Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 11, November 2014, pg.258 – 265

RESEARCH ARTICLE

EVALUATING PATH QUERIES OVER FREQUENTLY UPDATED ROUTE COLLECTION

Miss. S.Deepa*, **Mr. M.Baskar**, M.Sc., M.Phil.**
*M.Phil(Computer Science), Research Scholar,
Vivekanandha College for Women, Unjanai, Tiruchengode, India
**Assistant Professor in Computer Science
Vivekanandha College for Women, Unjanai, Tiruchengode, India

ABSTRACT: The recent advances in the infrastructure of Geographic Information Systems (GIS), and the proliferation of GPS technology, have resulted in the abundance of geo-data in the form of sequences of points of interest (POIs), waypoints etc. To sets of such sequences as route collections. The path queries on frequently updated route collections: given a route Collection and two point's ns and nt, a path query returns a path, i.e., a sequence of points that connects ns to nt. The introduce two path query evaluation paradigms that enjoy the benefits of search algorithms (i.e., fast index maintenance) while utilizing transitivity information to terminate the search sooner. Efficient indexing schemes and appropriate updating procedures are introduced. An extensive experimental evaluation verifies the advantages of our methods compared to conventional graph-based search.

Keywords: GIS, RTS, MRSE, Data Mining, GPS

1. INTRODUCTION

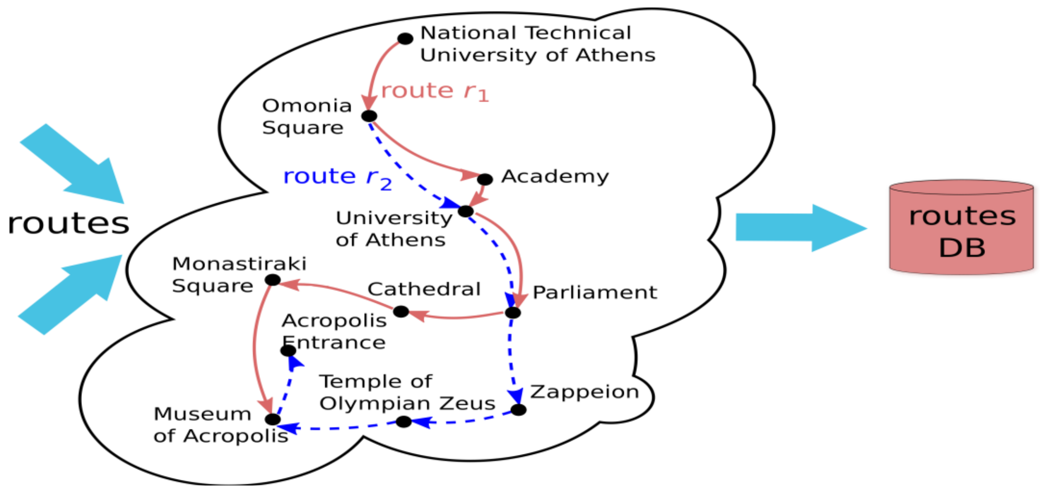
Data mining is the process of analyzing data from different perspectives and summarizing it into useful information. The data mining algorithms need to process large amounts of data, the desired patterns has to be found under acceptable computational efficiency limitations. The main goal of data mining is to discover new patterns for the users and to interpret the data patterns to provide meaningful and useful information for the users. Data mining has widely use in various do mains such as medical, healthcare, higher education, telecommunication etc. Databases today can range in size into the terabytes more than 1,000,000,000,000 bytes of data. Within these masses of data lies hidden information of strategic importance. But when there are so many trees, how do you draw meaningful conclusions about the forest? The newest answer is data mining, which is being used both to increase revenues and to reduce costs.

The potential returns are enormous. Innovative organizations worldwide are already using data Mining to locate and appeal to higher-value customers, to reconfigure their product offerings to Increase sales, and to minimize losses due to error or fraud. Data mining is a process that uses a variety of data analysis tools to discover patterns and Relationships in data that may be used to make valid predictions. The first and simplest analytical step in data mining is to describe the data summarize its statistical attributes (such as means and standard deviations), visually review it using charts and graphs, and look for potentially meaningful links among variables (such as values that often occur together). As emphasized in the section on the data mining process, collecting, exploring and selecting the right data are critically important. But data description alone cannot provide an action plan. The must build a predictive model based on patterns determined from known results, then test that model on results outside the original samples.

