Scalable Network Distance Browsing in Spatial Databases

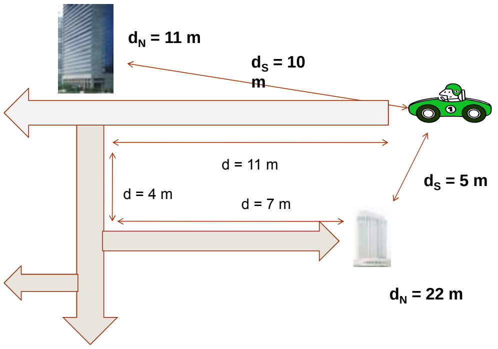
-Hanan Samet, Jagan Sankaranarayanan, Houman Alborzi, SIGMOD '08

By:
Nakul Desai

Outline

- Introduction to Spatial Networks and Network Distances
- Conventional Algorithms for Nearest Neighbor Queries in SNDB
- Shortest-Path Quadtrees
- Morton Blocks
- Distance Encoding
- Best-first k NN algorithm
- Execution and space requirements
- Experimental Results
- Conclusion
- References

Introduction to Road Networks and Network Distances

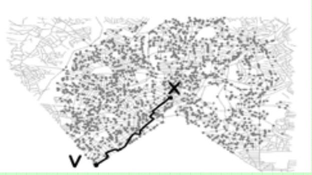


Contd…

- Mapping services such as google maps require a ***real-time response*** to queries such as finding shortest routes between any two locations along a spatial network.

Contd…

- Requirement for a real-time response prevents the use of conventional graph based algorithms like IER and INE that utilize Dijkstra's algorithm in some part of their solution.
- **Problem with Dijkstra's algorithm:** It examines every vertex closer to query point 'q' via the shortest-path from 'q' rather than visiting the vertices associated with the desired objects i.e. ***the algorithm visits many vertices before reaching the one we are interested in***



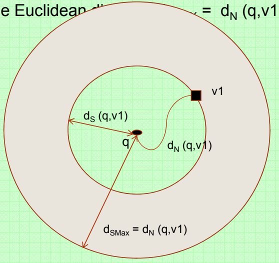
Contd…

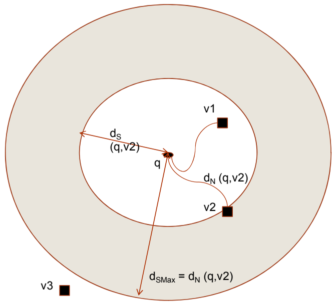
- **GOAL:** To examine only those vertices that are lie on the shortest-path from 'q' to the object.

  i.e. An algorithm that would take O(k) time to find the shortest-path between vertices of a spatial network, where 'k' is the number of vertices that lie on the shortest path.
- The algorithm is based on pre-computing the shortest-path distances between every pair of vertices in the spatial network and storing it along with the path information ***efficiently*** using some form of encoding.
- It uses a best first approach to finding the K Nearest Neighbors to a query point 'q'.

IER (Incremental Euclidean Restriction)

- Based on the fact that d_S (q, v) ≤ d_N (q, v). i.e the ***Euclidean distance*** lower bounds the ***Network Distance***.
- First retrieve the Euclidean NN ‘v1’ to ‘q’ using the R-tree based NN algorithm.
- Compute the Network Distance ‘d_N (q, v1)’ using Dijkstra’s algorithm.
- Due to the Euclidean Lower Bound Property, objects closer to q than v1 must lie within the Euclidean [illegible] = d_N (q,v1) i.e in the shaded region.



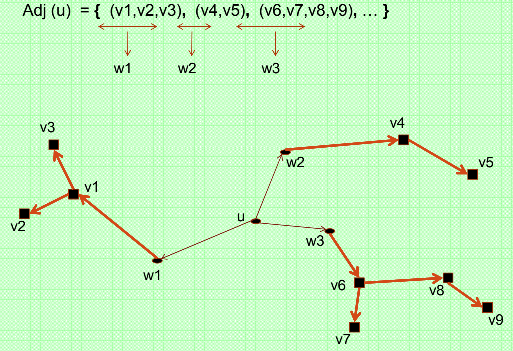


- Since d_N (q,v2) < d_N (q,v1), v2 becomes the current NN and d_{SMAX} is updated accordingly.
- The next Euclidean NN ‘v3’ falls out of the shaded region i.e its d_S (q,v3) > d_N (q,v2), the algorithm terminates with v2 as the NN.
- This can be extended to k NN accordingly by considering d_{SMAX} = d_N (q,vk), where vk is the kth Euclidean NN of q.

