IPHV7I20037X
International Journal Of Advanced Research and Innovation -Vol.7, Issue .II
ISSN Online: 2319 – 9253
Print: 2319 – 9245

EFFECTIVE EVALUTING PATH QUERIES OVER DATA RATE COLLECTION

[#1]B.SRI VANI-M.Tech Pursuing,
[#2] R.RADHA-Associate Professor,
Department of Computer Science & Engineering,
Malla Reddy College Of Engineering & Technology, Hyderabad.

Abstract: The recent advances in the infrastructure of Geographic Information Systems (GIS), and the proliferation of the GPS technology, have resulted in the abundance of geodata in the form of sequences of spatial locations representing points of interest (POIs), landmarks, waypoints etc. We refer to a set of such sequences as route collection. In many applications, the route collections are frequently updated as new routes are continuously created and included, or existing ones are extended or even deleted. This thesis studies three problems where given a frequently updated route collection the goal is to find a path, i.e., a sequence of spatial locations, that satisfies a number of constraints. According to the first problem a large collection of touristic routes is available and the goal is find a path that connects two landmarks through locations contained in the routes. Second, we focus on the pickup and delivery problems that appear in various logistics and transportation scenarios. A company that offers pickup and delivery services has already scheduled its fleet of vehicles to follow a collection of routes for servicing a number of customer requests. However during the day, new ad-hoc requests arrive at arbitrary times, and the objective is to find sequences of locations from the vehicle routes, i.e., paths, for picking up and delivering the new objects, and minimizing at the same time the company's expenses. Finally, we consider the problem of providing driving directions from one location of a city to another. In this context a collection of vehicle routes is constructed using everyday driving data on the road network of the city. This collection provides a trusted and "familiar" way of driving through the city, in other words it defines a "known" part of the city's network. The drivers consult the collection whenever they want to travel from one location to another, seeking for a path such that they will drive as less time as possible outside the known part of the road network without significantly increasing, at the same time, the total duration of their journey compared to the fastest way, i.e., the shortest path.

Keywords: GIS,POIs,GPS,RTS.

I. INTRODUCTION

Nowadays, severall applications involve storing and querying large volumes of data sequences. For instance, the recent advances in the infrastructure of geographic in formation systems(GIS), and the proliferation of the GPS technology, have resulted in the abundance of geodata in the form of sequence.We refer to a set of such sequences as route collection. The characteristics of a route collection vary with respect to the application requirements and its context. In some cases, routes are the only type of data available while in other cases, a graph, e.g., a road network, is primarily available and the routes are created by means of traversing this graph. Further, in some scenarios the routes are directly created and provided by the users whereas in other cases, they are generated after preprocessing the available data. Finally, in many applications, the route collections are frequently updated as new routes arecontinuously created and included, or existing ones are extended or even deleted. Inthe following we discuss three examples of route collections.

The 1first, r1, starts from the National Technical University of Athens and ends at the Museum of Acropolis. The second, r2, starts from the Omonia Square and ends at the Acropolis Entrance. Web sites such as www.ShareMyRoutes.com and www.TravelByGPS.com maintain a huge collection of routes, like the above, with POIs from all over the world. These collections are frequently updated as users continuously share new interesting routes. As a final example of route collections, consider a group of people that track their every day movement with their cars. Given the road network of a city, the movement of a vehicle is captured by a sequence of road intersections, in other words by a route. Then, these people share their driving data and thus, a collection of vehicle routes is defined. This route collection can be viewed as a trusted way of driving through the city, which is frequently updated as people make their way to previously unknown parts of the city or identify better ways for reaching already known locations. The drivers consult this shared route collection whenever they want to travel from one location of the city to another.route collection can be trivially transformed to a graph; hence, path and reachability queries can be evaluated using standard graph search techniques. Such methods follow one of two paradigms. The first employs graph traversal methods, such as depth-first search. The second compresses the graph's transitive closure, which contains reachability information, i.e., whether a

path exists between any pair of nodes. Both paradigms share their strengths and weaknesses. While the latter techniques are the fastest, they are mostly suitable for datasets that are not frequently updated, or when the updates are localized, since they require expensive precomputation. On the other hand, the former are easily maintainable, but are slow as they may visit a large part of the graph.As a final challenge, even new graph queries inspired by the existence of a route collection can be introduced. For example, consider the collection of vehicle routes constructed using every day driving data in the road network of a city. The problem of providing driving directions from one location of the city to another is a well studied problem and usually it is solved as a shortest path problem. However, givena share collection of vehicle routes and therefore, a trusted way of driving through the city, we introduce a new path query that captures the actual way people drive through a city. Particularly, people tend to follow roads they use in their every day life or roads that have followed in the past. In addition, even when they want to drive to a location for the first time, they usually ask their friends to recommend a "good" and safe way.

