**Title:**

**Continuous K-Nearest Neighbor Query over Moving Objects in Road Networks**

**What is the research problem?**

In all location based services, the primary idea which all papers talk about is cloaking/blurring the location of the user in the spatial region. This paper talk about the model/technique/algorithms used to continuously monitor kth nearest neighbor which helps in implementing the cloaking/blurring of user location.

**Overview/main points of the proposed approach/architecture**

* In most of the other cases the Euclidian distance is calculated, however this papers considers the actual road networks and considers the shortest path connecting the two entities in question.
* The paper proposes two approaches for a dynamic scenario where the objects are moving frequently.
* Broadly this paper recommends two approaches
  + The first approach focuses on the queries which can invalidate the current NN set and accordingly maintains results.
  + The second approach follows the shared execution to reduce the processing time.
* Earlier static evaluation was considered where snapshot was taken queries were answered based on that, however is ever increasing number of mobile user and wide range of location-aware devices now the focus has changed to continuous monitoring, hence continuous k-NN (CkNN) paradigm.
* One of the static approach is Snapshot NN methods
* The paper initially talk about a server which continuously monitors the position. The server doesn’t have knowledge about the velocity of the moving object.
* Continuous monitoring in Euclidian space currently has three algorithms YPK-CNN, SEA-CNN and CPM.
* In case of road network scenario the Euclidian distance will not be helpful as it is based on the road network. Also delay in the response may completely invalidate the rendered response.
* One of the approach suggested is Incremental Monitoring Algorithm (IMA).
* IMA expands the network around until k-NN are found. The idea is compared to an expansion tree. The queries are divided into two part, 1st which can change the existing expansion tree structure. Even if there is an update on the existing expansion tree structure, it tries to calculate the part of the tree which is still valid and re-uses it, in short in incrementally updates the tree using the delta updates. Queries which does not update the tree are simple ignored during this step.
* The papers given a pseudo Incremental Monitoring algorithm, which includes processing Object updates. Updates which make shrink/grow the expansion-tree structure while processing updates.
* Validating the expansion tree or a part of it after the update.
* GMA (Group Monitoring Algorithm) relies on the idea that K-NNs of any query in the path of two nodes belongs to the union of the data objects falling in that path and the K-NN set of its endpoints. Hence GMA monitor the k-NNs of the intersection and uses those to compute the result. Thus GMA has a benefit of shared execution among the query path and reduction of the problem from monitoring moving object to static network nodes
* In term of data structure used GMA additionally maintains a sequence table, however IMA stores the expansion tree of all the queries.
* GMA perform better than IMA when the number of queries are large compared to the number of nodes, also when the queries are concentrated on the small part of the network.

**Experimental evaluation:**

* IMA and GMA were evaluated using sub-network of city like San Francisco with 10K edge and N Object and Q queries and different types of distribution, Object speed and Query speed and number to k-NN’s
* Both the methods were observed to scale well compared to the traditional ways.

**Conclusions:**

* One of the first papers to be addressing continuous K-NN incremental monitoring on the road network.
* IMA processes each query and updates that might alter the expansion tree
* GMA in turn groups together the queries falling in the path between two consecutive intersections in the network
* GMA is generally better than IMA with respect to the space and CPU requirements