K-CNN Search along Route on Road Network

Jun Feng† Naoto Mukai‡ Toyohide Watanabe‡

Department of Information Engineering, Graduate School of Engineering, Nagoya University†

Department of Systems and Social Informatics, Graduate School of Information Science, Nagoya University‡

1. Introduction

The problem of *k* continuous nearest neighbors (*k*-CNN) search along a specific route on road network is to find out *k* nearest neighbor objects for any place on the route. It is an instance of spatial distance semi-join problem. The spatial distance join associates one or more sets of spatial object by distances between objects. The distance semi-join is a useful special case of the distance joins. It finds the nearest object in one spatial dataset for each object in another spatial dataset [1]. A distance is usually defined in terms of spatial attributes, but it may also be defined in many different ways according to various application-specific requirements. In GIS and ITS applications, for example, other metrics such as the shortest path can be used to measure a distance between two places on a road network.

In this paper, we propose new strategies for efficiently processing *k*-CNN search on road network based on our previous CNN search method [2]. The main contributions of the proposed solutions are: 1) we provide an approach of estimating the cutoff distance for *k*-CNN search on road network. This estimated distance allows the algorithms of *k*-CNN search for any point on the predefined route to avoid a *slow start* problem, which may cause a substantial delay in the query processing; and 2) adaptive algorithms are proposed to process *k*-CNN search in a way that *k*-CNN's are returned with an incremental precision. This algorithm adopts a fixed length queue for recording cutoff distance and the first *k* candidates in the search process.

2. Propositions

To solve the *k*-CNN problem, there are two main issues: one is the selection of computation point on the route; and another is the computation of *k*-NN for the computation point. Here, we give two propositions on the road network for nearest object search.

[Proposition 1] For a source point *c* and a target object *t*, when the length of a path from *c* to *t* is *r*, any target object nearer to *c* than *t* can only be found inside a circle region, denoted as *r-region*, whose center is *c* and whose radius is *r*. □

In Figure 1, the r-region generated for *c* is the circle r-region$_1$. The center is *c* and radius r$_1$ is the path length from *c* to *t*.

[Proposition 2] For two points *c* and *t'* on the road network with straight-line distance *d*, the test of whether there is a path shorter than *r* from *c* to *t'* can be based on a path search region, denoted as *p-region*, the sum of the straight-line distance between any nodes inside this region and *c*, and that between this node and *t'* is not longer than *r*. □

In Figure 1, the p-region generated for *c* and t' is the ellipse p-region$_1$.

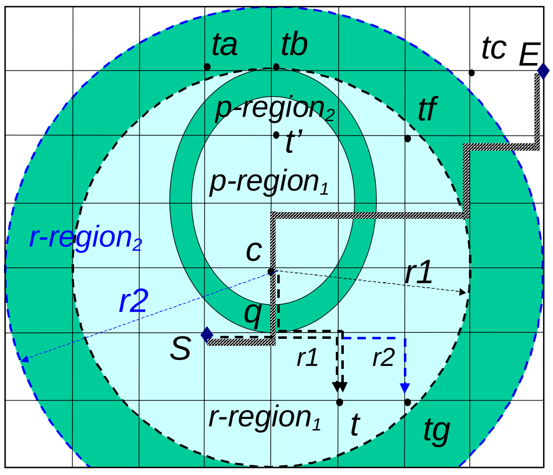

Figure 1. Search regions generated for a new computation point based on the first 2 NN's of the previous computation point.

3. CNN Search Method

We have solved CNN search based on the following ideas: 1) *Locating the target objects inside r-region on the road network*: With the assumption that the road network is too large to be processed in the main memory at once, the locating of target objects on the road network results in an expensive process of merging two indexes of road network and target objects. However, based on **Proposition 1**, if a straight-line restriction between the computation point and a NN candidate can be decided, the locating operation can only be done inside *r-region* for the computation points. 2) *Searching shortest path from the computation point to NN candidate inside p-region:* Based on **Proposition 2**, if a path length restriction from the computation point to a NN candidate can be decided, the path search can be done inside *p-region*.

