International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 2, February- 2013

User Query Processing Using Distance Oracle For Road Networks

Anil S Naik
Assistant professor
Department of Information Technology, WIT, Solapur University, Solapur

Abstract

The popularity of location-based services and the need to perform real-time processing on them has led to an interest in queries on road networks, such as finding shortest paths and finding nearest neighbours. The challenge here is that the efficient execution of operations usually involves the computation of distance along a Road network instead of "as the crow flies," which is not simple. This requires the precomputation of the shortest paths and network distance between every pair of points (i.e., vertices) with as little space as possible rather than having to store the n^2 shortest paths and distances between all pairs. A data structure called a road network oracle is introduced that resides in a database and enables the processing of many operations on road networks with just the aid of relational operators. Two implementations of road network oracles are presented. This Paper investigates the problem of efficiently computing exact and approximate shortest paths in graphs, with the main focus being on shortest path query processing. Strategies for computing answers to shortest path queries may involve the use of pre-computed data structures (often called distance oracles) in order to improve the query time. To reduce amount of Storage Space in Database using distance oracle concept.

1. Introduction

The growing popularity of online mapping services such as Google Maps, Microsoft Bing, and Yahoo! maps has led to an interest in responding to queries in real time, such as finding shortest routes between locations along a Road network as well as finding nearest objects from a set *S* (e.g., gas stations, markets, college institutions and restaurants) where the distance is measured along the shortest path in the network. Elements of *S* are usually constrained to lie on the network or at the minimum to be easily accessible from the network. The online nature of these services means that responses must be generated in real time. The challenge in performing queries on road networks is that operations involve the computation of distance along a Road network (i.e., network distance) instead of "as the crow flies," which is not simple.

Distance Oracle technique goal is to be able to determine the network distance between any pair of points (i.e., vertices) without having to store the n^2 distances between all pairs. We are willing to expend a bit more time to achieve this such as O(log n) instead of O(1) as well as accept an error in the accuracy of the distance that is provided. The strategy that we follow reduces the space requirements to as low as $O(n/\varepsilon^2)$, and is based on the ability to identify groups of source and destination vertices for which the distance is approximately the same within some ε.

2. Literature Review

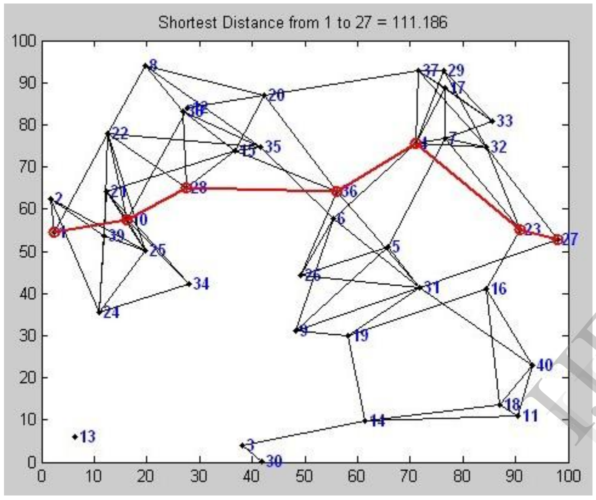
Operations on road networks are expensive because computing distances between two objects (e.g., postal addresses) on the road network requires the invocation of a shortest path algorithm). A popular shortest path algorithm is Dijkstra's algorithm, which if invoked between a source vertex *q* and a destination vertex *v*, ends up visiting every vertex that is closer to *q* via the shortest path from *q* than *v*.

The single source shortest path problem can be described as follows:

Let G= {V, E} be a directed weighted graph with V having the set of vertices. The special vertex s in V, where s is the source and let for any edge e in E, Edge Cost(e) be the length of edge e. All the weights in the graph should be non-negative.