Representing shortest path information

- ***Using Adjacency Lists:***

  Adj (u) = **{** (v1,v2,v3)**,** (v4,v5)**,** (v6,v7,v8,v9)**,** … **}**

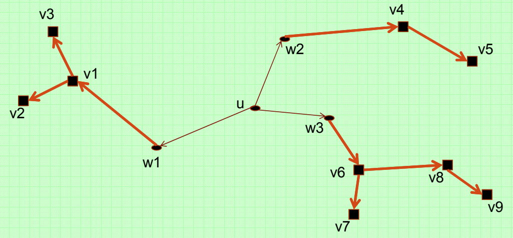


Representing shortest path information

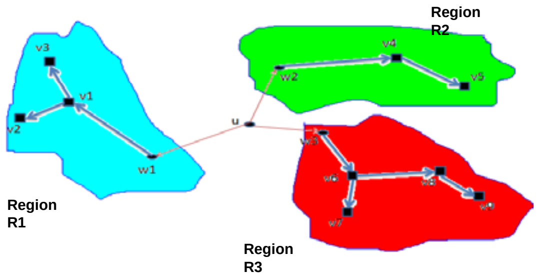
- ***Drawbacks of Adjacency Lists:***

  Absence of index → Searches are sequencial

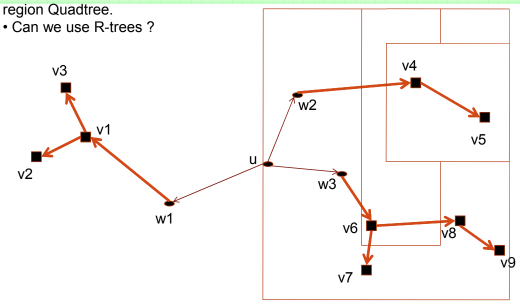
  Space Requirement for each List is O(N).
- ***Solution*** → ***Shortest path Map***



- Each element of the Adjacency List has some ***spatial coherence*** in the sense that they are in close spatial proximity.
- Thus each element can be viewed as a region ‘r_i ‘ corresponding to a vertex ‘w_i ‘ to which ‘u’ is connected by means of an edge ‘e_i ‘.
- We can now replace the adjacency list by a map corresponding to the vertex ‘u’ termed as the shortest-path map.

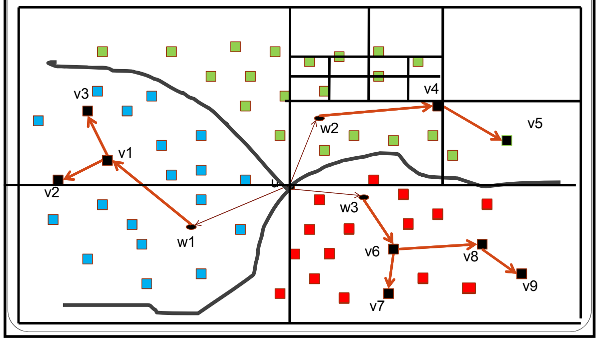


- The advantage of grouping vertices on the basis of the regions in which they lie and identifying each region by the first vertex on the shortest-path into it from vertex ‘u’ is that we can now make use of point location operations to determine the region that contains the destination vertex.
- We can now find the shortest path to a group of vertices.
- The regions can now be index based on a spatial index structure such as the region Quadtree.
- Can we use R-trees ?



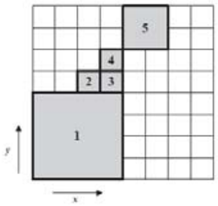
Shortest Path Quadtree

1) Color-coding the map
2) Store the regions in a ***region Quadtree***
3) Represent the regions by a ***Morton Block***

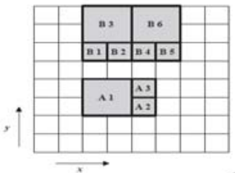


Morton Blocks

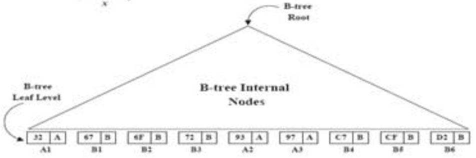
- A Morton Block is an Integer representing a Quadtree block.
- It is based on the Morton Order or the Z-Order, which is a space filling curve that visits all the points in 2-D space exactly once in a predetermined order.
- A mapping from 2-D space to a 1-D space of Integers.