In Figure 1, the predefined route is *Route (S, E)*, and the target object set is *T*. The first computation point is the start point S of *Route (S, E):* NN for S is t and the divergence point is q. The second computation point is decided as c. The path length from c to t is r1, where

$$r_1 = Path_{cq} + Path_{qt}$$

It means that t is a NN candidate for c with path length r_1 the value of $Path_{qt}$ has been computed in the previous NN search step for S, and the value of $Path_{cq}$ is the curve length between c and q. Based on **Proposition 1**, if *r-region* is decided for c with the radius r, the target objects nearer than t can be found only inside this *r-region*. For NN search of c, only the target objects inside *r-region*$_1$ are regarded as candidates, and the locating of candidates on the road network can be done inside r-region$_1$. Based on **Proposition 2**, to test whether the path length from c to a candidate t' inside *r-region* is shorter than r can be based on a *p-region*: the path length from the current computation point c to NN of the previous computation point is r_1, and the straight-line distance between c and the candidate node t' is d'. p-region$_1$ for the computation of the shortest path from c to t' is the white ellipse in Figure 1.

4. *k*-CNN Search Method

In the process of searching *k*-NN's for a computation point, a priority queue is used to record the sequence of target objects in order of distance from the computation point. When the target objects are indexed by a spatial index structure, i.e., R-tree [3], the distances are straight-line distances between the computation point and the nodes of R-tree or path length from the computation point to the candidate target objects. When an object with the computed path length turns out on the head of the queue, the first (or the nearest) neighbor is found. On the next time, the second (2-nearest) neighbor will be returned. However, in the process of CNN search, the priority queues for the computation points except the start point on the route only record the nodes or objects with the shorter path than r. In other words, all the objects found on the head of queue are in order, but it cannot say that the number of objects in the queue is enough for *k*-CNN search. To solve this problem, another data structure, called *k*-queue, is adopted to keep the up-to-now k candidates and distances for *k*-NN search. The greatest distance is regarded as a cutoff value for pruning nodes or objects: a node or an object with a longer path is not inserted into the priority queue, while there are at least k candidates kept in the priority queue. *k*-queue is defined as a fixed length queue, where

● k is the number of nearest neighbors found for every point on the predefined route. It is decided at the beginning of *k*-CNN search, and kept in the process of the search;

● the elements inside the queue are pairs of $<t$, *path-length*$_t>$. t is a target object and *path-length*$_t$ is the path length from the current computation point to t. There are at most k elements in the queue; and

● the path length of the tail element in the queue is regarded as a cutoff value, which is used to set *r-region* and *p-region*.

For the computation points on the route except the start point, the contents of *k*-queue are initialized as: *k*-NN's of the previous computation point in order. The distance for every object recorded in the queue is the sum of path length from the current computation point to the previous divergence and the path length from the divergence to that object. The *k*-queue for c in Figure 1 is like $\{<t, 3>, <t_g, 4>, \ldots\}$.

Observe Figure 1: the nearest neighbor of c can only be found inside r-region$_1$. However, if t is also the nearest neighbor for c, the second one may found inside r-region$_1$ when there is an object with a longer path length than that of t but shorter than that of t_g from c. This means that for the candidates inside r-region$_1$, if the path search is based on the *p-region*$_1$, it may lead to result missing. This is because p-region$_1$ assures that the path search only succeeds when the path length of the candidate is shorter than r_1.

The distance between internal nodes of R-tree is shorter than that between two objects inside the two nodes, and the distance between the two objects is shorter than the path length between the two objects on road network. The elements in *k*-queue are on the order of path length, the path length determines *r-region* and *p-region*, and the elements in the priority queue are sorted on the three kinds of length. The loose condition for the test of tree nodes becomes stricter and stricter for test of distance between objects and path length from one object to another.

5. Conclusion

From a viewpoint of decreasing the times of disk access, we proposed a method for *k*-CNN search on the large hierarchical road network by narrowing down the search region. However, searches based on the travel cost that may not be in direct proportion to their path length cannot be solved by the method proposed in this paper.

6. Reference

[1] G. R. Hjaltson and H. Samet: "Incremental Distance Join Algorithms for Spatial Databases", *Proc. of ACM-SIGMOD*, pp. 237-248 (1998).

[2] J. Feng and T. Watanabe: "Search of Continuous Nearest Target Objects along Route on Large Hierarchical Road Network", *Proc. of Data Engineering Workshop* (2003).

[3] A. Guttman: "R-Trees: A Dynamic Index Structure for Spatial Searching", *Proc. of ACM SIGMOD'84*, pp. 47-57 (1984).