International Journal of Research
Available at http://internationaljournalofresearch.org/
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 01 Issue 09
October 2015

A Novel Approach to Implement Online Shortest Path Computation by Using Live Traffic Index (LTI)

Kalyani Muntha[1], N Venkateswara Rao [2]

[1]PG Scholar,Dept of CSE, Rao & Naidu Engineering College, Ongole,Prakasam Dist,Andhra Pradesh
[2] Associate Professor,Dept of CSE, Rao & Naidu Engineering College, Ongole,Prakasam Dist,Andhra Pradesh

Abstract:

Nowadays, several online services provide live traffic data such as Google-Map, Navteq, INRIX Traffic Information Provider, and TomTom NV. But still computing the shortest path on live traffic is big problem. This is important for car navigation as it helps drivers to make decisions. In presented approach server will collect live traffic information and then announce them over wireless network. With this approach any number of clients can be added. This new approach called live traffic index-time dependant (LTI-TD) enables drivers to update their shortest path result by receiving only a small fraction of the index. The existing systems were infeasible to solve the problem due to their prohibitive maintenance time and large transmission overhead. LTI-TD is a novel solution for Online Shortest Path Computation on Time Dependent Network.

Keywords: LTI-TD, Shortest Path, Transmission Overhead.

I Introduction:

With the ever-growing popularity of onlinemap applications and their wide deployment in mobile devices and car-navigation systems, an increasing number of users search for point-to-point fastest paths and the corresponding travel-times. On static road networks where edge costs are constant, this problem has been extensively studied and manyefficient speed-up techniques have been developed to compute the fastest path in a matter of milliseconds. The static fastest path approaches make the simplifying assumption that the travel-time for each edge of the road network is constant (e.g., proportional to the length of the edge). However, in real-world the actual travel-time on a road segment heavily depends

on the traffic congestion and, therefore, is a function of time i.e., time-dependent. For example, Figure 1 shows the variation of travel-time (computed by averaging two-years of historical traffic sensor data) for aparticular road segment of I-10 freeway in Los Angeles as a function of arrival-time to the segment. As shown, the travel-time changes with time (i.e, the time that one

arrives at the segment entry determines the travel-time), and the change in travel-time is significant. For instance, from 8AM to 9AM the travel-time of the segment changes from 32 minutes to 18 minutes (a 45% decrease). By induction, one can observe that the time-dependent edge travel-times yield a considerable change in the actual fastest path between any pair of nodes throughout the day. Specifically, the fastest between a source and a destination node varies depending on the departure-time from the source. Unfortunately, all those techniques that assume constant edge weights fail to address the fastest path computation in real-world time-dependent spatial networks.

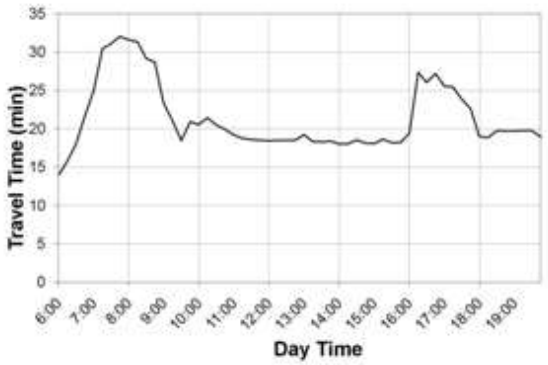


Fig 1:Analysis between Day Time/Travel Time

The time-dependent fastest path problem was first shown by Dreyfus to be polynomially solvable in FIFO networks by a trivial modification to Dijkstra algorithm where, analogous to shortest path distances, the arrival-time to the nodes is used as the labels that form the basis of the greedy algorithm. The FIFO property, which typically holds for many networks including road networks, suggests that moving objects exit from an edge in the same order they entered the edge1.

