IJCSN International Journal of Computer Science and Network, Volume 5, Issue 6, December 2016
ISSN (Online) : 2277-5420 www.IJCSN.org
Impact Factor: 1.02

Towards Online Shortest Path Using LTI

[1] **Neha Makariye,** [2] **Deepa Deshpande**

[1,2] Dr.B.A.M.University, Aurangabad

Abstract - Towards online shortest path problem aims at computing the shortest path based on live traffic circumstances and recommends the best path. This is very important in modern car navigation systems as it helps drivers to make sensible decisions. To our best knowledge, there is no efficient system/solution that can offer affordable costs at both client and server sides for online shortest path computation. Unfortunately, the conventional client-server architecture scales poorly with the number of clients. A promising approach is to let the server collect live traffic information and then broadcast them over radio or wireless network. This approach has excellent scalability with the number of clients. Thus, it develop a new framework called live traffic index (LTI) which enables drivers to quickly and effectively collect the live traffic information on the broadcasting channel. An impressive result is that the driver can compute/update their shortest path result by receiving only a small fraction of the index. Our experimental study shows that LTI is robust to various parameters and it offers relatively short tune-in cost (at client side), fast query response time (at client side), small broadcast size (at server side), and light maintenance time (at server side) for online shortest path problem.

Keywords - *Broadcasting, LTI, Shortest Path.*

1. Introduction

Computing the shortest path is important task in the spatial databases. The path computed using the pre-stored data is not accurate. Hence, there is necessity for the live traffic data. There are several online service traffic providers like Navteq[1],Tom tom[2],Google maps[3].But these traffic providers does not provide data continuously due to high costs. Client-server architecture is previously used for the shortest path retrievals where the client sends the request and server responds to it. This architecture scales poorly if there are more than two clients. According to telecommunication expert[4],In 2015 the capacity provided by the cellular networks should increase 100 times than in 2011.The communication costs spent on retrieving the shortest route is high Malviya et al[5] used the client server architecture for shortest routes. In this system, the server stores the estimated paths between source and destination and updates them periodically and returns the shortest path to the required user. As the system employs client server architecture, it does not work for more number of users. In addition, the shortest route is not accurate. Hence there is a need to broadcast the traffic data.In raw transmission model, the traffic data is transmitted over the wireless network. The navigation system receives the live traffic data from the broadcast channel and calculates the shortest path .The broadcast channel transmits the updated packets to the navigation system for each broadcasted cycle. The main disadvantage in this model is client receives more number of traffic updates than required. A new model known as index transmission model [6] is Introduced where the index is transmitted over the broadcast channel. The advantage in this model is client receives only the required portion of the road network. In [6], the time required for the index re-computation is 2 hours for San Franciso road network.it requires huge costs to update the index. Hence, in our paper a technique live traffic index is added to the index transmission model.

2. Related Work

In this paper, the important factors of OSP are considered (i) tune-in cost (ii) broadcast size (iii) maintenance time (iv)query response time. The tune in cost should be reduced because the power consumption is based on tune in cost. Due to short tune in cost, by using selective tuning [7] the client receives more services like parking slot, weather related updates. The maintenance time is defined as the time taken to update the index based on the live traffic data. The response time should be very fast in few milliseconds. The broadcast size is similar to the discontinuation of receiving the latest index data. The traffic data received by the client is maintained and shortest path is computed using the uninformed search where the graph nodes are arranged in the ascending order of the distances from the source and the shortest path to the destination is discovered. Bi-directional search [8] can

be employed where the Dijkstra's algorithm is executed from forward and backward. But the response time is high and client receives more number of updates. Goal directed methods eliminate the edges that is not required to the shortest path, goal directed methods include ALT [9] and arc Flags [10].In ALT, the distance between the nodes are pre computed using A* search, Landmarks. Delling and Wagner [11] introduced a paradigm for ALT [DALT] so that the edge weights can be updated in dynamic graphs. The maintenance cost is reduced. A tree structure is maintained for shortest path in Dynamic shortest path tree. In index transmission model, the traffic broadcast server transmits the live index not the complete data various Road map hierarchical methods like reach [12], highway hierarchies (HH), contraction hierarchies (CH) [13], and Transit node routing (TNR) [15].In HH, CH, TNR the query response time is fast because the shortcuts are calculated and stored in the index. In TNR, the shortest path is computed based on the two transit nodes A(S) from source and A(t) from destination. The maintenance time is high because there is no efficient method to update the pre computed data. The index structure is maintained in hierarchical way in hierarchical index structures. In Hierarchical multi graph model (HiTi) [16] the index is maintained in hierarchy not the road network. But the high maintenance time and broadcast size are the disadvantages. In Hierarchical encoded path view [HEPV], the nodes and edges are maintained in large graph which is split into smaller graphs for shortest path computation. But the index storage is a big problem.

