Chapter IX

eDAR Algorithm for Continuous KNN Queries Based on Pine

Maytham Safar
Kuwait University, Kuwait

Dariush Ebrahimi
Kuwait University, Kuwait

ABSTRACT

The continuous K nearest neighbor (CKNN) query is an important type of query that finds continuously the KNN to a query point on a given path. We focus on moving queries issued on stationary objects in Spatial Network Database (SNDB). The result of this type of query is a set of intervals (defined by split points) and their corresponding KNNs. This means that the KNN of an object traveling on one interval of the path remains the same all through that interval, until it reaches a split point where its KNNs change. Existing methods for CKNN are based on Euclidean distances. In this paper we propose a new algorithm for answering CKNN in SNDB where the important measure for the shortest path is network distances rather than Euclidean distances. We propose DAR and eDAR algorithms to address CKNN queries based on the progressive incremental network expansion (PINE) technique. Our experiments show that the eDAR approach has better response time, and requires fewer shortest distance computations and KNN queries than approaches that are based on VN3 using IE.
INTRODUCTION

Several types of nearest neighbor (NN) search algorithms have been proposed and studied in the context of spatial databases. The most common type is the point KNN query, which is defined as: given a set of spatial objects (or points of interest), and an input query point, retrieve the (K) nearest neighbors to that query point. The NN is the target object with the shortest path from the query point on the route. The efficient implementation of KNN query is of a particular interest in geographical information systems (GIS). For example, a GPS device in a vehicle gives information of an object’s location, which, once located onto a map, serves as a basis to find the K closest restaurants or gas stations with the shortest path to them.

Different variations of KNN queries have been introduced. One variation is the continuous KNNs of any point on a given path. As an example, when the GPS device of the vehicle initiates a query to continuously find the three nearest gas stations to the vehicle at any point of a path from source to destination. The result is a set of intervals or split points where the KNNs of a moving object on a path will be the same up to these points. The versatility of K nearest neighbors search increases substantially if we consider other variations of it such as the continuous KNN (CKNN). CKNN query is defined as the K nearest point of interest to every point on a path, and has found applications in the GISs (e.g., find my nearest three gas stations at any point during my route from city A to city B). The result of this type of query is a set of tuples \(<\text{result}, \text{interval}>\) such that \(\text{result}\) is the KNN of all points in the corresponding interval ordered by distances to the query point. The \(\text{interval}\) is defined by two end-points called \(\text{split points}\), which specify where on the path the KNNs of a moving object will change. This means that the KNNs of an object traveling in one interval of the path remain the same all through that interval until it reaches a split point where its KNNs change.

In spatial network databases, as well as in practice, objects are restricted to move only on a predefined set of paths as specified by the underlying network (road, railroad, river, etc.) Thus, the network distance (i.e., the length of the shortest path connecting two objects is what represents the real distance between objects and not the objects’ relative position in space). Surprisingly, most of the existing work considers only the Cartesian (Euclidean) spaces where the path between two objects is the straight line connecting them, which renders the distance computation impractical for spatial network databases (SNDBs.)

Our objects are restricted to move on pre-defined paths (e.g., road) that are specified by an underlying network. This means that the shortest network path (distance) between the objects (e.g., the vehicle and the gas station) depends on the connectivity of the network rather than the objects locations. In Safar (2005), we proposed a Voronoi-based progressive incremental network expansion (PINE) technique to solve the KNN queries. The main idea behind PINE is to first partition a large network into smaller regions by generating a network Voronoi diagram over the points of interest. Each cell of this Voronoi diagram is centered by one object (e.g., restaurant) and contains the nodes that are closest to that object in network distance. Next, we pre-compute the inter distances for each cell (i.e., distances across the border points of the adjacent cells). This would later reduce the pre-computation time and space to answer KNN queries. To find the first nearest neighbours of a query object, we find the first nearest neighbour by simply locating the Voronoi cell that contains the query object, then starting from the query point, we perform network expansion by computing the distance from our query to its first nearest neighbour (its Voronoi cell center point) and exploring the objects that are close to the query object then computing their distances to the query object during the expansion. In PINE, at the first scale inside the Voronoi cell, which contains the query object, the network expansion