However, the modified Dijkstra algorithm is far too slow for onlinemap applicationswhich are usually deployed on very large networks and require almost instant response times. On the other hand, there are many efficient precomputation approaches that answer fastest path queries in near real-time in static road networks. However, it is infeasible to extend these approaches to time-dependent networks.This is because the input size (i.e., the number of fastest paths) increases drastically in time-dependent networks. Specifically, since the length of a s-d path changes depending on the departure-time from s, the fastest path is not unique for any pair of nodes in time-dependent networks. It has been conjectured in and settled in that the number of fastest paths between any pair of nodes in time-dependent road networks can be super-polynomial. Hence, an algorithm which considers the every possible path (corresponding to every possible departure-time from the source) for any pair of nodes in large time-dependent networks would suffer from exponential time and prohibitively large storage requirements. For example, the timedependent extension of Contraction Hierarchies (CH) and SHARC speed-up techniques (which are proved to be very efficient for static networks) suffer from the impractical precomputation times and intolerable storage complexity . In this study, we propose a bidirectional time-dependent fastest path algorithm (BTDFP)based on A* search .

There are two main challenges to employ bidirectional A* search in time-dependent networks. First, finding an admissible heuristic function (i.e., lower-bound distance) between an intermediate vi node and the destination d is challenging as the distance between vi and d changes based on

the departure-time fromvi. Second, it is not possible to implement a backward search without knowing the arrival-time at the destination.We address the former challenge by partitioning the road network to non-overlapping partitions (an off-line operation) and precompute the intra (node-to-border) and inter (border-to-border) partition distance labels with respect to Lower-bound Graph G which is generated by substituting the edge travel-times in G with minimum possible travel-times.We use the combination of intra and inter distance labels as a heuristic function in the online computation.To address the latter challenge, we run the backward search on the lower-bound graph (G) which enables us to filter-in the set of the nodes that needs to be explored by the forward search. The remainder of this paper is organized as follows. In Section 2, we explain the importance of time-dependency for accurate and useful path planning.

In Section 3, we review the related work on time-dependent fastest path algorithms. In Section 4, we formally define the time-dependent fastest path problem in spatial networks. In Section 5, we establish the theoretical foundation of our proposed bidirectional algorithm and explain our approach. In Section 6, we present the results of our experiments for both approacheswith a variety of spatial networks with real-world time-dependent edge weights.

II. LITERATURE SURVEY

A. Spectral Clustering Based on the Graph Laplacian

A connection between the Cheeger cut and the second eigenvector of the graph p-Laplacian, a nonlinear generalization of the graph Laplacian. A p-Laplacian which is slightly from the one used has been used for semi supervised learning .The main motivation for the use of eigenvectors of the graph p-Laplacian was the generalized isoperimetric inequality. In which relates the second eigenvalue of the graph p-Laplacian to the optimal Cheeger cut. The isoperimetric inequality becomes tight as p, so that the second Eigen value converges to the optimal Cheeger cut value.

B. SHARC: Fast and Robust Unidirectional Routing Introduce SHARC-Routing, a fast and robust approach for unidirectional routing in large networks. The central idea of SHARC (Shortcuts + Arc-Flags) is the adaptation of techniques developed for Highway Hierarchies to Arc Flags. In general, SHARC-Routing iteratively constructs a contraction-based hierarchy during preprocessing and automatically sets arc- ags for edges removed during contraction. More precisely, arc- ags are set in such a way that a unidirectional query considers these removed component-edges only at the beginning and the end of a query as a result, able to route very efficiently in scenarios where other techniques fail due to their bidirectional nature. It turned out that SHARC was a promising candidate for routing in time-dependent networks.

C. Computing point to point shortest path from External Memory The ALT algorithm for the point-to-point shortest path problem in the context of road networks the suggest improvements to the algorithm itself and to its preprocessing stage. Also develop a memory-efficient implementation of the algorithm that runs on a Pocket PC (Personal Computer). It stores graph data in a ash memory card and uses RAM (Random Access Memory) to store information only for the part of the graph visited by the

current shortest path computation. The implementation works even on very large graphs, including that of the North America road network, with almost 30 million vertices.

D. Time-Dependent SHARC-Routing During the last years, many speed-up techniques for Dijkstra"s algorithm have been developed. As a result, computing a shortest path in a static road network is a matter of microseconds. However, only few of those techniques work in time-dependent networks. Unfortunately, such networks appear frequently in reality.