To overcome the flaws in the existing system it was necessary to develop the proposed system. Our proposed system live traffic index supports all the factors like short tune in cost, less maintenance time, small broadcast size, and quick query response time. To summarize, our work makes the following contributions: 1) The traffic provider collects the traffic data of road networks via road sensors and broadcasts to the traffic broadcast server. 2) The traffic broadcast server collects the traffic data and index structure is constructed for the traffic data. The (1, m) broadcasting scheme is used for transmitting. 3) The index structure is optimized by graph partitioning technique and stochastic process is applied. The index is organized by using Dynamic Shortest Path Tree (DSPT) [17] and broadcast size is also reduced. 4) The navigation client after receiving the required index from the Traffic broadcast server, the shortest path is computed using the dijkstra's algorithm. 5) Live Traffic index reduces the tune in cost and broadcast size and low maintenance time, fast query response time is achieved by the above features.

3. Proposed System

A road network monitoring system typically consists of a service provider, a large number of mobile clients and a traffic provider shows an architectural overview of this system in the context of our live traffic index framework. The traffic provider collects the live traffic circumstances from the traffic monitors via techniques like road sensors and traffic video analysis. The service provider periodically receives live traffic updates from the traffic provider and broadcasts the live traffic index on radio or wireless network. When a mobile client wishes to compute and monitor a shortest path, it listens to the live traffic index and reads the relevant portion of the index for computing the shortest path. In this work, this paper focus on handling traffic updates but not graph structure updates. For real road networks, it is infrequent to have graph structure updates when compared to edge weight updates.

3.1 System Architecture

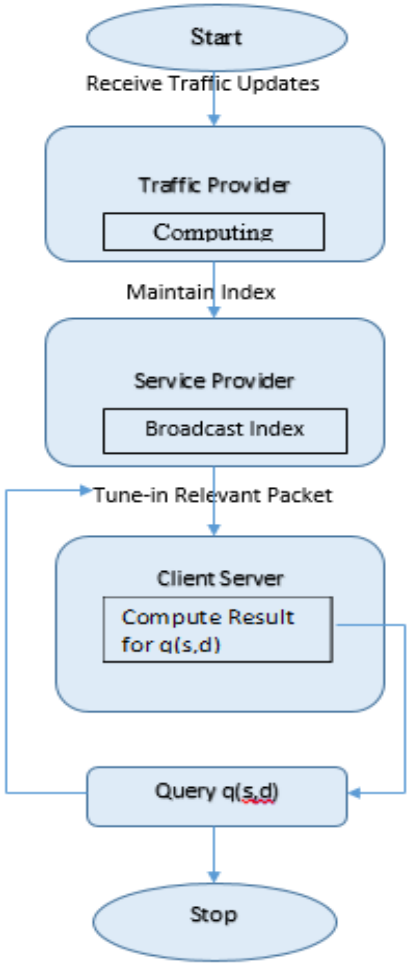

Fig. 1 System Architecture

3.2 Module Description

3.2.1. Real-time Traffic Simulator

The evaluation of spatiotemporal database systems or their components requires the definition of suitable benchmarks simulating the typical behavior of moving objects. While dealing with spatiotemporal data, the objects follow a given network and Traffic simulation involves a huge number of bottlenecks, because of the need to incorporate a numerous real-time factors such as number of objects, different classes of objects, speed of the moving objects, and location of the objects. Therefore, benchmarks require datasets with such "network-based" moving objects.

