CSEG601 & CSE5601
Spatial Data Management & Application :

LBS Query Processing Methods in Road Network DB

Sungwon Jung

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

Road Network Query Processing

- If a user at location ***q*** poses the range query "find the hotels within a 15km range", the result will contain ***a, b*** and ***c*** (the numbers in the figure correspond to network distance).
- Similarly, a nearest neighbor query will return hotel ***b***.
  - the Euclidean nearest neighbor is ***d***, which is actually farthest hotel in the network.

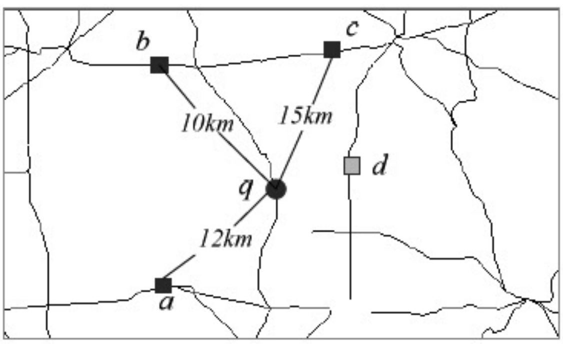
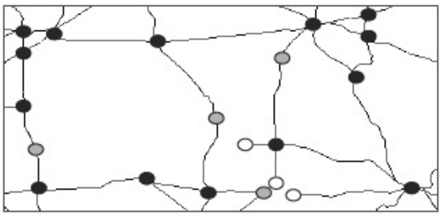


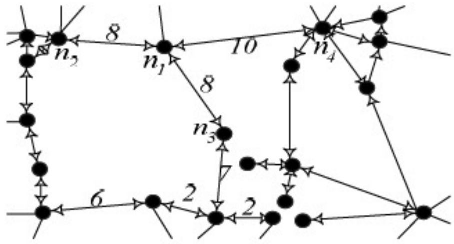
Figure 1.1: Road network query example

Graph Modeling of Road Network

- The graph nodes generated by this process are:
  - the network junctions (e.g., the black points)
  - the starting/ending point of a road segment (white)
  - depending on the application, additional points (gray) such as the ones where the curvature or speed limit changes



(a) A road network



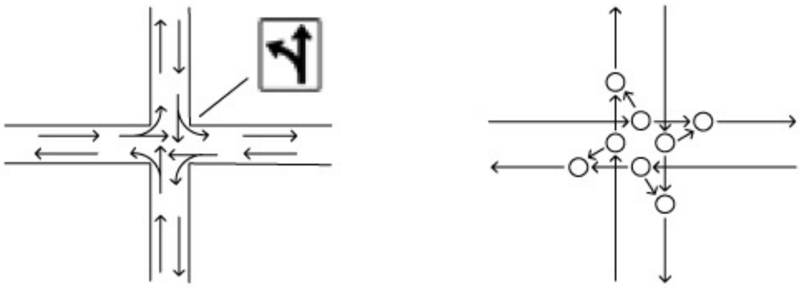
(b) The modeling graph

Graph Modeling of the Road Network

- Each edge connecting nodes n_i, n_j stores the *network distance* $\boldsymbol{d_N(n_i, n_j)}$.
- For nodes that are not directly connected, $\boldsymbol{d_N(n_i, n_j)}$ equals the length of the shortest path from n_i *to* n_j
- If unidirectional traffic is allowed (e.g., one-way road segments), $\boldsymbol{d_N(n_i, n_j)}$ is asymmetric
  - $\boldsymbol{d_N(n_i, n_j) \neq d_N(n_j, n_i)}$.
- *Euclidean lower-bound property*
  - $\boldsymbol{d_E(n_i, n_j) \leq d_N(n_j, n_i)}$

Graph Modeling of the Road Network

- Constraints, such as special traffic controls, can be modeled by including extra nodes to the graph