E. Shortest Path Tree Computation in Dynamic Graphs The Dynamic Shortest Path (DSP) problem is to compute S from D. This problem either focuses on a single edge weight change, or for multiple edge weight changes, some of them are incorrect or are not optimized. The correct and extend a few state-of-the-art dynamic SPT algorithms to handle multiple edge weight updates. Hence prove that these algorithms are correct dynamic algorithms may not out perform static algorithms all the time to evaluate the proposed dynamic algorithms, compare them with the well-known static Dijkstra"s algorithm.

III. LTI-TD FRAMEWORK

The broadcasting model uses transmission medium such as 3G, Mobile WiMAX. When the traffic provider broadcasts a dataset all drivers can listen to the dataset concurrently thus, this transmission model balances well independent of the number of driver. In the wireless broadcast model the traffic provider repeatedly transmits broadcast cycles, containing the database and air index. The broadcast cycle consists of fixed-size packets. The most common wireless broadcasting method is the (1, m) interleaving scheme, shown in Fig.3.

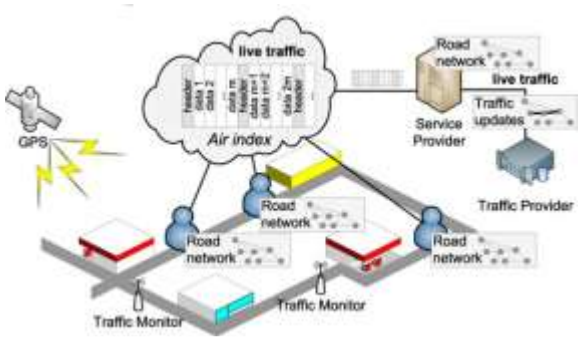


Fig.2. System architecture for LTI-TD.

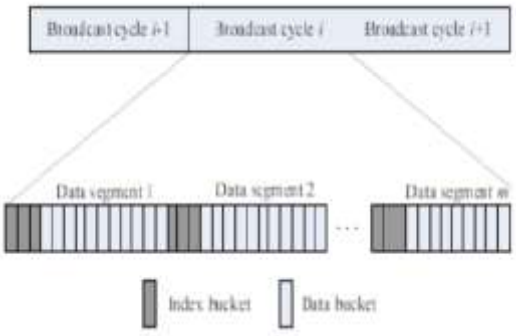


Fig.3. (1, m) interleaving scheme

The dataset is divided into m distinct segments, and each data segment is preceded by the index. This way the driver may receive a copy of the index immediately after the completion of the currently transmitted data segment. A driver can raise Algorithm 1 first in order to find the shortest path from a source to a destination after reading the necessary segment, it computes the shortest path. In each broadcasting cycle, the driver first collects live traffic updates from the traffic provider, and then updates the graphs. The ALT algorithm was proposed to find shortest path on road networks. With ALT, a set of nodes are chosen and then the shortest

International Journal of Research
Available at http://internationaljournalofresearch.org/
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 01 Issue 09
October 2015

path between all the nodes in the network are computed. The time-dependent ALT algorithm calculates the leaving time from a source to find the correct path. A driver can raise Algorithm 2 in order to find the shortest path from a source to a destination. First, the client generates a search graph G based on current position and destination. When the driver keeps listening to the broadcast channel until it discovers a necessary segment in order to keep the newness of LTI-TD, the system is required to broadcast the newest weight of edges alternating.

Algorithm ALT (graph G = (V, E), Vertices s and t):

1: L = generate Landmarks (G, k) {select set of k and mark}

2: for all v ∈ V do

3: parent (v) ← ⊥

4: state (v) ← unreached

5: dist(s, v) ← ∞

6: dist(s, s) ← 0

7: state(s) ← reached

8: while vertex v with state (v) = reached exists and state (t) 6= reached do

9: Select v ∈ V with state (v) = reached and minimal cost (v) = dist(s, v) + πLt (v)

10: for all u ∈ V with (v, u) ∈ E do

11: if dist(s, v) + len (v, u) + πLt (u) <dist(s, u) + πLt (u) then

12: parent (u) ← v

13: dist(s, u) ← dist(s, v) + len (v, u) 14: state (u) ← reached

15: state (v) ← settled

Algorithm driver(s: source; t: destination):

1: generate G based on s and d

2: listen to the channel for a segment

3: decide the necessary segments :

4: compute the shortest path (from s to t) on G.