3.2.2. Traffic Provider

Real-time traffic feeds are collected from the traffic monitoring resources planted along the roadside and other crowd-sourcing techniques by the provider. The collected information is shared to different service provider both as raw and optimized data. This paper provides the live feed to the service providers which are collected from the advanced real-time traffic simulator. This channel provides a preprocessed traffic data to the Service provider.

3.2.3. Service Provider

The preprocessed raw traffic data collected from the providers are being indexed and are orderly broadcasted towards multiple clients. The traffic datum are organized by indexing the collected information such as to provide real-time feeds to the clients, and also in the manner to ease the client with their computation cost.

3.2.4. Navigation Client

The clients which are in need of the real-time traffic feeds are tuned in a manner to receive the relevant data packets based on the indexed traffic data. The received packets are then used for computing the shortest path query, *q(s,d).* Generally this computation part is carried over in the server but here it does it in the client part to reduce computation overload for the server and to reduce significant computation cost.

3.3 Algorithm Use

All the above mentioned techniques are put together for computing the shortest path.There are three algorithms,one algorithm isrun to construct the LTI and other algorithm is run at the client and the third is run at the server. The first step is to construct the index.In service algorithm,the service provider constructs the index.First the live traffic data is put in the graph structure.The graph is partitioned into smaller graphs namely SG1,SG2……and also the graphpartitioning algorithm is applied such that the index is optimized. The service provider collects the live traffic update periodically and updates the subgraphs. The client executes the client algorithm. Based on the source and destination, the client generates the search graphs. Subsequently, the client listens to the header segments. Based on these segments, it updates the search graph and the shortest path is computed on the search graph.

3.3.1. Stochastic Partitioning Algorithm

Partition(G:the graph, y:the number of partitions)
1: (z,v):= G and n:=root of I
2: insert (n,G,v,z) into PQ in decreasing order to z
3: while |PQ| < y do
4: (n,G,v,z):=PQ.pop()
5:for k: =2 to y-|PQ|+1 do
6:decompose G into SG1……..
7:form a temporal index I' that joins SG1
8:if avg(S(I')) is better than best s then
9: update best s and best SG:={ SG1……}
10: join the best SG as n's children
11: for i: =1 to |best SG| do
12: insert (n,SG,v,z) into PQ
13: return

To minimize the overhead of pre-computed information, the paperstudy a graph partitioning optimization that minimizes the index overhead SGi through the entire index construction subject to a leaf entry constraint. Subsequently, this paper propose a stochastic process to optimize the index structure such that the size of the query search graph Gq is minimized. At every partitioning, this attempt to find the best structure for the potential queries by the stochastic process.The effect of g. In this work, LTI requires only one parameter g to construct the index which is used to control the number of sub-graphs being constructed [20]. Our proposed techniques attempt to optimize the index subject to g. intuitively, similar to other hierarchical indices, the number of leaf entries, g, not only affects the size of leaf entries but also the search performance.

3.3.2. Client Algorithm

Algorithm

Client (I:LTI,s:source,t:destination)
1: generate G^q from I based on s and d
2: listen to the channel for a header segment
3: read the header segment
4: decide the necessary segments to be read

5: wait for those segments, read them to update the weight of G^q
6: compute the shortest path (from s to t) on G^q

Present our complete LTI framework, which integrates all techniques been discussed. A client can invoke Algorithm 2 in order to find the shortest path from a source s to a destination t. First, the client generates a search graph Gq based on s (i.e., current location) and d. When the client tunes-in the broadcast channel, it keeps listening until it discovers a header segment .After reading the header segment, it decides the necessary segments (to be read) for computing the shortest path. These issues are addressed .The client then wait for those segments, read them, and update the weight of Gq. Subsequently, Gq is used to compute the shortest path in the client machine locally [20].

3.3.3. Service Algorithm

Algorithm Service (G: graph)
1: construct I and {SGi} based on G
2: for each broadcast cycle do
3: collect traffic update from the traffic provider
4: update the subgraphs {SGi}
5: broadcast the subgraphs {SGi}

The first step is devoted to construct the live traffic index; they are offline tasks to be executed once only. The service provider builds the live traffic index by partitioning the graph G into a set of subgraphs fSGig such that they are ready for broadcasting. The paper develop an effective graph partitioning algorithm for minimizing the total size of subgraphs and study a combinatorial optimization for reducing the search space of shortest path queries.In each broadcasting cycle, the server first collects live traffic updates from the traffic provider, updates the subgraphs fSGig, and eventually broadcasts them[20].