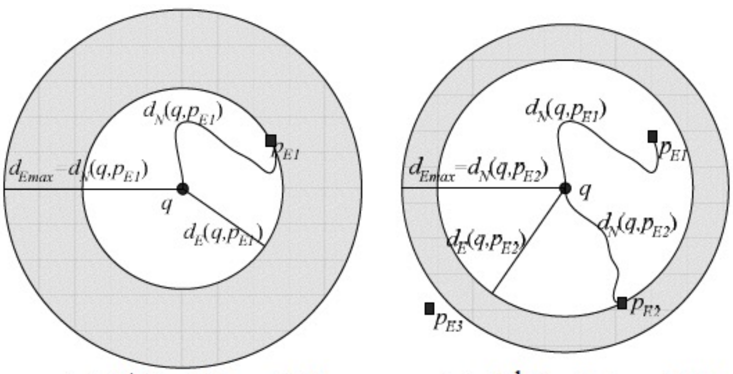


(a) A road junction (b) The modeling graph

Figure 3.2: Example of pragmatic constraint

Nearest Neighbor Queries in Spatial Network DB

- IER(Incremental Euclidean Restrictions) Algorithm

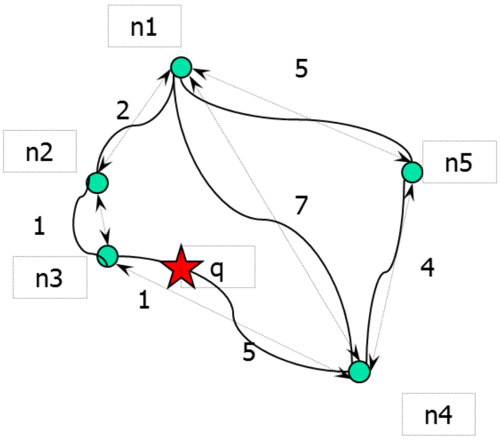


(a) 1st Euclidean NN (a) 2nd Euclidean NN

Figure 4.1: Finding the NN p_{E2}

Nearest Neighbor Queries in Spatial Network DB

- IER(Incremental Euclidean Restrictions) Algorithm
  - Find the first 1 nearest neighbors of location q



- Find the Euclidean nearest neighbor n3
- Compute the network distance:
  d_N(q, n3)= Compute_ND(q, n3)
- Set $d_{Emax} = d_N$(q, n3)
- Repeat the process of retrieving other nodes. To node nk ,
  - if d_N(q, nk) < d_N(q, n3), then set $d_{Emax} = d_N$(q, nk)
  - Otherwise, return the node which has set d_{Emax} and stop

Nearest Neighbor Queries in Spatial Network DB

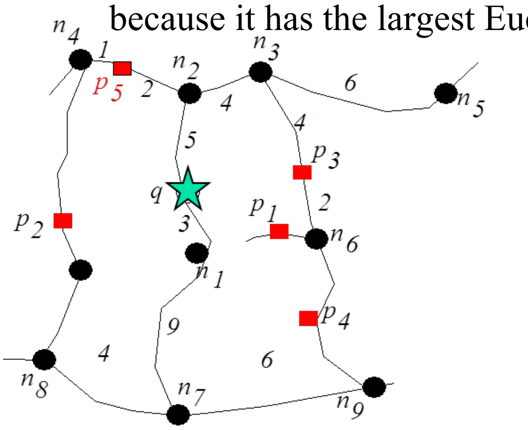
- IER Algorithm:
  - Find the k nearest neighbors of location q

Algorithm IER (q, k)
/* q is the query point */
$\{p_1,\ldots,p_k\}$=Euclidean_NN(q,k);
for each entity p_i
$\quad d_N(q,p_i)$=*compute_ND*(q,p_i)
sort $\{p_1,\ldots,p_k\}$ in ascending order of $d_N(q,p_i)$
$d_{Emax}= d_N(q,p_k)$
repeat
$\quad (p,d_E(q,p))$=next_Euclidean_NN(q);
$\quad$ if $(d_N(q,p)<d_N(q,p_k))$ // p closer than the k^{th} NN
$\quad\quad$ insert p in $\{p_1,\ldots,p_k\}$ // remove ex-k^{th} NN
$\quad\quad d_{Emax} = d_N(q,p_k)$
until $d_E(q,p)>d_{Emax}$
End IER

Figure 4.2: Incremental Euclidean Restriction

Nearest Neighbor Queries in Spatial Network DB

- **INE**(Incremental Network Expansion) Algorithm
  - According to IER algorithm, p5 will be retrieved as the last one, because it has the largest Euclidean distance to ***q***



$d_{Nmax} = \infty$

$Q = <(n_1,3), (n_2,5)>$

de-queue $(n_1,3)$ from Q and (n_1,n_7)

$Q = <(n_2,5),(n_7,12)>$

de-queue $(n_2,5)$ from Q and (n_2,n_4)

$S_{cover} = \{p_5\}$, $1NN = \{p_5\}$, $d_{Nmax} = d_N(q, p_5) = 7$

$Q = <(n_4,8),(n_7,12)>$ and (n_2,n_3)

$Q = <(n_4,8),(n_3,9),(n_7,12)>$

de-queue $(n_4,8)$ from Q, $d_N(q,n_4) = 8 > d_{Nmax} = 7$, and Stop

Algorithm INE (q, k)
1. $n_i n_j$=*find_segment*(q)
2. S_{cover}=*find_entities*$(n_i n_j)$; // S_{cover} is the set of entities covered by $n_i n_j$
3. $\{p_1,\ldots,p_k\}$=the k (network) nearest entities in S_{cover} sorted in ascending order of their network distance ($p_m, p_{m+1}\ldots p_k$ may be $\varnothing$ if S_{cover} contains $< k$ points)
4. $d_{Nmax}=d_N(q,p_k)$ // if $p_k=\varnothing$, $d_{Nmax}=\infty$
5. $Q = <(n_i, d_N(q,n_i)), (n_j, d_N(q,n_j))>$ //sorted on d_N
6. de-queue the node n in Q with the smallest $d_N(q,n)$
7. while $(d_N(q,n)<d_{Nmax})$
8.   for each non-visited adjacent node n_x of n
9. S_{cover}=*find_entities*$(n_x n)$;
10.   update $\{p_1,\ldots,p_k\}$ from $\{p_1,\ldots,p_k\}\cup S_{cover}$
11. $d_{Nmax}=d_N(q,p_k)$
12.   en-queue $(n_x, d_N(q,n_x))$
13.   de-queue the next node n in Q