II.CONTRIBUTIONS

Based on the above observations, the contributions of this thesis include the following:
We target path query evaluation on large disk resident route collections like the ones containing touristic routes, that are frequently updated.The proposed framework, i.e., the indices and the traversal policies, constitutes the basis for applying our work to other types of queries under various constraints. An extensive experimental evaluation verifies the advantages of our methods compared to conventional graph-based search.

We formulate dynamic Pickup and Delivery with Transfers (dPDPT) as a path problem.For this purpose, we introduce a conceptual graph, called dynamicplan graph that captures all possible actions for picking an object and delivering it to the destination with respect to the existing vehicle routes. Then, we define two cost metrics, termed operational and customer cost, that captureour method is significantly faster than a two-phase local search method inspired by the related work, while the quality of the solution is only marginally lower.
We introduce the Most Trusted Near Shortest Path (MTNSP) as a preferable way of driving through a city when a collection of trusted vehicle routes is available. For this purpose, we define the notion of the known graph as a subgraph of the road network that is constructed merging the available trusted routes. We also define two costs for a path between two locations on the road network, measuring the total traveling time needed and the total time spent outside the known graph.Finally, we propose a methodology for identifying the path that has the lowest total time outside the known graph among the paths with length, at most α times larger than the length of the shortest path as the answer to a MTNSP query. An extensive experimental analysis shows the advantage of our methodology compared to a label-setting algorithm that exploits the euclidean distance of the network intersections to prune its search space. The methodology discussed and the results obtained appear in [10].

III.RELATED WORK

Techniques for evaluating path/reachability queries follow two paradigms: (1) searching, and (2) encoding the graph's transitive closure (T C). Searching methods deal with path queries, while T C methods primarily target reachability queries. As wediscuss next, some of the T C techniques can be extended to evaluate path queries. Table 2.1 summarizes the related work in terms of the: (1) graph type supported, (2) support for reachability query, (3) support for path query, and (4) capacity to handle updates.
Searching. The simplest way to evaluate path queries is to traverse the graph at query time exploiting a search algorithm, e.g., depth-first or breadth-first search [27]. This approach has minimum space requirements, since it only needs to store the adjacency lists of the graph. In addition, the adjacency lists can be easily updated. On the other hand, in the worst case, it may need to visit all nodes of the graph to answer a query. Encoding the T C. The transitive closure (T C) of a graph G(N, E) is the graph G∗(N, E∗), where an edge (ni , nj) is in E if a path from ni to nj exists in G. Using T C a reachability query can be answered in constant time. However, even though efficient algorithms for computing the T C have been proposed, e.g., [2, 50, 39], the computation and storage cost are prohibitive for large disk-resident graphs.Therefore, various methods compress the T C.
Computing the optimal 2-hop scheme is NP-hard. The work in [22] and [23]presents an approximation algorithm based on set covering [43] that constructs a 2-hop scheme larger by a logarithmic O(log|N|) factor than the optimal one, but it still requires the computation of the T C. Therefore, this approach cannot be applied to large graphs. In addition, the work does not handle frequent updates. Compared to 2-hop our methodology is less efficient in evaluating path queries, but is significantly cheaper to construct and maintain. here is a number of works that transform the input graph to a DAG by replacing each strongly connected component with a super node. For example, [20] proposes a geometric-based method and [21] another one based on graph partitioning for the

efficient construction of 2-hop. [40] proposes the 3-hop indexing scheme. The basic idea is to use a chain of nodes, instead of a single node, to encode the reachability information. [1] proposes a labeling scheme that assigns to each node a sequence of intervals, based on the postorder traversing of graph's spanning tree. Updates are handled by leaving gaps in postorder numbers. Although not discussed, path queries can be answered on the DAG by computing the ancestors of the target node.

