ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Towards Online Shortest Path Computation

B.Prameela
M.Tech Student,
Dept of CSE,
KITS for Women's, Kodad, T.S, India.

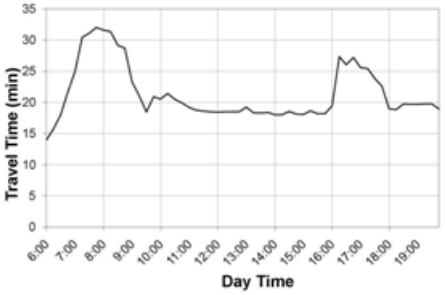
Mr.B.Ramesh
Associate Professor,
Dept of CSE,
KITS for Women's, Kodad, T.S, India.

Abstract:

The problem of point-to-point fastest path computation in static spatial networks is extensively studied with many precomputation techniques proposed to speed-up the computation. Most of the existing approaches make the simplifying assumption that travel-times of the network edges are constant. However, with real-world spatial networks the edge travel-times are time-dependent, where the arrival-time to an edge determines the actual travel-time on the edge. Inthis paper, we study the online computation of fastest path in time-dependent spatial networks and present a technique which speeds-up the path computation. We show that our fastest path computation based on a bidirectional time-dependent A* search significantly improves the computation time and storage complexity. With extensive experiments using real data-sets (including a variety of large spatial networks with real traffic data) we demonstrate the efficacy of our proposed techniques for online fastest path computation.

1 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.



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.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

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.

2 PRELIMINARY:

2.1 Performance Factors:

The main performance factors involved in OSP are: (i) tunein cost (at client side), (ii) broadcast size (at server side), and (iii) maintenance time (at server side), and (iv) query response time (at client side). In this work, we prioritize the tune-in cost as the main optimized factor since it affects the duration of client receivers into active mode and power consumption is essentially determined by the tuning cost (i.e., number of packets received) , In addition, shortening the duration of active mode enables the clients to receive more services simultaneously by selective tuning . These services may include providing live weather information, delivering latest promotions in surrounding area, and monitoring availability of parking slots at destination. If we minimize the tune-in cost of one service, then we reserve more resources for other services.

2.2 Adaptation of Existing Approaches:

In this section, we briefly discuss the applicability of the state-of-the-art shortest path solutions on different transmission models. As discussed in the introduction, the result transmission model scales poorly with respect to the number of clients. The communication cost is proportional to the number of clients (regardless of whether the server transmits live traffic or result paths to the clients). Thus, we omit this model from the remaining discussion.

2.2.1 Raw Transmission Model:

Under the raw transmission model, the traffic data (i.e., edge weights) are broadcasted by a set of packets for each broadcast cycle. Each header stores the latest time stamp of the packets, so that clients can decide which packets have been updated, and only fetch those updated packets in the current broadcast cycle. Having downloaded the raw traffic data from the broadcast channel,

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

the following methods either directly calculate the shortest path or efficiently maintain certain data structure for the shortest path computation.

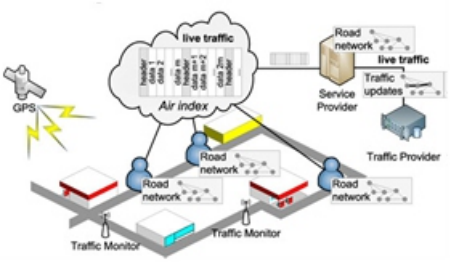
2.2.2 Index Transmission Model:

The index transmission model enables servers to broadcast an index instead of raw traffic data. We review the state-ofthe- art indices for shortest path computation and discuss their applicability on the index transmission model. Road map hierarchical approaches try to exploit the hierarchical structure to the road map network in a pre-processing step, which can be used to accelerate all subsequent queries. These speed-up approaches include reach , highway hierarchies (HH), contraction hierarchies (CH) , and transitnode routing (TNR)

3 LTI OVERVIEW AND OBJECTIVES:

3.1 LTI Overview:

A road network monitoring system typically consists of a service provider, a large number of mobile clients (e.g., vehicles), and a traffic provider (e.g., GoogleMap, NAVTEQ, INRIX, etc.). Fig. 3 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 (e.g., 3G, LTE, Mobile WiMAX, etc.). 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.



5.1 Broadcasting Scheme:

The broadcasting model uses radio or wireless network (e.g., 3G, LTE, Mobile WiMAX) as the transmission medium.

When the server broadcasts a data set (i.e., a "programme"), all clients can listen to the data set concurrently. Thus, this transmission model scales well independent of the number of clients. A broadcasting scheme is a protocol to be followed by the server and the clients. The (1,m) interleaving scheme [23] is one of the best broadcasting schemes. Table 1 shows an example broadcasting cycle with m ¼ 3 packets and the entire data set contains six data items.