Quadtree Block	z-Value	Level in Quadtree	Morton Block
1	00	1	000000 01
2	011000	3	011000 11
3	011010	3	011010 11
4	011011	3	011011 11
5	1101	2	110100 10



		Morton Block	
Rect	Blk.	Binary	Hex
A	A 1	001100 10	32
	A 2	100100 11	93
	A 3	100101 11	97
B	B 1	011001 11	67
	B 2	011011 11	6F
	B 3	011100 10	72
	B 4	110001 11	C7
	B 5	110011 11	CF
	B 6	110100 10	D2

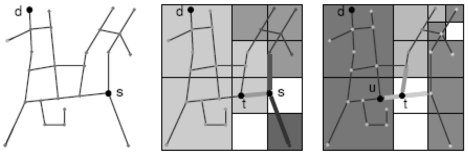


- A link and a distance interval are associated with each morton block.
- The procedure to form morton blocks is as below:

```
 Procedure Mortonize[u, T]
Input: u ϵ V , T is a Region Quadtree on V
Output: MortonList: list of Morton blocks with associated links and distance intervals
1. MortonList ← empty
2. for each leaf-block 'b' ϵ T visited in Morton-order do
3.     if all points v in b are of same color then
4.         append b to MortonList
5.     else
6.         recursively split b until S, the resultant set of blocks, is single colored
7. merge S with MortonList
8. while Morton blocks can be merged do
9.     merge sibling blocks if of the same color
10. for each Morton block b ϵ MortonList do
11.λ⁻ = Minimum ratio of the network distance ($d_N$ (u,v)) to the spatial distance ($d_S$ (u,v)) from u to all the destination vertices in morton block b
12.λ⁺ = Maximum ratio of the network distance ($d_N$ (u,v)) to the spatial distance ($d_S$ (u,v)) from u to all the destination vertices in morton block b
13.Associate (λ⁻ , λ⁺ ) with b
14.Return Morton list.
```

Retrieving the Shortest Path

- Given a source vertex 's' , a destination vertex 'd', the next link 't' in the shortest path between s and d is obtained by performing a simple binary search for a morton block containing d from the morton list.
- Since each morton block for a vertex s is associated with a link, t ← b.link
- Now t is the next link after s in the shortest path between s and v.
- The above algorithm is repeated until v is obtained.
- dist = dist + d_N (s,t)
- The shortest path between s and d requires exactly k steps, where, k = |No of vertices in the shortest path between s and d |

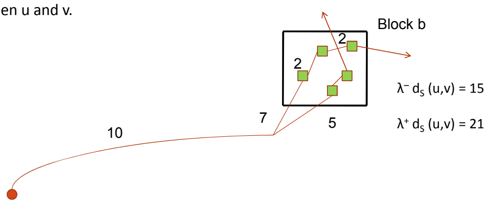


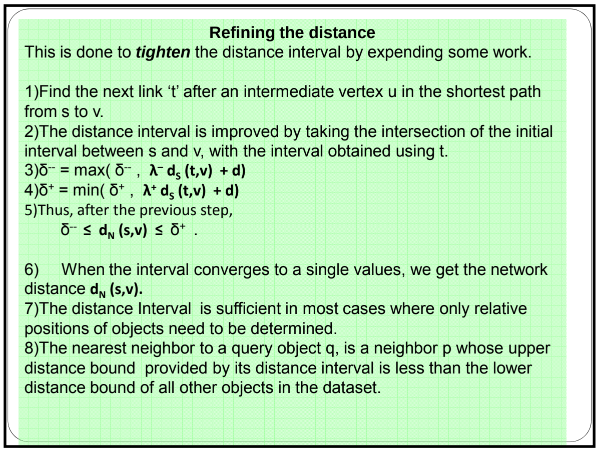
Distance Encoding

Most spatial applications require an approximate estimate of the distance between two vertices u and v on a spatial network.

$$\lambda^- d_S (u,v) \le d_N (u,v) \le \lambda^+ d_S (u,v)$$

Since λ^- and λ^+ are associated with a morton block 'b' , given vertices u and v an initial Interval d_N (u,v) is made available for the shortest path distance between u and v.





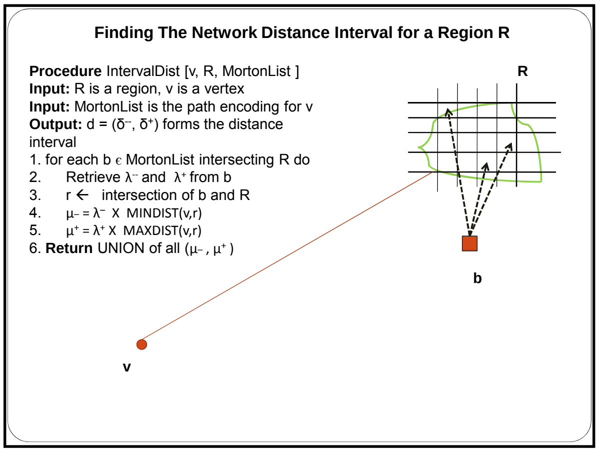
Refining the distance

This is done to ***tighten*** the distance interval by expending some work.