A drastic alternative to the use of Dijkstra's algorithm is to precompute and store the shortest paths between all possible vertices in the Road network.The algorithm works as follows for a given source vertex (node) in the graph, it finds the path with lowest cost (i.e. the shortest path) between that vertex and every other vertex. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined. For example, if the vertices of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities. As a result, the shortest path first is widely used in network routing protocols, most notably IS-IS and OSPF (Open Shortest Path First).

In particular, it is not uncommon for Dijkstra's algorithm to visit a very large number of the vertices of the network in the process of finding the shortest path between vertices that are reasonably far from each other in terms of Figure 1: calculates the shortest path and distance between two nodes on a map: Shortest distance between node 1 to 27.



- ➢ The major disadvantage of the algorithm is the fact that it does a blind search there by consuming a lot of time waste of necessary resources.
- ➢ Another disadvantage is that it takes more Space to store n number of vertex.

3. Proposed System

We propose a new system which is based on Distance Oracle. The Distance Oracle can Provide the shortest path between any two places in Road Network in O(log n) time (less Query Execution time compared to Dijikstra's Algorithm) while reducing the storage to O(n) or O(n log n). Also it can store the distances in a table , so it is possible to transform search operations to SQL queries.

The techniques that we develop in this Paper is based on our inference that given our assumptions on the proximity of the vertices that comprise A and those that comprise B, and the lack of proximity between A and B, that the network distance to the vertices in A from the vertices in B will more or less be similar and can be approximated by a single value (termed the path coherence property).

The novelty of our approach is that in the case of the computation of the distance between two vertices, we show how to correlate the extent of this reduction of the space requirements with the approximation error in the value of the distance that is obtained. This is achieved via the introduction of a more general construct, termed an approximate distance oracle for Road networks, that is capable of responding to network distance queries between any two vertices of the Road network with a specified approximation ε—that is, given a start vertex u and a destination vertex w in Road network G, the network distance $S_{\varepsilon}(u,w)$ produced by the oracle Sε is no more or less than an ε fraction of the actual network distance $d_G(u,w)$ between u and w in G.

4. Objectives & Scope

To implement a Distance oracle based distance search system for a Road Network. The users of the system must be able to search the distance between any two places in the system and the path. Also users can use this system to find places near an area around some fixed radius.

We implement the Distance Oracle system and compare the performance of the system with the Dijikstra algorithm against time and memory and prove our proposed system works better

5. Approximate Distance Compute Implementations

The techniques that we develop in this Paper is based on our inference that given our assumptions on the proximity of the vertices that comprise A and those that comprise B, and the lack of proximity between A and B, that the network distance to the vertices in A from the vertices in B will more or less be similar and can be approximated by a single value (termed the path coherence property).

The use case diagram captures the requirements of the system in brief as shown in fig 5.1. The novelty of our approach is that in the case of the computation of the distance between two vertices, we show how to correlate the extent of this reduction of the space requirements with the approximation error in the value of the distance that is obtained. This is achieved via the introduction of a more general construct, termed an approximate distance Oracles for Road networks, that is capable of responding to network distance queries between any two vertices of the Road network with a specified approximation ε—that is, given a start vertex u and a destination vertex w in Road network G, the network distance $S_{\varepsilon}(u,w)$ produced by the Oracles Sε is no more or less than an ε fraction of the actual network distance $d_G(u,w)$ between u and w in G.

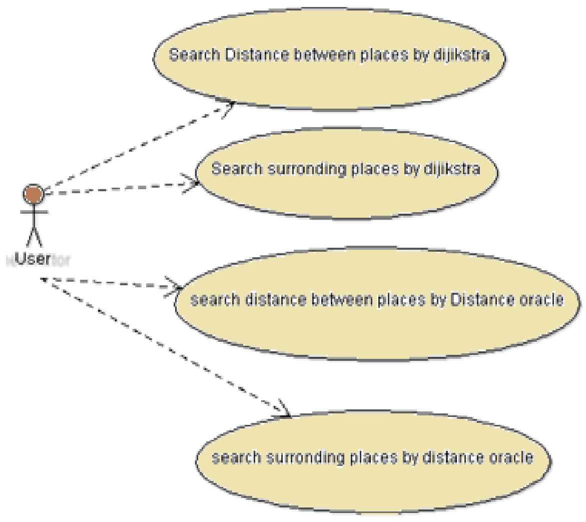

Search Distance between places by dijikstra
Search surronding places by dijikstra
User
search distance between places by Distance oracle
search surronding places by distance oracle

Fig 5.1 use case diagram of a system.