First, the server partitions the data set into m equi-sized data segments. Each packet contains a header and a data segment, where a header describes the broadcasting schedule of all packets. In this example, the variables i and n in each header represent the last broadcasted itemand the total number of items. The server periodically broadcasts a sequence of packets (called as a broadcast cycle).

4 LTI CONSTRUCTION:

In Section 4.1, we carefully analyze the hierarchical index structures and study how to optimize the index. InSection 4.2, we present a stochastic based index construction that minimizes not only the size overhead but also reduces the search space of shortest path queries. To the best of our knowledge, this is the first work to analyze the hierarchicalindex structures and exploit the stochastic process to optimizethe index.

4.1 Analysis of Hierarchical Index Structures:

Hierarchical index structures (e.g., HiTi [21], HEPV [20], and Hub Indexing [19], TEDI [33]) enable fast shortest path computation on a portion of entire index which significantlyreduces the tune-in cost on the index transmission model. Given a graph G ¼ ðVG; EGÞ (i.e., road network), this type of index structures partitions G into a set of small sub-graphs SGi and organizes SGi in a hierarchical fashion (i.e., tree). In Fig. 5, we illustrate a graph being partitioned into 10 subgraphs (SG1, SG2; . . . ; SG10) and the corresponding hierarchical index structure. Every leaf entry in a hierarchical structure represents a subgraph SGi that consists of the corresponding nodes and edges from the original graph. For instance, SG1 consists of two nodes VSG1 ¼ fa; bg and one edge ESG1 ¼ fða; bÞg. A non-leaf entry stores the inter-connectivity information between the child entries.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

4.2 Index Construction:

The above discussion shows that it is hard to find a hierarchical index structure I that achieves all optimization objectives. One possible solution is to relax the optimization objectives which makes them be the tunable factors of the problem. While the overhead of pre-computed information (O2) and the number of relevant entries (O3) cannot bedecided straightforwardly, we decide to relax the first objective (i.e., minimizing the size of leaf entries) such that it becomes a tunable factor in constructing the index. To minimize the overhead of pre-computed information (O2), we study a graph partitioning optimization that minimizes the index overhead DSGi through the entire index construction subject to a leaf entry constraint (O1). Subsequently, we propose a stochastic process to optimize the index structure such that the size of the query search graph Gq is minimized (O3).

5 LTI TRANSMISSION:

In this section, we present how to transmit LTI on the air index. We first introduce a popular broadcasting scheme called the ð1;mÞ interleaving scheme in Section 5.1. Based on this broadcasting scheme, we study how to broadcast LTI in Section 5.2 and how a client receives edge updates on air in Section 5.3.

5.1 Broadcasting Scheme:

The broadcasting model uses radio or wireless network (e.g., 3G, LTE, Mobile WiMAX) as the transmission medium. When the server broadcasts a data set (i.e., a "programme"), all clients can listen to the data set concurrently. Thus, this transmission model scales well independent of the number of clients. A broadcasting scheme is a protocol to be followed by the server and the clients.

5.2 LTI on Air:

To broadcast a hierarchical index using the (1,m) interleaving scheme, we first partition the index into two components:

5.3 Client Tune-in Procedures of Air LTI:

We proceed to demonstrate how a client (i.e., driver) receives edge weights from the air index using the hierarchical structure.

Fig. 9 shows the content of a broadcast cycle for a LTI structure in Fig. 7. In this example, the air index uses a ð1; 2Þ interleaving scheme and each data packet stores the edge weight of different subgraphs. For instance, the edge weight of subgraph SG1 are stored in the 2nd packet of a broadcast cycle. Assume that a driver is moving from node b to node d and his navigation system first tunes-in to the air index at the 3rd packet of segment 1. the index structure and the weight of edges. The former stores the index structure (e.g., graph vertices, graph edges, and shortcut edges) and the latter stores the weight of edges. In order to keep the freshness of LTI, our system is required to broadcast the latest weight ofedges periodically.

6 LTI MAINTENANCE:

In order to keep the freshness of the broadcasted index, the cost of index maintenance is necessarily minimized. In this section, we study an incremental update approach that can efficiently maintain the live traffic index according to the updates. As a remark, the entire update process is done at the service provider and there is no extra data structurebeing broadcasted to the clients. There is a bottom-up framework [21] available to maintain the hierarchical index structure according to the updates. Their idea is to re-compute the affected subgraphs starting from lowest level (i.e., leaf subgraphs). Unfortunately, as shown in Section 2.2, a small portion of edge updates trigger updates in the majority of packets (i.e., subgraphs). Thus, the above update technique incurs high computational cost on updating the affected subgraphs.