1.1 OVERVIEW OF ROUTE COLLECTION

Updating Route Collections

The case when new routes are added in the collection, while addresses deletions. The all index structures are stored as inverted file on secondary storage. To handle frequent updates, we perform lazy updates, deferring propagation of changes to the disk by maintain additional information in main memory. Then, at some time, a batch update process reflects all changes to the disk resident indices. Insertions are handled by merging memory-resident information with disk-based indices, while deletions require rebuilding of the affected lists.



Routes of Database

THE LINK TRAVERSAL SEARCH PARADIGM

Although the algorithms of Section 3 perform fewer iterations than conventional depth-first search on the route collection graph GR, they share three shortcomings. First, they perform redundant iterations by visiting non-links. To understand this, consider that the current search node is not a link and belongs to a single route. Further, assume that the algorithm has visited which is the link immediately before. Observe that if the termination condition does not hold at then it neither holds. To make matters worse, retrieving routes is pointless as it contains a single route in which all nodes after are already in the stack.

The second shortcoming is that the termination check is expensive. For current search node, recall that both RTS and RTST retrieve lists routes and routes from R-Index, while RTST additionally retrieves all lists transform T -Index for each included in routes. This cost is amplified by the number of iterations, as the algorithms perform the check for every node popped. The final shortcoming is due to the traversal policy. For each route that the current search node belongs to, the algorithms insert into the stack route subsequences that contain a very large number of nodes. This increases the space requirements of Q (and consequently of sets H, A). More importantly, however, some of these nodes may never be visited, which results to redundant I/Os incurred to retrieve them.

A good model should never be confused with reality (you know a road map isn't a perfect representation of the actual road), but it can be a useful guide to understanding your business.

The final step is to empirically verify the model. For example, from a database of customers who have already responded to a particular offer, you've built a model predicting which prospects are likeliest to respond to the same offer.

2. LITERATURE SURVEY

P.Bouros, S.Skiadopoulos, T.Dalamagas, D.Sacharidis, and T.K.Sellis. The propose a novel framework, called Mobile Commerce Explorer (MCE), for mining and prediction of mobile users' movements and purchase transactions under the context of mobile commerce. To our best knowledge, this is the first work that facilitates mining and prediction of mobile users' commerce behaviors in order to recommend stores and items previously unknown to a user. The perform an extensive experimental evaluation by simulation and show that our proposals produce excellent results. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein Searching temporal patterns on personal histories that have hundreds or thousands of events with tens of thousands of histories in a database can take a long time. Our experience in building a query interface extension for Amalgam revealed some performance problems using SQL. A temporal pattern query in SQL is not feasible for the hospital's database of thousands of patients because of prohibitively high number of self-join operations. Only after building additional indices and preprocessing (which it can take hours) could a temporal pattern query be managed Even so, the running time increases exponentially with the number of elements in the pattern. J. Cheng, J. X. Yu, X. Lin, H.Wang, and P. S. Yu To consider path queries on frequently updated route collections: given a route collection and two points ns and nt, a path query returns a path, i.e., a sequence of points, that connects ns to nt. We introduce two path query evaluation paradigms that enjoy the benefits of search algorithms (i.e., fast index maintenance) while utilizing transitivity information to terminate the search sooner. Efficient indexing schemes and appropriate updating procedures are introduced. An extensive experimental evaluation verifies the advantages of our methods compared to conventional graph-based search.

3. ALGORITHM

FILTER ALGORITHM

```
Input: D (F0, F1... Fn−1) // a training data set with N features

S0 // a subset from which to start the search

δ // a stopping criterion

Output: Sbest // an optimal subset

step1: begin

step2: initialize: Sbest = S0;

step3: γbest = eval (S0, D, M); // evaluate S0 by an independent measure M

step4: do begin

step5: S = generate (D); // generate a subset for evaluation

step6: γ = eval(S, D, M); // evaluate the current subset S by M

step7: if (γ is better than γbest)

step8: γbest = γ;

step9: Sbest = S;

step10: end until (δ is reached);

step11: return Sbest;

step12: end;
```

4. EXPERIMENTAL RESULT

This section presents a detailed study of all algorithms introduced. This Section details the setting, while evaluate index construction, querying and index maintenance, respectively, of all methods.

