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

Yuan-Ko Huang[1], Zhi-Wei Chen[1], and Chiang Lee[2]

Department of Computer Science and Information Engineering, National Cheng-Kung University, Tainan, Taiwan, R.O.C.
[1]{hyk,howard1118}@dblab.csie.ncku.edu.tw,
[2]leec@mail.ncku.edu.tw

Abstract. Continuous *K*-Nearest Neighbor (C*K*NN) query is an important type of spatio-temporal queries. A C*K*NN query is to find among all moving objects the *K*-nearest neighbors (*K*NNs) of a moving query object at each timestamp. In this paper, we focus on processing such a C*K*NN query in road networks, where the criterion for determining the *K*NNs is the shortest network distance between objects. We first highlight the limitations of the existing approaches, and then propose a cost-effective algorithm, namely the *Continuous KNN algorithm*, to overcome these limitations. Comprehensive experiments are conducted to demonstrate the efficiency of the proposed approach.

1 Introduction

With the advent of mobile and ubiquitous computing, processing continuous spatio-temporal queries over moving objects has become a necessity for many applications, such as traffic control systems, geographical information systems, and location-aware advertisement. One type of spatio-temporal queries is the *continuous K-nearest neighbor query* (or C*K*NN query for short) that *finds among all moving objects the K-nearest neighbors* (*KNNs*) *of a moving query object at each timestamp*. An example of a C*K*NN query is "finding two nearest cabs of a pedestrian based on his/her current moving speed and direction."

Early methods [1,2,3] proposed to efficiently process such a C*K*NN query focus exclusively on Euclidean spaces, which are inapplicable to road networks. Recently, several studies [4,5,6,7] have investigated how to search the *K*NNs in road networks. These techniques are designed to deal with C*K*NN queries over static objects. Mouratidis et al. [8] first address the issue of continuous monitoring *K*NNs on moving objects. They choose to re-evaluate snapshot *K*NN query at those time instants at which updates occur. However, due to the nature of discrete location updates, the *K*NNs within two consecutive updates are unknown.

In this paper, our efforts on processing such a C*K*NN query are devoted to overcoming the limitations of the previous works mentioned above. That is, we investigate the C*K*NN problem under the following three conditions: (1) All objects (including the query object) move continuously in a road network. (2)

Q. Li et al. (Eds.): APWeb/WAIM 2009, LNCS 5446, pp. 27–38, 2009.

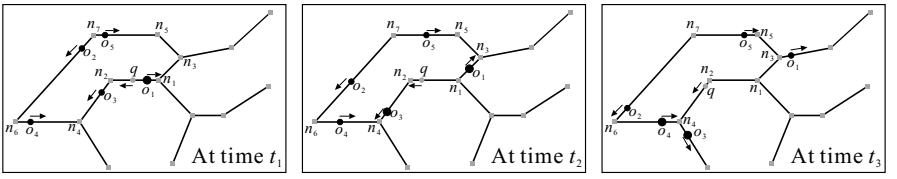

Fig. 1. Example of a C*K*NN query

The distance between two objects is defined as the distance along the shortest path between them in the network. (3) The *K*NNs set of the query object at each timestamp should be completely determined. Let us consider an example in Figure 1, where a set of objects o_1 to o_5 and a query object q move in a road network which is represented as a graph consisting of nodes and edges. In this example, each object moves steadily towards the direction indicated by the corresponding arrow. Assume that the moving query object q wants to know within a certain period of time the nearest neighbor on its route (i.e., a C1NN query is issued). The query result will consist of tuples $< [t_1, t_2], \{o_1\} >$, $< [t_2, t_3], \{o_3\} >$, ..., where each tuple $< [t_a, t_b], \{o_i\} >$ represents that object o_i is the 1NN of q within the time interval $[t_a, t_b]$.

To efficiently process C*K*NN query over moving objects in road networks, we first address the problem of how to significantly reduce the overhead of representing the network distance between moving objects at each timestamp. Although the network distance can be computed based on the Dijkstra's algorithm [9] or the A* algorithm [10] that have been shown to be simple and efficient for computing the network distance between stationary objects, recomputing the network distance whenever objects change locations would incur extremely high computational cost which makes the idea infeasible, especially for mobile environments in which objects move continuously. In order to greatly reduce the recomputation cost, we design a technique using the information about the relative speed of moving objects and the shortest path between network nodes. By exploiting this technique, the network distance between two objects at each timestamp is simply represented as a linear function of time, and thus can be easily computed. Another major problem we need to tackle is to avoid repetitively processing snapshot *K*NN queries at each timestamp. Let us consider again the example in Figure 1, where object o_3 replaces o_1 to be the 1NN at t_2 and is replaced by o_4 at t_3. We term these time points at which the *K*NNs set changes from one to another (e.g., t_2 and t_3) the *KNNs-changing time points*. An important characteristic of C*K*NN query is that the *K*NNs in between two consecutive *K*NNs-changing time points remain the same. Based on this characteristic, the problem of performing repetitive queries can be greatly reduced to finding the *K*NNs-changing time points and their corresponding *K*NNs. In this paper, we design a C*K*NN method combined with the proposed distance model to determine the *K*NNs-changing time points with corresponding *K*NNs.