5.1 Pseudo Code for Distance Oracles

```
1  Input :  S  - place , D – destination
2  Output :  distance
3  A  = getNearestFromDB(S);
4  B = getNearestFromDB(D);
5  ResultSet rs = ExecuteQuery("select * from distancemat where sourcevertex=A and destvertex = B ");
6   Distance = rs.distance;
```

5.2 Algorithm for Finding Places around R by using Distance Oracles

```
1  Input : R   - place , D – distance
2  Output : Neighbour
3  for  i=1 : no of nodes
4  if distanceusingOracles(node(i) , R)<= D
5  addtoNeighbour(Neighbour,node(i));
6  end node(i);
7  end;
```

Algorithm finds Neighbours places around R with in Mentioned range. It checks all nodes with in Particlar range and Displays all Place names. First it finds nodes neighbours to Source node, add to list if user Queries neighbor nodes then it displays on GUI.

5.3 Road Network Oracles

Our precomputed input representation of a road network containing *n* vertices is of size $O(n^3)$. Our goal here is to convert it to a database friendly representation that is much smaller in size. The Road Network Oracles (or simply *Oracles*) *O* of a road network *G* is a data structure that completely encapsulates all the n^2 shortest paths and network distances between every pair of vertices in *G*. The obvious storage choice for *O* in a relational database system is a *database relation*.

The schema of a relation that captures the shortest paths is given by: $O(AB, \Psi)$, where *AB* represents the group identity of the vertices in the road network belonging to *A* and *B*, such that is the common intermediate vertex Ψ on the shortest paths between them. Another relation records the network distances separately using another relation with the schema: $O(AB, d_{apx})$, where again *AB* is the group identity of the source and destination vertices whose network distances are approximated with d_{apx}. Of course, for the sake of simplicity, if we assume (not without merit) that the vertices that make up *AB* in the relation storing the shortest paths are also the same ones that make up *AB* in the relation storing the network distances, then we can combine these two relations into a single relation with the schema: $O(AB, \Psi, d_{apx})$.

5.3.1 Operations on Road Networks using SQL:

Given an implementation of $O(AB,\Psi \quad , d_{apx})$, we now show how we can efficiently perform query processing on it using SQL. In particular, we demonstrate how many operations on road networks can be expressed using SQL (and relational operators) in the context of a database system. Our example assumes the following setup. Let *R* be a relation of restaurants with schema *(id, type, price)*, where *id* uniquely identifies restaurants on the road network, *type* is the kind of the cuisine served by the restaurant, and *price* is the average cost.

We also define another relation *Q* of movie theaters given by the same schema *(id, movie id)* where *id* uniquely identifies the movie theater on the road network, and *movie id* is a movie playing in the theater. Given a source *p* and destination *q*, let $Z(p, q)$ map them to a group key *AB* such that *A* contains *p*, and *B* contains *q* so that one can find a tuple in *O* with the aid of B-tree on *O.AB* such that the value of *AB* equals $Z(p, q)$. We present the following queries on a Road network.

Approximate Network distance: Run the Paper and given source and destination Network distance can be obtained by:
query ="select * from distancemat where srcvertex= " + from + " and destvertex=" + to + "";
Region Search: Given a Query location q obtain all restaurants in R that are within 10 miles of q which serve Italian food. Of course this query would make more sense if the Oracles can given a ε guarantees on the quality of the network distance answers.
k-Nearest neighbour Search: Given a Query location q,determine the closest restaurant in R to q which serve Italian food.

6. System Architecture

The System architecture is shown below.