1)Find the next link ‘t’ after an intermediate vertex u in the shortest path from s to v.
2)The distance interval is improved by taking the intersection of the initial interval between s and v, with the interval obtained using t.
3)$\delta^{-} = \max(\delta^{-}, \boldsymbol{\lambda^{-} d_S (t,v) + d})$
4)$\delta^{+} = \min(\delta^{+}, \boldsymbol{\lambda^{+} d_S (t,v) + d})$
5)Thus, after the previous step,

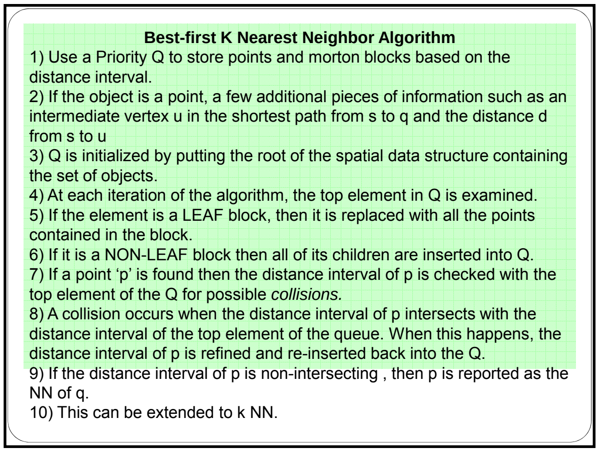
$$\delta^{-} \le \boldsymbol{d_N (s,v)} \le \delta^{+} .$$

6) When the interval converges to a single values, we get the network distance $\boldsymbol{d_N (s,v)}$.
7)The distance Interval is sufficient in most cases where only relative positions of objects need to be determined.
8)The nearest neighbor to a query object q, is a neighbor p whose upper distance bound provided by its distance interval is less than the lower distance bound of all other objects in the dataset.



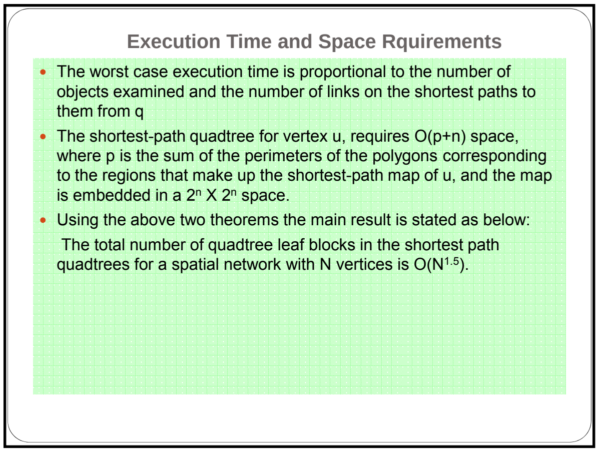
Finding The Network Distance Interval for a Region R

```
Procedure IntervalDist [v, R, MortonList ]
Input: R is a region, v is a vertex
Input: MortonList is the path encoding for v
Output: d = (δ⁻⁻, δ⁺) forms the distance interval
1. for each b ϵ MortonList intersecting R do
2.      Retrieve λ⁻⁻ and λ⁺ from b
3.      r ← intersection of b and R
4.      μ₋ = λ⁻ X MINDIST(v,r)
5.      μ⁺ = λ⁺ X MAXDIST(v,r)
6. Return UNION of all (μ₋ , μ⁺ )
```



