad \therefore R = \{(c,2,0),(a,0,1),(b,0,1),(e,4,1)\}$

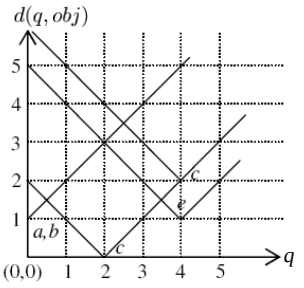


Fig 8. Plotting 5 tuples in *R* on the chart

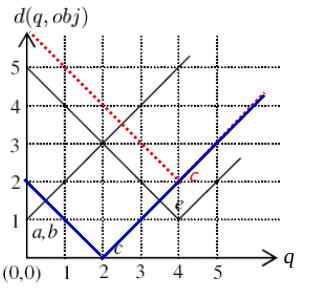
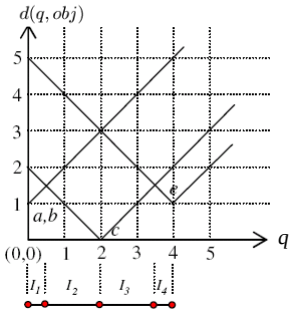


Fig 9. Filtering redundant tuples

Example of CNN Query Processing (cont'd)

- [Step 2.4] Divide the subpath into valid intervals with a set of the same NNs



I	R_I
$I_1=[0, \frac{1}{2}]$	$\{a, b\}$
$I_2=[\frac{1}{2}, 2]$	$\{c, a\}$
$I_3=[2, 3\frac{1}{2}]$	$\{c, e\}$
$I_4=[3\frac{1}{2}, 4]$	$\{c, e\}$

(a) Initial Result

I	R_I
$I_1=[0, \frac{1}{2}]$	$\{a, b\}$
$I_2=[\frac{1}{2}, 2]$	$\{a, c\}$
$I'_3=[2, 4]$	$\{c, e\}$

(b) Final Result

Fig 11. CNN Search result for $SP_1=\{n_3, n_5, n_7\}$

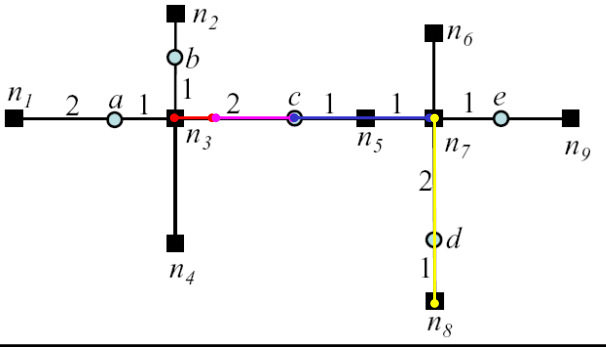
Fig 10. Dividing the query path into valid intervals

Example of CNN Query Processing (cont'd)

- In the same way, if we compute the CNN result for $SP_2=\{n_7, n_8\}$, we can obtain $R_{SP_2}=\{([4,7],\{d,e\})\}$

- [Step 3] Merge R_{SP_1} and R_{SP_2} for SP_1 and SP_2, respectively, in order to obtain the final query result R_P

$$\therefore R_P = R_{SP_1} \cup R_{SP_2} = \{([0,\tfrac{1}{2}],\{a,b\}),([\tfrac{1}{2},2],\{a,c\}),([2,4],\{c,e\}),([4,7],\{d,e\})\}$$



Interval	Results
$[0, \frac{1}{2}]$	{a,b}
$[\frac{1}{2}, 2]$	{a,c}
[2,4]	{c,e}
[4,7]	{d,e}

Fig 12. CNN Search result for $P=\{n_3, n_5, n_7, n_8\}$