Computing Shortest Path Based on Live Traffic Circumstance

[#1] CH.Pravallika, [*2] Dr.S.Venkateswarlu, [#3] G. Raphi

[1] *M.Tech Student, Narasaraopeta Engineering College, Narasaraopet, Guntur dist, A.P, India.*

[2] *Associate Professor, Narasaraopeta Engineering College, Narasaraopet, Guntur dist, A.P, India.*

[3] *Assistant Professor, Narasaraopeta Engineering College, Narasaraopet, Guntur dist, A.P, India.*

[1] teenachebrolu@gmail.com

[2] svenkateswarlu04@yahoo.co.in

***Abstract* — shortest path computation based on live traffic circumstances is a challenging task and it is imperative as it largely contributed in saving time and resources. So, my project aims at computing the shortest path based on live traffic circumstances. 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, we 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.**

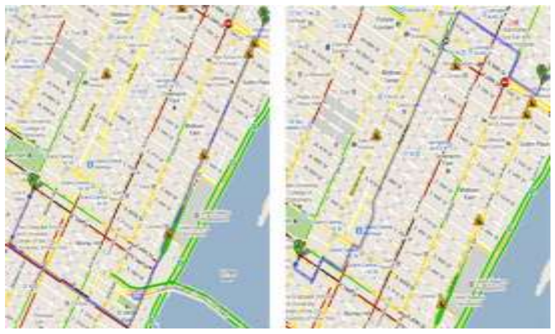
I. INTRODUCTION

Now-a-days, using navigation systems in car to reach their destinations become common practice. In that purview, shortest path computation is an important function in modern car navigation systems and has been extensively studied in [1], [2], [3], [4], [5], [6], [7], [8]. This function helps a driver to figure out the best route from his current position to destination. Typically, the shortest path is computed by offline data pre-stored in the navigation systems and the weight (travel time) of the road edges is estimated by the road distance or historical data. Unfortunately, road traffic circumstances change over time. Without live traffic circumstances, the route returned by the navigation system is no longer guaranteed an accurate result. We demonstrate this by an example in Fig. 1. Suppose that we are driving from Lord & Taylor (label A) to Mt Vernon Hotel Museum (label B) in Manhattan, NY. Those old navigation systems would suggest a route based on the pre-stored distance information as shown in Fig. 1a. Note that this route passes through four road maintenance operations (indicated by maintenance icons) and one traffic congested road (indicated by a red line). In fact, if we take traffic circumstances into account, then we prefer the route in Fig. 1b rather than the route in Fig. 1a.

Nowadays, several online services provide live traffic data (by analyzing collected data from road sensors, traffic cameras, and crowdsourcing techniques), such as Google- Map, Navteq [9], INRIX Traffic Information Provider [10], and TomTom NV [11], etc. These systems can calculate the snapshot shortest path queries based on current live traffic data; however, they do not report routes to drivers continuously due to high operating costs. Answering the shortest paths on the live traffic data can be viewed as a continuous monitoring problem in spatial databases, which is termed online shortest paths computation (OSP) in this work. To the best of our knowledge, this problem has not received much attention and the costs of answering such continuous queries vary hugely in different system architectures.

Typical client-server architecture can be used to answer shortest path queries on live traffic data. In this case, the navigation system typically sends the shortest path query to the service provider and waits the result back from the provider (called result transmission model). However, given the rapid growth of mobile devices and services, this model is facing scalability limitations in terms of network bandwidth and server loading. According to the Cisco Visual Networking Index forecast [12], global mobile traffic in 2010 was 237 petabytes per month and it grew by 2.6-fold in 2010, nearly tripling for the third year in a row. Based on a telecommunication expert, the world's cellular networks need to provide 100 times the capacity in 2015 when compared to the networks in 2011. Furthermore, live traffic is updated frequently as these data can be collected by using crowdsourcing techniques (e.g., anonymous traffic data from Google map users on certain mobile devices). As such, huge communication cost will be spent on sending result paths on this model. Obviously, the client-server architecture will soon become impractical in dealing with massive live traffic in near future. Ku et al. [13] raise the same concern in their work which processes spatial queries in wireless broadcast environments based on Euclidean distance metric.

Malviya et al. [14] developed a client-server system for continuous monitoring of registered shortest path queries. For each registered query (s, t), the server first pre-computes K different candidate paths from s to t. Then, the server periodically updates the travel times on these K paths based on the latest traffic, and reports the current best path to the corresponding user. Since this system adopts the client-server architecture, it cannot scale well with a large number of users, as discussed above. In addition, the reported paths are approximate results and the system does not provide any accuracy guarantee.