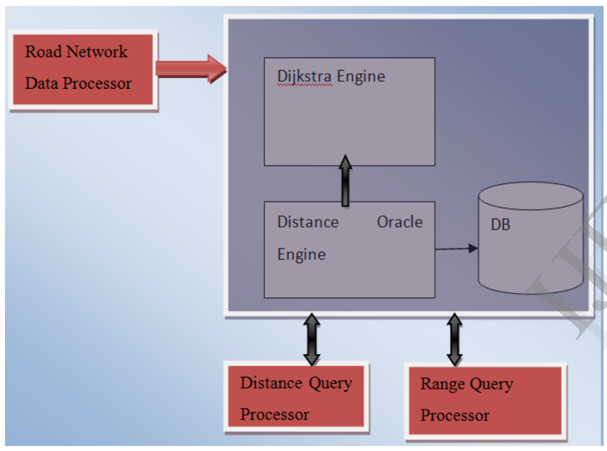


Figure 6.1 System Architecture

The system is divided into following modules.
There are totally 5 modules of the System these are Road Network Data Processor, Dijkstra's engine, Distance Oracles Engine, Distance Query Processor, Range Query Processor.

- Road Network Data Processor: This module reads the road data matrix in the spread sheet of .xsl format and stores in internal data structure of graph with nodes and connections.
- Dijikstra engine: This module implements the Dijikstra algorithm. By using the Dijikstra shortest path algorithm, it can provide the distance between two points.
- Distance Oracles Engine: This module implements the Distance Oracles algorithm to store the reduced set of linked nodes and their distance pre computed using Dijikstra into the Database.
- Distance Query Processor: This module finds the shortest distance between two points by using Dijikstra Algorithm and then Distance Oracles. The results of path and the time taken to compute the path are displayed in GUI.
- Range Query Processor: This module finds the places around any point within a range of distance d provided by the user by using Dijikstra Algorithm and then Distance Oracles. The results of places and the time taken to compute the path are displayed in GUI.
- Database: MySQL database store data in two tables *distancemat* and *approxlocation* , distanemat table contains distance between places stored in table and approxlocation table contains vertex with nearest nieghbour nodes.

7. Experimental Result

7.1 Performance of time and Storage Space by Dijikstras's Algorithm and Distance Oracles Algorithm.

Figure 7.1 and 7.2 Shows Compare Query time for execution and Storage Space by Dijikstras's Algorithm and Distance Oracles Algorithm respectively.

- The Performance graph for comparing time against the size the road network between Dijikstra and distance Oracles must be displayed
- The Performance graph for comparing Storage Space for the road network.
- Fig 7.1 Shows Size of road network increases, Storage Space increases by using Dijkstra's Algorithm and Compare to Distance Oracles memory Space will be linear. It takes very less space compared to Dijkstra's Algorithm.
- The main Paper goal is to Reduce the Query execution time and Storage Space.

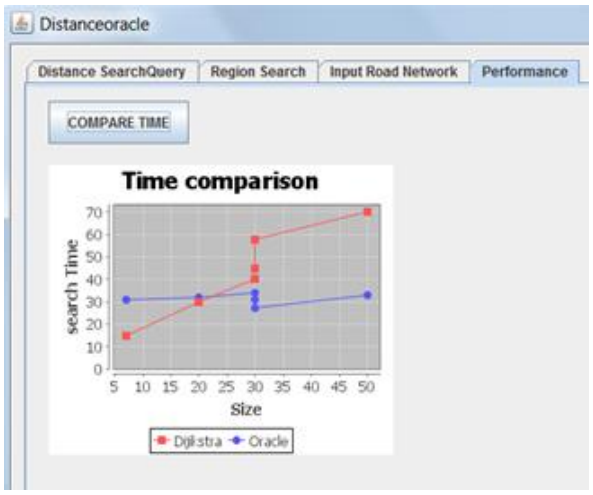


Figure 7.1: Performance of Algorithms in Query execution time comparison.