EXPERIMENTAL SETUP

The route traversal methods, RTS and RTST, and the link traversal algorithms, LTS, LTST and LTS-k. To gauge performance we compare against conventional depth-first search (DFS) on the reduced routes graph GR. All algorithms are written in C++ and compiled with the evaluation is performed on a 3 GHz Intel Core 2 Duo CPU with 4GB RAM running Debian Linux. We generate synthetic route collections varying the following parameters:

(1) The number of routes in the collection, |R|,

(2) The route length,

(3) The number of distinct nodes in the routes, |N|, and

(4) The links/nodes ratio. In each experiment, we vary one of the parameters while we keep the others to their default values.

EVALUATING PATH QUERIES

The efficiency of the proposed methods for processing PATH queries. All reported values are the averages taken by posing 5,000 distinct queries. Note that in Sections all considered queries have an answer, i.e., a path exists; the case of queries with no answer is investigated in the Section. Route vs link traversal search. The route traversal search methods RTS and RTST against the basic link traversal search algorithm LTS in terms of the execution time, while varying |R|, |N| and in respectively.

Varying the number of routes |R|. As |R| increases, finding a path between two nodes becomes easier. This is exhibited by RTST and LTS. In contrast, the execution time of RTS increases with |R| as it performs more iteration compared to RTST, which has a stronger termination condition, and to LTS, which only visits links.

Varying the route length The same observations hold when the route length increases. The performance of RTS deteriorates faster, since, in addition to requiring more iteration, each iteration costs more, as RTS inserts in the stack longer subsequences of routes.

Varying the number of nodes |N|. When |N| increases, finding a path becomes harder. The advantage of RTST over RTS decreases with |N|, because the benefit of a stronger termination condition diminishes as the total execution time is dominated by the number of iterations required. The advantage of LTS over RTS decreases because the benefit of traversing the links diminishes as each link is contained in fewer routes. Note that even for large |N|, not examined in This experiments set, RTS can never outperform LTS as they employ the same termination condition and RTS will always need more iterations than LTS. The same argument carries to RTST compared to LTST.

5. CONCLUSION AND FUTURE SCOPE

The problem of evaluating path queries on large disk-resident routes collections that are frequently updated. It introduced two generic search based paradigms, route traversal search and link traversal search, that exploit local transitivity information to expedite path query evaluation. The involved index structures and their maintenance strategies are designed to cope with frequent updates

The first time to define and solve the problem of multi-keyword ranked search over encrypted cloud data, and establish a variety of privacy requirements. Among various multi-keyword semantics, we choose the efficient principle of "coordinate matching", i.e., as many matches as possible, to effectively capture similarity between query keywords and outsourced documents, and use "inner product similarity" to quantitatively formalize such a principle for similarity measurement. For meeting the challenge of supporting multi-keyword semantic without privacy breaches, first propose a basic MRSE scheme using secure inner product computation, and significantly improve it to achieve privacy requirements in two levels of threat models. Thorough analysis investigating privacy and efficiency guarantees of proposed schemes is given, and experiments on the real-world dataset show our proposed schemes introduce low overhead on both computation and communication.

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-hop labels,”** in SODA, 2002, pp. 937–946.

(3) R. Schenkel, A. Theobald, and G. Weikum, “**Hopi: An efficient connection index for complex xml document collections,”**inEDBT, 2004, pp. 237–255.

(4) **“Efficient creation and incremental maintenance of the hopi index for complex xml document collections,”** in ICDE, 2005, pp.360–371.

(5) J. Cheng, J. X. Yu, X. Lin, H.Wang, and P. S. Yu, **“Fast computation of reachability labeling for large graphs,”** in EDBT, 2006, pp. 961–979.

(6) **“Fast computing reachability labelings for large graphs with high compression rate,”** in EDBT, 2008, pp. 193–204.

(7) R. Bramandia, B. Choi, and W. K. Ng, “On **incremental maintenance of 2-hop labeling of graphs,”** in WWW, 2008, pp. 845–854.

(8) R. Jin, Y. Xiang, N. Ruan, and D. Fuhry, “**3-hop: a high compression indexing scheme for reachability query,”** in SIGMODConference, 2009, pp. 813–826.