Algorithm RTS

Input: nodes ns and nt of a route collection R, R-Index
Output: a path from ns to nt
Parameters:
stack Q: // the search stack
set H: // contains all nodes pushed in Q
set A: // contains the direct ancestor of each node in H
Method:
1. push ns to Q;
2. insert ns in H
3. while Q is not empty do
4. popnq from Q; // pop current search node nq
5. if there is a route rc∈ R containing nq before nt then
returnConstructPath(ns, nq, nt, A, rc);
6. for each entry hri :oqii in routes[nq] do
7. let nr be the node after nq in ri;
8. while nr ∈ H/ do // access each node nr after nq
// in r until the first nr node
// contained in H
9. push nr to Q;
10. insert nr in H;
11. inserthnr , nri in A; // where nr is the direct
// ancestor of nr in ri
12. let nr be the next node in ri;
13. end while
14. end for
15. end while
16. return null;

IV. THE RTST ALGORITHM

RTST expands the search similar to RTS, but employs a stronger termination check based on the transitions between routes. This additional reachability information is modeled by the transition graph GT , and is explicitly materialized in the T –Index structure. Definition 2.8 (Transition graph). The transition graph of a route collection R is a labeled undirected graph GT (R, ET), where its vertices are the routes in R, and a labeled edge (ri , rj, nℓ) exists in ET ifri and rj share a link node nℓ. Intuitively, an edge (ri , rj, nℓ) in the GT denotes that all nodes in ri before link nℓ can reach those after nℓ in rj, and vice versa, i.e., all nodes in rj before nℓ can reach those after nℓ in ri.

V. CONCLUSION AND FUTURE WORK

We plan to extend our work in three directions: (i) address other kinds of updates on route collections, (ii) evaluate other types of queries mostly considering constraints, and (iii)combine query evaluation with keyword search. First, we plan to study the maintenance issues in cases apart from adding new routes. For example, in the dynamic pickup and delivery problem, new customer requests can be served by inserting nodes in existing vehicle routes. This update method will also change the time intervals of the nodes in the routes.
Second, we plan to evaluate queries similar to trip planning [18] and optimal sequenced route [19] queries. Specifically,consider a set of classes C such that each node n in aroute collection is an instance of a class in C, e.g., the nodes in a touristic route are instances of classes C = {Museum, Stadium, Restaurant}. An interesting query is to find a path from a node nsto nt that passes

first through a Museum, then a Stadium and finally a Restaurant.

REFERENCES

[1] P. Bouros, S. Skiadopoulos, T. Dalamagas, D. Sacharidis, and T. K.Sellis, "Evaluating reachability queries over path collections," inSSDBM, 2009, pp. 398–416.
[2] E. Cohen, E. Halperin, H. Kaplan, and U. Zwick, "Reachability and distance queries via 2-h labels," in SODA, 2002, pp. 937–946.
[3] R. Schenkel, A. Theobald, and G. Weikum, "Hopi: An efficientconnection index for complex xml document collections," inEDBT, 2004, pp. 237–255.
[4] J. Cheng, J. X. Yu, X. Lin, H. Wang, and P. S. Yu, "Fast computationof reachability labeling for large graphs," in EDBT, 2006, pp. 961–979.
[5] R. Agrawal and H. V. Jagadish, "Direct algorithms for computingthe transitive closure of database relations," in VLDB, 1987, pp.255–266.
[6] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. PODS, 2001.
[7] B. Bamba, L. Liu, P. Pesti, and T. Wang. Supporting anonymous location queries in mobile environments with PrivacyGrid. WWW, 2008.
[8] C. Bettini, S. Mascetti, X. S. Wang, and S. Jajodia. Anonymity in locationbased services: Towards a general framework. 8th Intl. Conf. on Mobile Data
Management *(*MDM*)*, May 2007.
[9] S. Berchtold, C. B¨ohm, D. A. Keim, F. Krebs, and H.-P. Kriegel. On optimizing nearest neighbor queries in high-dimensional data spaces. ICDT
2001.
[10] A. R. Beresford and F. Stajano. Location privacy in pervasive computing. IEEE Pervasive Computing, 2(1), 2003.
[11] R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar. Preserving user location privacy in mobile data management infrastructures. Privacy Enhancing Technology Workshop, Cambridge, UK, June 2006.
[12] K. Cheverst, N. Davies, K. Mitchell, and A. Friday. Experiences of developing and deploying a context-aware tourist guide: the GUIDE project.ACM MobiCom, 2000.