1 Introduction

Motivation Rapid growth of smart phones and online mapping services (e.g., Google Maps, Microsoft MapPoint) has led to the need to respond in real-time to range queries, shortest path queries along the network between locations, nearest neighbour queries from a set of facilities (such as restaurants, theaters, gas stations) lying along the network. As a result, there has been a lot of work done in the broad field of spatio-temporal database community for finding real-time solutions for queries on spatial networks.

Range search query on a spatial network The formal problem statement is as follows: We have a planar spatial network $G = (V, E)$, where $V$ represents the set of nodes/vertices (i.e., intersection points of roads) and $E$ is the set of edges connecting any two vertices. With each edge $e \in E$, we shall associate a weight $w(e)$ which is the length of the edge. A set $S$ of facilities are lying along the network. The user gives a two-dimensional query point $q$ and a query value $d_q$. The objective is to report all the facilities in $F$ which are within a network distance of $d_q$ from $q$.

Objective The major obstacle which comes in the way of coming up with an efficient solution for this problem is the following: calculating the network distance between two objects takes non-constant time (i.e., $\Omega(1)$), unlike a euclidean metric where distance between two objects can be found in $O(1)$ time. The major objective of this work is to devise a technique which will allow us to find network distance between two objects efficiently.

Currently there exist a couple of good techniques to answer the 'range searching query' on a spatial network. Sadly these are all experimental solutions with no provably good time guarantees. It is possible to construct scenarios where these solutions would fail to do any better than a brute-force approach. In our work we would like to give theoretical bounds to the algorithms we shall devise. This will lead to a more concrete understanding of this problem.

2 Related Work

Euclidean space For static objects there has been extensive work done on range searching queries in the database and computational geometry community. Orthogonal range searching query is perhaps one of the most widely studied problem in the computational geometry and database community due to its wide range of applications. Here the set of objects are points lying in a $d$-dimensional space and the query is an axis-parallel orthogonal query box.

Traditionally, the researchers in computational geometry have been interested in building solutions for this problem which aim at coming up worst-case efficient query time solutions, i.e., ensuring that the query time is low for all possible values of the query. The most popular among them are the range-trees [10] and the K-d tree [6]. The original range-tree when built on $n$ points in $d$-dimensional space took $O(n \log^d n)$ space and answered queries in $O(\log^{d-1} n + k)$ query time, where $k$ are the number of points lying inside the query region $q$. By using the technique of fractional cascading we can reduce the query time by a log factor [8]. In a dynamic setting, the range-tree uses $O(n \log^{d-1} n)$ space, answers queries in $O(\log^{d-1} n \log n)$ time and handles updates in $O(\log^{d-1} n \log \log n)$ time [8]. A K-d tree when built on $n$ points in $d$-dimensional space
takes up linear space and answers a range-query in \(O(n^{1-1/d} + k)\) time. Updates can be handled efficiently in it.

In the field of databases there have been a significant number of index structures proposed to answer an orthogonal range-query. The researchers in database community try to come up with practical solutions which are optimized to work well for the average case queries (with the assumption that the worst-case query will occur rarely). Perhaps no other structure has been more popular than the R-tree proposed in 1984 by Guttman [17]. According to Google scholar, as on 16th June 2011 this paper has been cited 5260 times! R-trees is extremely popular for indexing multi-dimensional information. It’s main advantage arises from the fact that it can handle multiple queries efficiently (range-queries being one of them). Contrast this with a range-tree which is tailor made for handling only range-queries. Variants of R-tree such as R* tree [5], R+ tree [30], Hilbert tree and Priority R-tree are also quite popular index structures for storing multi-dimensional data.

**Circular range searching** [7] and **Halfspace range searching** [7] are some other important range queries where the point set are queried with a disk and a halfspace, respectively.