Algorithm traffic-provider G: graph):

1: construct G.

2: for each broadcast cycle do

3: collect traffic updates from the traffic provider

4: update the graphs G.

5: broadcast the graph G

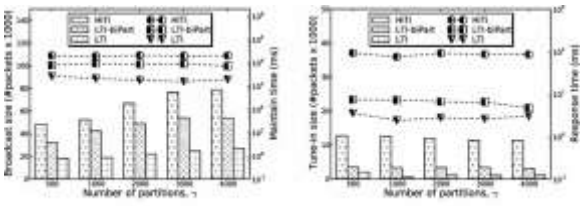
IV. EXPERIMENTAL EVALUATIONS

In this section, we empirically evaluate the performance ofsome representative algorithms using the broadcasting architecture; we ignore the client-server architecture due to massive live traffic in near future (see Section 1). From our discussion in Section 2, bi-directional search [3], ALT on 8.1 Effectiveness of Optimizations First, we evaluate the effectiveness of the optimizations proposed in Section 4. The fully optimized LTI is compared dynamic graph (DALT) , and dynamic shortest paths tree [, are applicable to raw transmission model. On the other hand, contraction hierarchies ,

International Journal of Research
Available at http://internationaljournalofresearch.org/
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 01 Issue 09
October 2015

Hierarchical MulTi-graph model , and our proposed live traffic index are applicable to index transmission model. We omit some methods (such as TNR [1], Quadtree , SHARC , and CALT) due to their prohibitive maintenance time and broadcast size. In the following, we first describe the road map data used in experiments and describe the simulation of clients' move-ments and live traffic circumstances on a road map.

Then, we study the performance of the above methods with respect to various factors. Map data. We test with four dif-ferent road maps, including New York City (NYC) (264k nodes, 733k edges), San Francisco bay area road map (SF) (174k nodes, 443k edges),San Joaquin road map (SJ) (18k nodes, 48k edges), and Oldenburg road map (OB) (6k nodes, 14k edges). All of them are available at [43] and [44].Simulation of clients and traffic updates. We run the networkbased generator [44] to generate the weight of edges. It initializes 100,000 cars (i.e., clients) and then generates 1,000 new cars in each iteration. It runs for 200 iterations in total, with the other generator parameters as their default values. The weight of an edge is set to the average driving time on it.



(a) service provider (b) client

Fig :4 Varying number of partitions, Ɣ

TABLE 1:Performance of Different Methods:

City		Method					
		raw transmission model			index transmission model		
		BD	DALT	DSPT	CH	HiTi	LTI
NYC	T	18300.7	18300.7	18300.7	18617.9	26834.5	704.7
	R	72.39	54.53	374.93	2.28	157.11	4.26
	B	22930	22930	22930	24802	124870	30661
	M	-	-	-	15759.6	105451	5575.4
SF	T	11149.4	11149.4	11149.4	10212.1	12468.9	602.8
	R	89.74	45.03	94.51	0.72	75.89	2.40
	B	13863	13863	13863	13453	52377	18850
	M	-	-	-	5411.4	19264.4	2094.9
SJ	T	1191.2	1191.2	1191.2	624.0	1524.1	331.1
	R	5.20	1.98	9.37	0.08	10.72	1.37
	B	2525	2525	2525	1370	4602	2827
	M	-	-	-	276.3	735.3	96.6
OB	T	352.0	352.0	352.0	258.9	516.3	118.4
	R	1.11	0.45	31.12	0.091	3.66	0.68
	B	604	604	604	348	1336	666
	M	-	-	-	104.3	134.6	16.0

against to LTI-biPart (that is constructed by only the graph partitioning technique, described in Section 4.2) and Hi (which is the most representative model of hierar-chical index structures). For fairness, we internally tune the HiTi graph model by varying the number of children subgraphs, and the eight-way regular partitioning is the best HiTi graph model among all testings. Fig. 12 plots the performance of all three methods as a function of the number of partitions g on the SF data set. For the sake of saving space, we plot the costs at service provider (i.e., broadcast size and maintenance time) into one figure and plot the costs at client (i.e., tune-in size and response time) into another figure. The number of packets(left y-axis) is represented by bars, whereas the time (right y-axis) is represented by lines. 8.2 Scalability Experiments Next, we compare the discussed solutions on four different road maps.