(a) Shortest route using live pre-stored (b) Shortest route using weights traffic (by LTI)

Figure 1 Two alternative shortest paths in Manhattan, NY

An alternative solution is to broadcast live traffic data over wireless network (e.g., 3G, LTE, Mobile WiMAX, etc.). The navigation system receives the live traffic data from the broadcast channel and executes the computation locally (called raw transmission model). The traffic data are broadcasted by a sequence of packets for each broadcast cycle. To answer shortest path queries based on live traffic circumstances, the navigation system must fetch those updated packets for each broadcast cycle. However, as we will analyze an example in Section 2.2, the probability of a packet being affected by 1% edge updates is 98.77%. This means that clients almost fetch all broadcast packets in a broadcast cycle.

The main challenge on answering live shortest paths is scalability, in terms of the number of clients and the amount of live traffic updates. A new and promising solution to the shortest path computation is to broadcast an air index over the wireless network (called index transmission model). The main advantages of this model are that the network overhead is independent of the number of clients and every client only downloads a portion of the entire road map according to the index information. For instance, the proposed index in constitutes a set of pairwise minimum and maximum traveling costs between every two sub-partitions of the road map. However, these methods only solve the scalability issue for the number of clients but not for the amount of live traffic updates. As reported in, the re-computation time of the index takes 2 hours for the San Francisco (CA) road map. It is prohibitively expensive to update the index for OSP, in order to keep up with live traffic circumstances.

Motivated by the lack of off-the-shelf solution for OSP, in this paper we present a new solution based on the index transmission model by introducing live traffic index (LTI) as the core technique. LTI is expected to provide 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 OSP. We summarize LTI features as follows.

1) The index structure of LTI is optimized by two novel techniques, graph partitioning and stochastic-based construction, after conducting a thorough analysis on the hierarchical index techniques. To the best of our knowledge, this is the first work to give a thorough cost analysis on the hierarchical index techniques and apply stochastic process to optimize the index hierarchical structure.

2) LTI efficiently maintains the index for live traffic circumstances by incorporating Dynamic Shortest Path Tree (DSPT) into hierarchical index techniques. In addition, a bounded version of DSPT is proposed to further reduce the broadcast overhead.

By incorporating the above features, LTI reduces the tune-in cost up to an order of magnitude as compared to the state-of-the-art competitors; while it still provides competitive query response time, broadcast size, and maintenance time. To the best of our knowledge, we are the first work that attempts to minimize all these performance factors for OSP

II. PRELIMINARY

A. *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.

The index maintenance time and broadcast size relate to the freshness of the live traffic information. The maintenance time is the time required to update the index according to live traffic information. The broadcast size is relevant to the latency of receiving the latest index information. As the freshness is one of our main design criteria, we must provide reasonable costs for these two factors. The last factor is the response time at client side. Given a proper index structure, the response time of shortest path computation can be very fast (i.e., few milliseconds on large road maps) which is negligible compared to access latency for current wireless network speed. The computation also consumes power but their effect is outweighed by communication. It remains, however, an evaluated factor for OSP.

B. 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).

C. DISADVANTAGES OF EXISTING SYSTEM:

1. Scalability limitations in terms of network bandwidth and server loading.

2. Online Shortest Paths computation is not much attention.

III. PROBLEM STATEMENT

A. 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. 2 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.

In this work, we focus on handling traffic updates but not graph structure updates. For real road networks, it is infrequent to have graph structure updates (i.e., construction of a new road) when compared to edge weight updates (i.e., live traffic circumstances). Thus, we assume that the graph structures are distributed to every client in advance (e.g., by monthly updates or at system boot-up) via typical transmission protocol (i.e., HTTP and FTP).

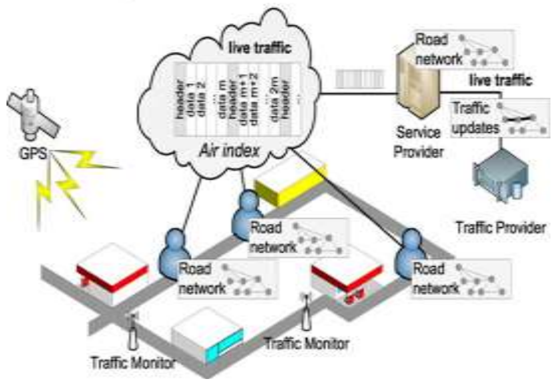


Figure 2: LTI System Overview

B. Lti Construction

Hierarchical index structures (e.g., HiTi [21], HEPV, and Hub Indexing [19], TEDI [33]) enable fast shortest path computation on a portion of entire index which significantly reduces the tune-in cost on the index transmission model. Given a graph G ¼ (VG, EG) (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. In Fig. 3, SG_5 has two border nodes[2] *k* and *m* so that SG_5 keeps a shortcut $\triangle SG_5 = \{(k;m)\}$ and its corresponding weight.