3.3.4. Dijkstra's Algorithm

Dijkstra's Algorithm is to find the shortest path between two intersections on a city map, a starting point and a destination. The order is conceptually simple: to start, mark the distance to every intersection on the map with infinity. This is done not to imply there is an infinite distance, but to note that intersection has not yet been visited; some variants of this method simply leave the intersection unlabeled [20]. Now, at each iteration, select a current intersection. For the first iteration, the current intersection will be the starting point and the distance to it (the intersection's label) will be zero. For subsequent iterations (after the first), the current intersection will be the closest unvisited intersection to the starting point—this will be easy to find. Continue this process of updating the neighboring intersections with the shortest distances and can trace your way back, following the arrows in reverse [20].

4. Experimental Result

This paper solve the scalability and the accuracy problems by optimizing three key factors that affect the performance of distance oracles, namely landmark selection, distributed BFS, and distance estimation.Our experiment results on both real networks and synthetic networks show that our approach is practical for billion-node graphs, and it beats existing approaches in terms of scalability and accuracy on million-node or smaller graphs.

Performance Factor

1. Response time(ms):

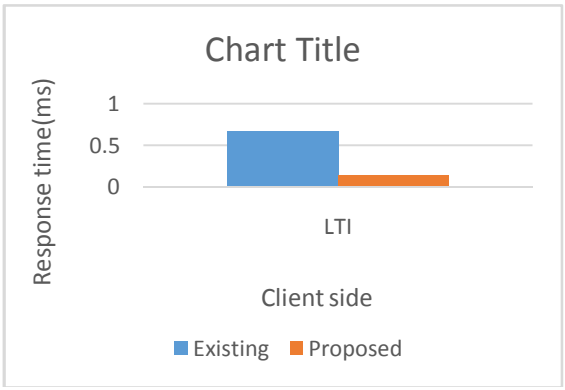

Fig. 2 Shows response time

The response time is less as compare to the existing system as shown in the above analysis.

2. MaxPackets:

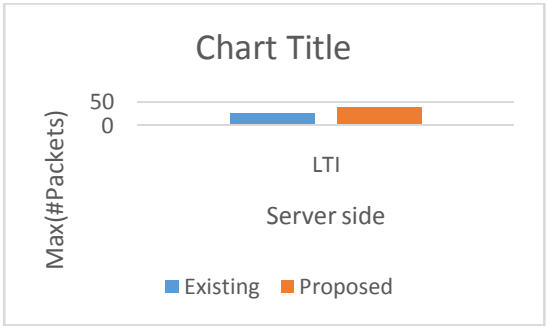

Fig.3 Shows max packets

The maximum packets send at the server side in timeline is more as compare to existing system.The system is tested

against different data set response times and max packetsize is the resultant results are graphed for analysis. The analysis has rendered a great support for the system, I could collect good number of evidences for the system.

3. Live-Historical Traffic Condition

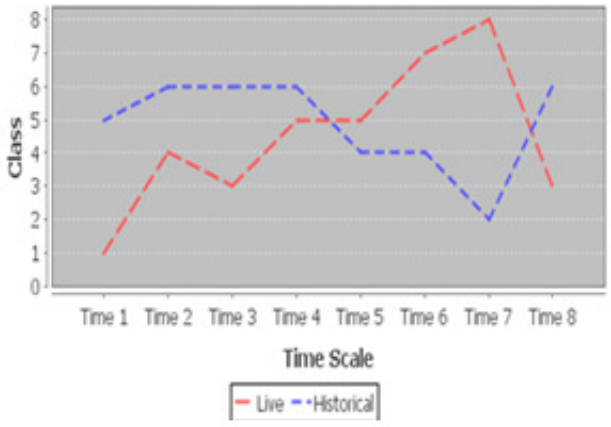

Fig.3 Shows Live- historical traffic conditions

The experimental results shows that using livetraffic index the shortest path is computed between nodes.The shortest path is computed very fast because the traffic data is broadcasted in the index format, a graph is plot on the live traffic data and historical traffic data. By gathering the live traffic data there are many changes fluctuations in the graph whereas by using the historic data there are no fluctuations.