End INE

Figure 4.4: Incremental Network Expansion

Range Queries in SNDB

- RER(Range Euclidean Restriction) method
  - Given a source point ***q***, a value ***e*** and a spatial dataset ***S***, a range query retrieves all objects of ***S*** that are within network distance ***e*** from ***q***
  - $\boldsymbol{d_N(q,p) \le e \rightarrow d_E(q,p) \le e}$

Algorithm RER(q, e)

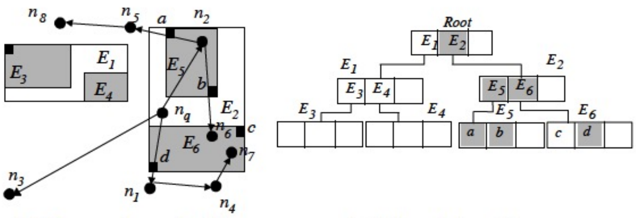
/* *q: query point, e: the network distance threshold* */
1. *result*=$\varnothing$
2. S' = Euclidean-range(q, e)
3. $n_i n_j$=*find_segment*(q)
4. $Q=<(n_i,d_N(q,n_i)), (n_j,d_N(q,n_j))>$
5. de-queue the node n in Q with the smallest $d_N(q,n)$
6. while $(d_N(q,n)\le e$ and $S' \ne \varnothing)$
7.   for each non-visited adjacent node n_x of n
8.   for each point s of S'
9.   if *check_entity*$(n_x n, s)$
10.   *result*=*result*$\cup\{s\}$; $S' = S' - \{s\}$
11.   en-queue $(n_x, d_N(q,n_x))$
12.   de-queue the next node n in Q
13. end while

End RER

Figure 5.1: Range Euclidean Restriction

Range Queries in SNDB

- RNE(Range Network Expansion) method
  - RNE first computes the set ***QS*** of qualifying segments within network range ***e*** from ***q***
  - It then retrieves the data entities falling on these segments



(a) Network and objects (b) The object R-tree

Figure 5.2: Example of RNE

Root Intermediate node:
For E_1, $QS_1 = \emptyset$
For E_2, $QS_2 = \{(n_2n_5),(n_2n_q),(n_2n_6),(n_qn_3),(n_qn_1),(n_4n_7)\}$
E_2 Intermediate node:
For E_5, $QS_5 = \{(n_2n_5),(n_2n_q),(n_2n_6)\}$
For E_6, $QS_6 = \{(n_2n_6),(n_qn_1),(n_4n_7)\}$
E_5 leaf node
$Result_5 = \{a,b\}$, $Result = \{a,b\}$
E_6 leaf node
$Result_6 = \{d\}$, $Result = Result \cup \{d\} = \{a,b,d\}$

Algorithm RNE(*node_id*, *QS*, *result*)
1. if (*node_id* is an intermediate node)
2. compute QS_i for each entry E_i in *node_id* // join
3. for each entry E_i in *node_id*
4. if ($QS_i \neq \varnothing$)
5. RNE(E_i.*node_id*, QS_i, *result*)
6. else // *node* is a leaf node
7. $result_{node\_id}$ =plane-sweep(*node_id.entries*, QS_i)
8. sort $result_{node\_id}$ to remove duplicates
9. $result = result \cup result_{node\_id}$

End RNE

- Alternative Method:
  - The MBR of all segments in ***QS*** is applied as a range query to the object R-tree

Closet-Pairs Euclidean Restriction

Algorithm CPER (*S*,*T*, *k*)
/* *S* and *T* are two entity data sets; *k* is the number of closest pairs to be retrieved*/
1. $\{(s_1,t_1),\ldots,(s_k,t_k)\}$=Euclidean_CP($S$,$T$,$k$);
   // find the k Euclidean closest pairs
2. for i=1 to k
3. $d_N(s_i,t_i)$=*compute_ND*(s_i,t_i)
4. sort (s_i,t_i) in ascending order of their $d_N(s_i,t_i)$
5. $d_{Emax}=d_N(s_k,t_k)$
6. repeat
7. (s',t') = next_Euclidean_CP(S,T)
8. $d_N(s',t')$=*compute_ND*(s',t')
9. if ($d_N(s',t')<d_{Emax}$
   // (s',t') is closer in the network than (s_k,t_k)
10. insert (s',t') in $\{(s_1,t_1),\ldots,(s_k,t_k)\}$
11. $d_{Emax}=d_N(s_k,t_k)$
12. until $d_E(s',t')>d_{Emax}$

End CPER