7 PUTTING ALL TOGETHER:

We are now ready to 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 (cf. Section 5.2), it keeps listening until it discovers a,header segment (cf. Fig. 9). After reading the header segment, it decides the necessary segments (to be read) for computing the shortest path. These issues are addressed in Section 5.3. The client then waits for those segments, reads them, and update the weight of Gq. Subsequently, Gq is used to compute the shortest path in the client machine locally (cf. Fig. 7 and Section 4.1).

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Note that Algorithm 2 is kept running in order to provide online shortest path until the client reaches to the destination. We then discuss about the tasks to be performed by the service provider, as shown in Algorithm 3. 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. We 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 Section 4.2. In each broadcasting cycle, the server first collects live traffic updates from the traffic provider, updates the subgraphs fSGig (discussed in Section 6), and eventually broadcasts them.

Algorithm 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 ▷ Section 5.3
4: decide the necessary segments to be read ▷ Section 5.3
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 ▷ Section 4.1

Algorithm 3 Service Algorithm

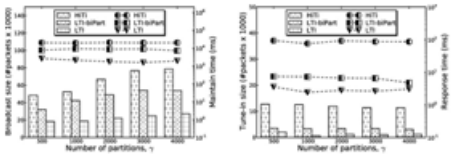
Algorithm *Service*(G:graph)
1: construct I and $\{SG_i\}$ based on G ▷ Section 4.2
2: **for** each broadcast cycle **do**
3: collect traffic updates from the traffic provider
4: update the subgraphs $\{SG_i\}$ ▷ Section 6
5: broadcast the subgraphs $\{SG_i\}$ ▷ Section 5.2

8 EXPERIMENTAL EVALUATION:

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) [28], and dynamic shortest paths tree [22], are applicable to raw transmission model. On the other hand, contraction hierarchies [30],

Hierarchical MulTi-graph model [21], and our proposed live traffic index are applicable to index transmission model. We omit some methods (such as TNR [1], Quadtree [36], SHARC [39], and CALT [31]) 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' movements 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 different 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

TABLE 4
Performance of Different Methods

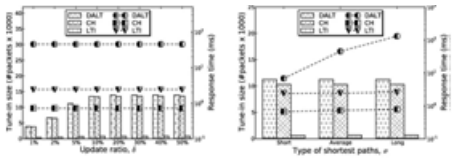
Client side: Tune-in cost (#packets), Response time (ms)
Server side: Broadcast size (#packets), Maintenance time (ms)

City		Method					
		raw transmission model			index transmission model		
		BD	DALT	DSPT	CH	HiTi	LTI
NYC	T	18300.7	18300.7	18300.7	18617.9	25834.5	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

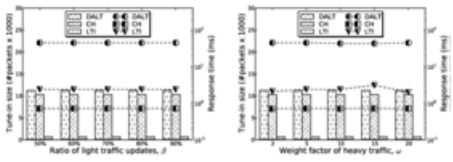
against to LTI-biPart (that is constructed by only the graph partitioning technique, described in Section 4.2) and HiTi [21] (which is the most representative model of hierarchical index structures). For fairness, we internally tune the HiTi graph model by varying the number of children subgraphs, and 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.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The result is shown in Table 4. 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 after receiving the live traffic updates. Thus, their response time is slower5 than BD and DALT on the raw transmission model. Based on the responsetime, DALT is the best approach among the methods in his category.



(a) varying δ (b) varying σ



(c) varying β (d) varying ω

9 CONCLUSION:

In this paper we studied online shortest path computation; 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 experiments 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.

9.ACKNOWLEDGMENTS:

I am B.Prameela and would like to thank the publishers, researchers for making their resources material available. I am greatly thankful to Associate Prof: MR. Mr.B.Ramesh for their guidance. We also thank the college authorities, PG coordinator and Principal for providing the required infrastructure and support. Finally, we would like to extend a heartfelt gratitude to friends and family members.

REFERENCES:

[1] H. Bast, S. Funke, D. Matijevic, P. Sanders, and D. Schultes, “InTransit to Constant Time Shortest-Path Queries 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, series Rand CorporationResearch Study Princeton Univ. Press, 1963.
[4] R.J. Gutman, “Reach-Based Routing: A New Approach to ShortestPath Algorithms Optimized for Road Networks,” Proc. SixthWorkshop Algorithm Eng. and Experiments and the First WorkshopAnalytic Algorithmics 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. European 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.
[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.

Author’s Details:

Miss B.Prameela. MTech student, in M.Tech Student, Dept of CSE in KITS for women’s,kodad, T.S, India.

MR. Mr.B.Ramesh working as a Associate at CSE in KITS for women’s,kodad, T.S, IndiaJNTUH Hyderabad. He has 2 years of UG/PG Teaching Experience