**Range queries on moving objects** In the field of moving object databases, there has been work on indexing moving objects which can answer range queries or window queries. However, most of these assume the objects to have an unconstrained two-dimensional movement. A couple of index structures such as FNR-tree [11], MON-tree [9] and [25] assume the objects to be constrained to move along the spatial network and appropriately model the index structures. These structures store the past history of movement of all the objects and range queries are asked on this past data. However, the range queries which they handle are different than what we are attempting in this paper. The range query which they want to answer is the following: Given a spatio-temporal window, return all the objects whose movements overlap the query window. So in some sense they still consider euclidean metric distance.

**Range query on spatial networks** The work in [24] presents the first known index structure and algorithms for spatial queries such as range queries, nearest neighbours, spatial joins and closest pairs using the network distance instead of the euclidean distance. For the range query they propose two algorithms: a) **Range Euclidean Restriction (RER)**, and b) **Range Network Expansion (RNE)**.

In RER technique, first a circular range query is performed on \(S\) to retrieve a subset \(S'\) of facilities which lie within a euclidean distance of \(d_q\) from the query point \(q\). This helps in taking care of false misses. From the query point \(q\), we shall be expanding along the network to find out all the edges in the network which are within a ‘network distance’ of \(d_q\) from \(q\). All the edges are visited in increasing order of their ‘network distance’ (similar to Dijkstra’s algorithm). For each edge visited, all the facilities in \(S'\) which have an overlap with the edge are reported. Clearly we can construct inputs for which RER technique will fail miserably. In an extreme case it might happen that all the facilities in set \(S\) fall within euclidean distance of \(d_q\) from \(q\), but none of them fall with network distance of \(d_q\) from \(q\). In such a case no facility will get reported but it might happen that we explore \(O(|E|)\) edges in the network in the process. The overall query time would be \(O(|E||S|)\) (ignoring the cost of maintaining a queue) and we end up reporting none of the facilities!

In RNE technique they first compute a set \(QS\) of all the edges in the network which are within a network distance of \(d_q\) from \(q\). The facilities are indexed in an R-tree. An intersection join in performed between \(QS\) and the facilities R-tree which is shown to be optimal. Alternate techniques for performing the join are also mentioned. Though the join operation is efficient, the first step of finding the set \(QS\) is the major bottleneck of this process. In an extreme scenario, it might happen that \(O(|E|)\) edges in the network might be within a network distance of \(d_q\) from \(q\) but none of the facilities lie on them!

There a couple of other works [29] which claim to handle range queries based on network distance but they do not provide any strong theoretical bounds on the space and the query time requirements.

**Highway dimension** Recently there was a paper [1] which introduced the notion of highway dimension. They observed that there were a lot of heuris-
tics for finding shortest-path among vertices which are based on a wide variety of ideas, such as arc flags [21, 18, 4], A* search with landmarks [15], highway hierarchies [27, 28], reach [14, 13, 16], transit nodes [2, 3], and contraction hierarchies [12]. In experiments on real-world data these heuristics performed a lot better than a plan Dijkstra. Also the preprocessing involved in them is practical and space occupied was only slightly more than than the road network alone. However, these were all purely experimental with no good theoretical bounds. It is possible to construct inputs for which these heuristics will perform miserably.

In [1] they tried to bring all these techniques within a theoretical framework and tried to identify factors which lead to good performance of these heuristic techniques. In this process they define the notion of highway dimension and showed that having ‘small’ highway dimension gives a theoretical guarantee of good query performance for these heuristic algorithms. Interestingly, the line of approach of this work seems promising for coming up with good theoretical guarantees for algorithms which perform range queries on spatial networks.

Other problems Apart from the shortest path query, the nearest neighbour or K-nearest neighbour query in terms of network distance happens to be an active field of research [31, 19, 26, 23]. It looks possible that we can borrow ideas from shortest path and K-nearest neighbour algorithms for answering the range query problem.

In [32] they provide various techniques to estimate the number of nodes and edges which lie within $d_q$ distance of the query point $q$ which might help us in predicting the space and query time requirements for range queries. In some scenarios users of location based services would not want to reveal their local information. In [20] they come up with nearest neighbour and range search queries on spatial network with network distance without revealing the local information of the user. In this work we are looking at static range queries but there has been work done on dynamic range queries in spatial networks [22].

References