The result is shown in Table 1. Note that all methods on the raw transmission model have the same tune-in size and broadcast size. The only difference is the response time as it represents the local computation time for each client. Apart from BD and DALT, other methods require each client to maintain some index structures locally af-ter

receiving the live traffic updates. Thus, their response time is slower5 than BD and DALT on the raw transmis-sion model. Based on the responsetime, DALT is the best approach among the methods in his category.

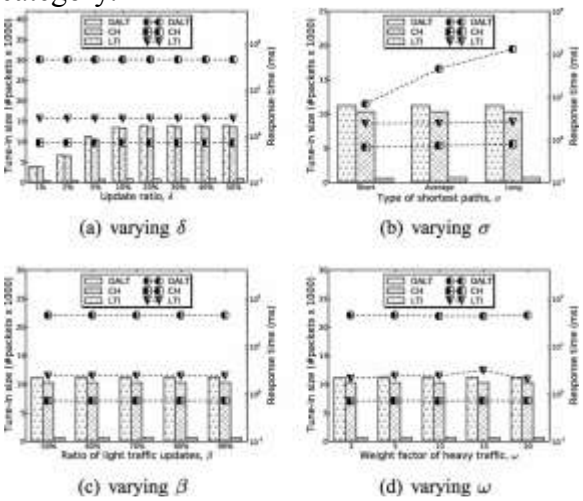


Fig 5 :Scalability experiments (client).

V .CONCLUSION:

In this paper we studied online shortest path computa-tion; the shortest path result is computed/updated based on the live traffic circumstances. We carefully analyze the existing work and discuss their inapplicability to the problem (due to their prohibitive maintenance time and large transmission overhead). To address the problem, we suggest a promising architecture that broadcasts the index on the air. We 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. Our ex-periments confirm that LTI is a Pareto optimal solution in terms of four performance factors for online shortest path computation. In the future, we will extend our solution on time dependent networks. This is a very interesting topic since the decision of a shortest path depends not only on current traffic data but also based on the predicted traffic circumstances.

V I.REFERENCES:

[1] H. Bast, S. Funke, D. Matijevic, P. Sanders, and D. Schultes, “InTransit to Constant Time Shortest-Path Que-ries in RoadNetworks,” Proc. Workshop Algorithm Eng. and Experiments(ALENEX), 2007.

[2] P. Sanders and D. Schultes, “Engineering Highway Hierarchies,”Proc. 14th Conf. Ann. European Symp. (ESA), pp. 804-816, 2006.

[3] G. Dantzig, Linear Programming and Extensions, se-ries Rand CorporationResearch Study Princeton Univ. Press, 1963.

[4] R.J. Gutman, “Reach-Based Routing: A New Ap-proach to ShortestPath Algorithms Optimized for Road Networks,” Proc. SixthWorkshop Algorithm Eng. and Experiments and the First WorkshopAnalytic Algorith-mics and Combinatorics (ALENEX/ANALC),

[5] B. Jiang, “I/O-Efficiency of Shortest Path Algorithms: An Analysis,”Proc. Eight Int’l Conf. Data Eng. (ICDE), pp. 12-19, 1992.

[6] P. Sanders and D. Schultes, “Highway Hierarchies Hasten ExactShortest Path Queries,” Proc. 13th Ann. Eu-ropean Conf. Algorithms(ESA), pp. 568-579, 2005.

[7] D. Schultes and P. Sanders, “Dynamic Highway-Node Routing,”Proc. Sixth Int’l Conf. Experimental Algorithms (WEA), pp. 66-79,2007.

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 01 Issue 09
October 2015

[8] F. Zhan and C. Noon, "Shortest Path Algorithms: An EvaluationUsing Real Road Networks," Transportation Science, vol. 32, no. 1,pp. 65-73, 1998.

[9] "Google Maps," http://maps.google.com, 2014.

[10] "NAVTEQ Maps and Traffic," http://www.navteq. com, 2014.

[11] "INRIX Inc. Traffic Information Provider," http:// www.inrix. com, 2014.

[12] "TomTom NV," http://www.tomtom.com, 2014.