Best-first K Nearest Neighbor Algorithm

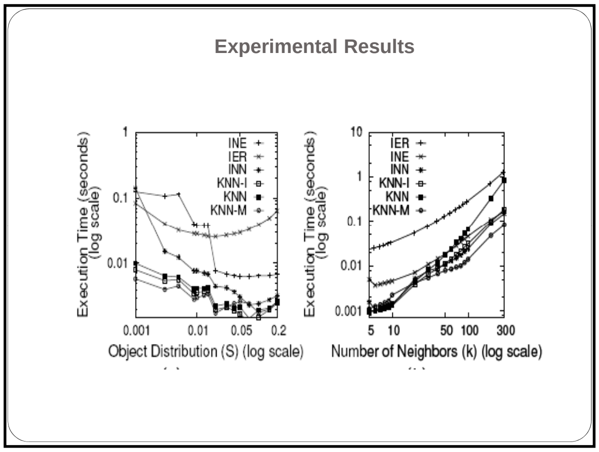
1) Use a Priority Q to store points and morton blocks based on the distance interval.
2) If the object is a point, a few additional pieces of information such as an intermediate vertex u in the shortest path from s to q and the distance d from s to u
3) Q is initialized by putting the root of the spatial data structure containing the set of objects.
4) At each iteration of the algorithm, the top element in Q is examined.
5) If the element is a LEAF block, then it is replaced with all the points contained in the block.
6) If it is a NON-LEAF block then all of its children are inserted into Q.
7) If a point ‘p’ is found then the distance interval of p is checked with the top element of the Q for possible *collisions.*
8) A collision occurs when the distance interval of p intersects with the distance interval of the top element of the queue. When this happens, the distance interval of p is refined and re-inserted back into the Q.
9) If the distance interval of p is non-intersecting , then p is reported as the NN of q.
10) This can be extended to k NN.



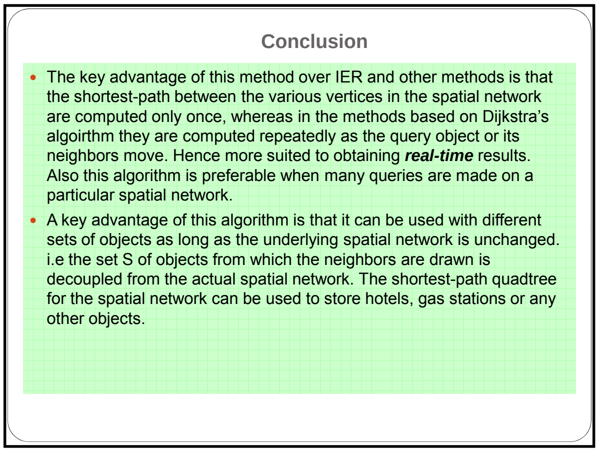
Execution Time and Space Rquirements

- The worst case execution time is proportional to the number of objects examined and the number of links on the shortest paths to them from q
- The shortest-path quadtree for vertex u, requires O(p+n) space, where p is the sum of the perimeters of the polygons corresponding to the regions that make up the shortest-path map of u, and the map is embedded in a 2^n X 2^n space.
- Using the above two theorems the main result is stated as below:

  The total number of quadtree leaf blocks in the shortest path quadtrees for a spatial network with N vertices is $O(N^{1.5})$.

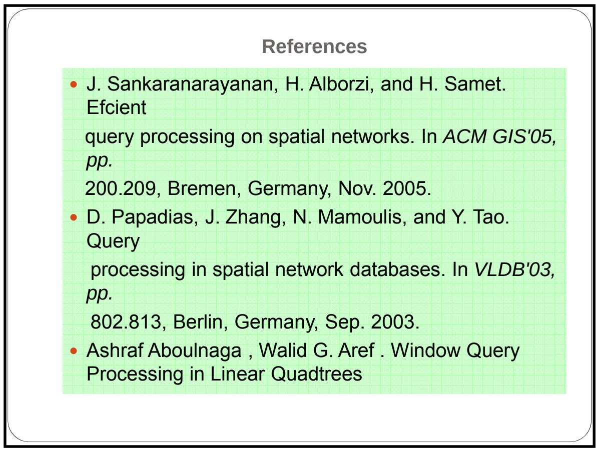


Experimental Results



Conclusion

- The key advantage of this method over IER and other methods is that the shortest-path between the various vertices in the spatial network are computed only once, whereas in the methods based on Dijkstra’s algorithm they are computed repeatedly as the query object or its neighbors move. Hence more suited to obtaining ***real-time*** results. Also this algorithm is preferable when many queries are made on a particular spatial network.
- A key advantage of this algorithm is that it can be used with different sets of objects as long as the underlying spatial network is unchanged. i.e the set S of objects from which the neighbors are drawn is decoupled from the actual spatial network. The shortest-path quadtree for the spatial network can be used to store hotels, gas stations or any other objects.



References

- J. Sankaranarayanan, H. Alborzi, and H. Samet. Efcient
  query processing on spatial networks. In *ACM GIS'05, pp.*
  200.209, Bremen, Germany, Nov. 2005.
- D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query
  processing in spatial network databases. In *VLDB'03, pp.*
  802.813, Berlin, Germany, Sep. 2003.
- Ashraf Aboulnaga , Walid G. Aref . Window Query Processing in Linear Quadtrees