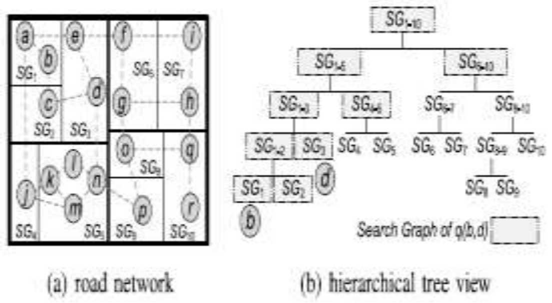


Figure 3: Hierarchical Index structure

To answer a shortest path query q(s, t) using the hierarchical structures, a common approach is to fetch the relevant entries from the index using a bottom-up execution fashion. For the sake of analysis, we use HiTi as our reference model in the remaining discussion. Our analysis can be adapted to other approaches since their execution paradigm shares the same principle.

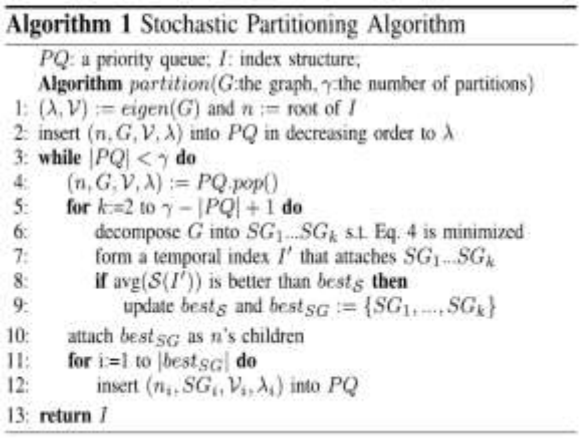
In Fig. 3, the relevant entries of a shortest path query q(b; d) are shaded in gray color. Besides the source and destination leaf entries (SG_1 and SG_3), we need to fetch the entries from two leaf entries towards the root entry (SG_{1-2}, SG_{1-3}, SG_{1-5}, and SG_{1-10}) and their sibling entries (SG_2, SG_{4-5}, and SG_{6-10}).

D. 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 be decided 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 $\triangle SG_i$ 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 G^q is minimized (O3).

At every partitioning, we attempt to find the best structure for the potential queries by the stochastic process.


Algorithm 1 Stochastic Partitioning Algorithm

```
    PQ: a priority queue; I: index structure;
    Algorithm partition(G:the graph, γ:the number of partitions)
 1: (λ, V) := eigen(G) and n := root of I
 2: insert (n, G, V, λ) into PQ in decreasing order to λ
 3: while |PQ| < γ do
 4:     (n, G, V, λ) := PQ.pop()
 5:     for k:=2 to γ − |PQ| + 1 do
 6:         decompose G into SG_1...SG_k s.t. Eq. 4 is minimized
 7:         form a temporal index I' that attaches SG_1...SG_k
 8:         if avg(S(I')) is better than best_S then
 9:             update best_S and best_SG := {SG_1, ..., SG_k}
10:     attach best_SG as n's children
11:     for i:=1 to |best_SG| do
12:         insert (n_i, SG_i, V_i, λ_i) into PQ
13: return I
```

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. Our proposed techniques attempt to optimize the index (O2 and O3) 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.

IV. IMPLEMENTATION

MODULES:

1. Tune-in Cost (Client Side)
2. Broadcast Size(Server Side)
3. Maintenance Time(Server Side)
4. Query Response Time(Client Side)

V. MODULES DESCRIPTION

C. Tune-in Cost (Client Side):

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 avail-ability of parking slots at destination. If we minimize the tune-in cost of one service, then we reserve more resources for other services.

Broadcast Size and Maintenance Time (Server Side):

The index maintenance time and broadcast size relate to the freshness of the live traffic information. The maintenance time is the time required to update the index according to live traffic information. The broadcast size is relevant to the latency of receiving the latest index information. As the freshness is one of our main design criteria, we must provide reasonable costs for these two factors.

Query Response Time (Client Side):

The last factor is the response time at client side. Given a proper index structure, the response time of shortest path computation can be very fast (i.e., few milliseconds on large road maps) which is negligible compared to access latency for current wireless network speed. The computational so consumers power but their effect is outweighed communication. It remains, however, an evaluated factor for OSP.

D. Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- ➢ What data should be given as input?
- ➢ How the data should be arranged or coded?
- ➢ The dialog to guide the operating personnel in providing input.
- ➢ Methods for preparing input validations and steps to follow when error occur.

E. OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

E. OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- ❖ Convey information about past activities, current status or projections of the Future.
- ❖ Signal important events, opportunities, problems, or warnings.
- ❖ Trigger an action.
- ❖ Confirm an action.

VI. CONCLUSION

In this paper I 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.

REFERENCES

[1] H. Bast, S. Funke, D. Matijevic, P. Sanders, and D. Schultes, "In Transit to Constant Time Shortest-Path Queries in Road Networks," 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 Corporation Research Study Princeton Univ. Press, 1963.

[4] 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.

[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 Exact Shortest 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 Evaluation Using Real Road Networks," Transportation Science, vol. 32, no. 1, pp. 65-73, 1998.

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

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

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

[12] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2010-2015," 2011.

Selected Paper from International Conference on Computing (NECICC-2k15)