5. Conclusion

In this paper studies online shortest path computation; the shortest path result is computed/updated based on the live traffic circumstances. It carefully analyze the existing work and discuss their inapplicability to the problem. To address the problem, it suggest a promising architecture that broadcasts the index on the air. This first identify an important feature of the hierarchical index structure which enables us to compute shortest path on a small portion of index. This important feature is thoroughly used in our solution, LTI.

References

[1] "Navteq maps and traffic," http://www.navteq.com,2014
[2] "TomTom NV," http://www.tomtom.com, 2014.
[3] "Google Maps," http://maps.google.com, 2014.
[4] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2010-2015," 2011.
[5] N.Malviya,S.Madden, and A. Bhattacharya, "A Continuous Query System for Dynamic Route Planning," Proc. IEEE 27th Int'lConf Data Eng. (ICDE), pp. 792-803, 2011.
[6] G. Kellaris and K. Mouratidis, "Shortest Path Computation on Air Indexes," Proc. VLDB Endowment, vol. 3, no. 1, pp. 741-757, 2010.
[7] J.X. Yu and K.-L. Tan, "An Analysis of Selective Tuning Schemes for Nonuniform Broadcast," Data and Knowledge Eng., vol. 22, no. 3, pp. 319-344, 1997.
[8] G.Dantzig, Linear Programming and Extensions, series Rand Corporation Research Study Princeton Univ.Press, 1963.
[9] A.V. Goldberg and R.F.F. Werneck, "Computing Point-to-Point Shortest Paths from External Memory,"Proc. SIAM Workshop Algorithms Eng. and Experimentation and the Workshop Analytic Algorithmics and Combinatorics(ALENEX/ANALCO), pp. 26-40, 2005.
[10] M. Hilger, E. K€ohler, R. M€ohring, and H. Schilling,"Fast Point-to- Point Shortest Path Computations with Arc-Flags," The Shortest Path Problem: Ninth DIMACS Implementation Challenge, vol. 74,pp. 41- 72, American Math. Soc., 2009.
[11] D. Delling and D. Wagner, "Landmark-Based Routing in Dynamic Graphs," Proc. Sixth Int'l Workshop Experimental Algorithms (WEA),pp. 52-65, 2007. [12] R.J. Gutman, "Reach-Based Routing: A New Approach to Shortest Path Algorithms Optimized for Road Networks," Proc. Sixth Workshop Algorithm Eng. and Experiments and the First Workshop Analytic Algorithmics and Combinatorics (ALENEX/ANALC),pp.100-111, 2004.
[13] B. Jiang, "I/O-Efficiency of Shortest Path Algorithms:An Analysis,"Proc. Shortest Path Queries," Proc. 13th Ann.European Conf.
[14] P. Sanders and D. Schultes, "Highway Hierarchies Hasten Exact Shortest Path Queries," Proc. 13th Ann.European Conf. Algorithms(ESA), pp. 568-579, 2005.
[15] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, "Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks," Proc. Seventh Int'l Workshop Experimental Algorithms (WEA),pp. 319-333,2008.
[16] 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. 2002.
[17] E.P.F. Chan and Y. Yang, "Shortest Path Tree Computation in Dynamic Graphs," IEEE Trans. Computers, vol. 58, no. 4, pp. 541-557, Apr. 2009.
[18] T. Beuhler and M. Hein, "Spectral Clustering Based on The Graph – Laplacian," Proc. Int'l Conf. Machine Learning (ICML), p. 11, 2009.
[19] T. Imielinski, S. Viswanathan, and B.R.Badrinath, "Data on Air:Organization and Access," IEEE

Trans.Knowledge and Data Eng.,vol. 9, no. 3, pp. 353-372.

[20] IEEE Transactions on Knowledge and Data Engineering, Vol. 26, No. 4, April 2014,"Towards Online Shortest Path Computation" Leong Hou U, Hong Jun Zhao, Man Lung Yiu, Yuhong Li, and Zhiguo Gong.