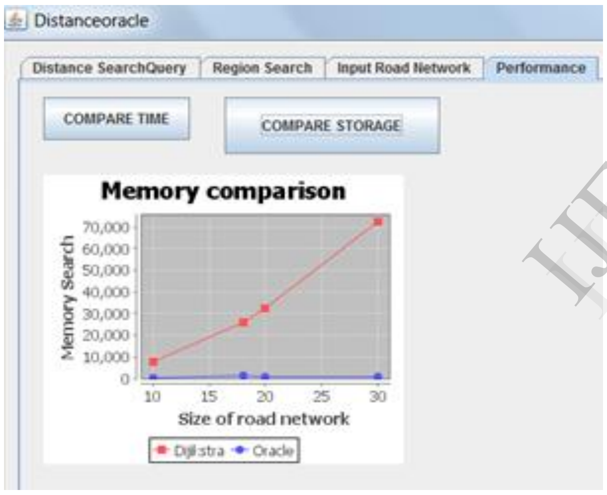


Figure 7.2: Performance of Algorithms in Storage Space comparision

8. Conclusion & Future Enhancements

In this Paper we presented an alternate way of performing operations on road networks using road network Oracles, which reside in a database system and enable operations on road networks using SQL. Our approach of converting road networks into Oracles provides an opportunity to move away from traditional methods of working with road networks towards a way that is scalable, efficient, database friendly, and being able to support Internet-scale real time operations.

We presented a few possible implementations of the Oracles that use the Road Information in a road network to provide group memberships to vertices in the road network. The path-distance Oracles can be shown to be linear in n as well as being able to provide an intermediate vertex in the shortest path as well as an ϵ-approximate network distance between any pair of vertices drawn from the road network. Distance Oracle Algorithm processes query faster and the Storage Space is less Then Dijikstra's Algorithm.

Another open problem is whether dynamic Oracles can be designed so that they can deal with updates as is the case when road segments or vertices become closed or road segments become one way, as well as other dynamic traffic conditions such as congestion which may make some routes more acceptable. Finally, there is the issue of how to optimize operations involving the Oracles so that operations can be optimized effectively in the context of a relational database system. In particular, how can a query optimizer be taught more strategies to perform road network queries more effectively.

References

[1] J. Sankaranarayanan and H. Samet, "Distance Oracles for Road Networks," *Proc. IEEE Int'l Conf. Data Eng. (ICDE)*, pp. 652-663, Apr. 2009.

[2] Christian Sommer, and Y. Tao, "Approximate Shortest Path and Distance Queries in Networks," *Proc. Conf. Very Large Data Bases (VLDB)*, pp. 802-813, Sept. 2003.

[3] H.-J. Cho and C.-W. Chung, "An Efficient and Scalable Approach to CNN Queries in a Road Network," Proc. *Conf. Very Large Data Bases (VLDB)*, pp. 865-876, Sept. 2005.

[4] N. Jing, Y.-W. Huang, and E.A. Rundensteiner, "Hierarchical Encoded Path Views for Path Query Processing: An Optimal Model and Its Performance Evaluation," *IEEE Trans. Knowledge and Data Eng.*, vol. 10, no. 3, pp. 409-432, May 1998.

[5] S. Jung and S. Pramanik, "An Efficient Path Computation Model for Hierarchically Structured Topographical Road Maps," *IEEE Trans. Knowledge and Data Eng.*, vol. 14, no. 5, pp. 1029-1046, Sept./Oct. 2002.

[6] H. Samet, Foundations of Multidimensional and Metric Data Structures. Morgan-Kaufmann, 2006.

Mr. Anil s Naik received the B.E degree in Electronics and Communication Engineering in 2009 from VTU University, Belgaum, India and M. Tech degree in Information Technology in 2011 from VTU Belgaum, India. He is presently working as an Assistant Professor in Department of IT Walchand Institute of Technology, Solapur, Maharashtra, India. His research interests include Data Mining,Software Engineering.