Data & Knowledge Engineering 103 (2016) 19–43

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Nearest neighbor query processing using the network voronoi diagram

Mei-Tzu Wang *

Department of Information Management, Chinese Culture University, 55, Hwa-Kang Road, Yang-Ming-Shan, Taipei, Taiwan, ROC

ARTICLE INFO

Article history:
Received 25 April 2014
Received in revised form 21 February 2016
Accepted 23 February 2016
Available online 6 March 2016

Keywords:
Moving object
Nearest neighbor
Network voronoi diagram
Query processing
Spatial databases

ABSTRACT

The purpose of this paper is twofold: to develop sound and complete rules, along with algorithms and data structures, to construct the network voronoi diagram (NVD) on a road network. To compute the NVD, attention is focused on how to distinguish divisible paths from others, i.e., those whose midpoints are exactly their border points. Research shows that the dominance relation introduced in this paper plays an important role in making that distinction. To generate and prune candidate paths concurrently, a border-point binary tree is introduced. The pre-computed NVD is organized as linked lists and is available for access by a NVD list-based query search method (NVDL), which can compute NN in a single step. Experiments show that the NVDL method reduces execution time by 28% for sparse data distribution on a real road network compared to the existing INE method. The NVDL's query time remains nearly constant regardless of how data points are distributed on the road network or where the query point is positioned. In addition, this approach prevents the NVDL from experiencing the slow convergence condition that often occurs when using the incremental approach.

1. Introduction

As a type of spatial query, the nearest neighbor approach has been widely studied in the literature and applied in various applications [9,22,24,25,29]. Spatial object modeling that supports an increasing number of applications has been extended from merely describing the object's static feature (i.e., point, line, and region) with spatial operations to incorporating a discrete time dimension called the spatiotemporal data model [13,14,37]. In recent decades, the time dimension incorporated into the model has become a continuous type with the development of mobile computing that includes GPS. The data model then is called a moving object, which can support modern applications. This model can be used in applications related to satellites, passengers, and public security with the following post queries: “Which satellite will approach this spacecraft's route within the next minute?”; “Which taxi is closest to a passenger who requests service?”; “Did the trucks with dangerous goods approach a high-risk facility?” [14]. A moving query point's position can be estimated within a small deviation if a motion vector is stored in a moving object's database that can be updated in an event-driven approach [38].

There are two spaces for spatial queries, namely, unconstrained space and constrained space. Each has its own applications. In recent decades, the restricted space (e.g., road network), which is used in the second and third examples set forth above, has received more attention [6,8,18,20,28,35,39]. The major difference between the two types of spaces is that the distance used by the restricted space is measured by network distance, which is more costly than the Euclidean distance used by unrestricted space. This paper proposes a new method for evaluating a NN (nearest neighbor) query on restricted space where the query point is modeled as a moving object equipped with GPS service and the data points are static objects. This method will be particularly

* Tel.: +886 2 28610511 35922.
E-mail address: meii@faculty.pccu.edu.tw.

http://dx.doi.org/10.1016/j.datak.2016.02.003
0169-023X/

adequate for applications that demand a real-time response or that have sparse data points. The potential application is for emergency services, e.g., for a seriously hurt pedestrian who must quickly find the nearest hospital or for people who need to find the nearest exit quickly because the large building in which they are positioned is on fire.

Why do we want to find another method for evaluating NN when the literature has described many methods of doing so? There are two general approaches to finding NN: an online incremental approach and a pre-computed approach using indexes. The incremental approach is efficient in the average case. However, for the case in which the query point is far from all data points, there is a problem. It would take a long time to find NN because the incremental approach expands the search area incrementally (or slowly) from the query point to visit the vertex that is the most likely to reach NN; we call this "the slow convergence condition." Clearly, this case is impermissible in real-time applications. The second approach uses spatial indices to filter numerous data points to reduce the exact computation of NN in the refinement step. That approach must prepare some spatial index, and the filtering rate that affects the performance is not always satisfactory. Furthermore, it repeats computation for all candidates, although different types of distance are used during different steps. Clearly, this approach cannot also guarantee a real-time response.

One naïve idea for NN search in road networks is to pre-compute the network distances for all pairs of points, with one point representing the possible position of the query point and the other point representing the location of a data point. This pre-computing method is straightforward and infeasible because all possible positions of a query point correspond to uncountable points of the road network. The problem can be solved by computing a large scale of points instead of points such as edges when it is realized that all possible positions with the same NN are clustered. This set of positions forms a continuous part of the road network, such as an edge, an edge fragment, or successive edges or edge fragments. As provide an example, the road network in Fig. 1 is partitioned into several blocks, each of which has the same NN as shown in Fig. 2. When the query point is positioned at the point indicated by Fig. 1, the NN d_6 is found by looking up the 12th list in Fig. 2, which indicates that all points in edge (i, j) ranging from 12.5 to 16 relative to the initial vertex i have the NN d_6.

Our attention is now focused on the pre-computing method without indexing because that method is most likely to provide a uniform real-time response. The network voronoi diagram (NVD)-based method is our choice, which means we need to construct a network voronoi diagram for road networks that contains a set of data points in which we are interested. As the author knows, there is no existing algorithm to construct NVD. If we can do this work, then not only can we evaluate NN in real time in a single step without the disadvantages of the two NN approaches set forth above but also those who intend to find k-NN based on NVD such as VN^3 can use the constructed NVD. The contribution of this paper is summarized as follows.

1. Propose a set of sound and complete rules for constructing the NVD.
2. Provide data structures and algorithms to implement the NVD.
3. Organize the resulting NVD in linked lists to be accessed by the NN query evaluation method.
4. Conduct experiments on a real road network to show that NVD can be completely constructed by the proposed rules.
5. Conduct empirical experiments on a real network to show that NVDL has better performance than the incremental approach in situations involving sparse data distribution.

The rest of the paper is organized as follows. Section 2 reviews previous works, and Section 3 introduces basic terminology. In Section 4, some theorems are developed to distinguish divisible paths from others; thus, the proposed rules are developed. In Section 5, data structures, including adjacent lists, BPB trees, and NVD lists are introduced. Section 6 proposes algorithms to generate the NVD, and Section 7 conducts experiments on a real network to show that the NVD can be completely constructed; in addition, the NVDL's performance is evaluated. Section 8 provides our conclusion.

2. Related works

The incremental approach seems to be very efficient for finding the nearest neighbor on the road network because it always intends to visit the vertex that is most likely closer to NN among all of the vertices under consideration. The crucial factor in increasing efficiency is the strategy adopted to determine the next one to visit. However, not all strategies are always adequate in all conditions.

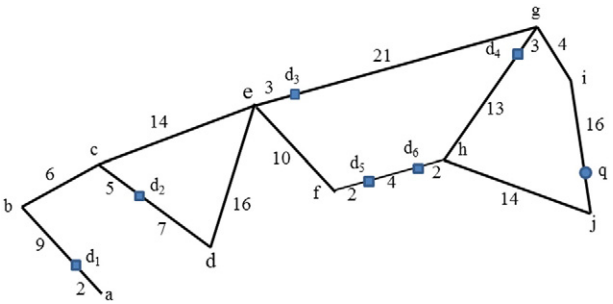

Fig. 1. Road network example.