Online Shortest Path using LTI

M.N Sindhu¹, A.S Rama Chandra Murthy²
¹Lakireddy Balireddy College of Engineering, Mylavaram, Krishna Dist, India.
²Sr. Assistant Professor, Lakireddy Balireddy College of Engineering, Mylavaram, Krishna Dist, India.

Abstract: Finding the shortest path is the one of the tedious problem on the live traffic. There are many existing applications are introduced to overcome this problem but still there is lack of finding the shortest path in live traffic. To overcome this several research has been done on finding shortest traffic especially using wireless sensor networks. The existing method collects the live traffic information and then broadcast them over radio or wireless network. When the communication is not good this approach will fail to give the information. To overcome this Online traffic information system is implemented to every vehicle for communication with the live traffic. Online system will store the live information five min before so that if any communication problem arises our system gives the alerts to the users. Now it is easy to identify the shortest path with in small fraction of time.

KEYWORDS: Shortest path, air index, broadcasting

I. INTRODUCTION

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 Online 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 crowd sourcing techniques), such as Google- Map [9], Navteq [10], INRIX Traffic Information Provider [11], and Tom Tom NV [12], 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 [13], 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 [14], the world’s cellular networks need to provide 100 times the capacity in 2015 when compared to the networks in 2011. Furthermore, live traffic are updated frequently as these data can be collected by using crowd sourcing 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 the this model. Obviously, the client-server architecture will soon become impractical in dealing with massive live traffic in near future. Ku et al. [15] raise the same concern in their work which processes spatial queries in wireless broadcast environments based on Euclidean distance metric. Malviya et al. [16] developed a client-server system for continuous monitoring of registered shortest path queries. For each registered query $\delta; t$, 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.

II. BACKGROUND OF RELATED WORK

In this section, we briefly discuss the applicability of the state-of-the-art shortest path solutions on different transmission models.

1. Towards Online Shortest Path Computation

Creators build up another structure called movement list (LTI) which empowers drivers to rapidly and adequately gather the activity data on the TV channel. The primary disadvantage of this paper is that creators said that this structure is restricted to a couple of portable frameworks just and not all. Again the expense adequacy is less [1].

2. Another methodology for registering briefest way for Road Networks:

The creator propose remote telecast as an option. Also, to encourage successive and exact movement overhauls in this paper, creator outlined another framework SG-LTS (Sub Graph based Traffic Share) framework. The fundamental downside of this paper is that the creator doesn’t proposed any other MST calculation. Creator just demonstrated the idea of sub chart figuring [2].

3. Upgraded online most brief way utilizing movement list approach:

The creators propose a calculation to discover briefest way utilizing Dijkstra calculation. The calculation in this paper can just discover one way and is not qualified for discovering two most limited way for same source and sink hubs. It is the primary disadvantage of this paper [3].

4. Online Shortest Path taking into account Traffic Circumstances:

Creator add to another structure called movement list (LTI) which empowers drivers to rapidly and adequately gather the activity data on the television station. A noteworthy result is that the driver can register/upgrade their briefest way come about by getting just a little division of the record. The primary disadvantage is this paper doesn’t propose any system for substitute most brief way if activity is discovered [4].

5. Most brief Path Algorithm for Virtual Network Construction of Online Shortest Path Computation:

In this paper, creator propose another development for virtual system in order to shape a great deal of virtual systems. Register most brief way utilizing LTI (Traffic Index) The activity supplier gathers the movement statuses from the movement screens by means of systems like street sensors and movement feature investigation. Online element most brief way calculation the briefest way result is processed/redesigned in light of the activity circumstances. The primary issue of this paper is that the redesigned movement sets aside a great deal additional time when contrasted with typical and framework turns out to be moderate [5].

6. Online Shortest Path Computation on Time Dependent Network:

In displayed methodology server will gather movement data and afterward declare them over remote system. With this approach any number of customers can be included. This new approach called activity record time subordinate (LTITD) empowers drivers to overhaul their most limited way come about by getting just a little portion of the file. The proposed framework is infeasible to tackle the issue because of their restrictive upkeep time and expansive transmission overhead [6].

III. CONTRIBUTION
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) [17], [18]. 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 [17] 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 [17], 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. 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 [19], [20], [21]. 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. (Section 4) LTI efficiently maintains the index for live traffic circumstances by incorporating Dynamic Shortest Path Tree (DSPT) [22] into hierarchical index techniques. In addition, a bounded version of DSPT is proposed to further reduce the broadcast overhead. (Section 6) 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. The rest of the paper is organized as follows. We first introduce four main performance factors for evaluating OSP and overview the state-of-the-art shortest path computation methods in Section 2.

System Architecture

IV. PROPOSED METHODOLOGY

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.

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.
1. The server periodically updates the travel times on these paths based on the latest traffic, and reports the current best path to the corresponding user.

2. Efficiently maintains the index for live traffic circumstances.

3. 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.

4. 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.

5. 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.

V. EXPERIMENTAL RESULTS

In this section, we empirically evaluate the performance of some 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 dynamic graph (DALT) [28], and dynamic shortest paths tree [22], are applicable to raw transmission model. On the other hand, contraction hierarchies [30], Hierarchical MulTigraph 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 Olden burg 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 network k-based 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. We adopt the approach in [28] to simulate live traffic updates. The initial weights of edges are assigned by the above network-based generator. In each iteration, we randomly select a set of edges subject to the update ratio $d$ and specific weight update settings. In our work, each weight update can be either a light traffic change, a heavy traffic change, or a road maintenance. The proportion of these update types are $b$, $\frac{1}{2}b$, and $\frac{1}{2}b$, respectively, where $U_{ET AL.}: TOWARDS ONLINE SHORTEST PATH COMPUTATION$ $1021 b$ is a ratio parameter. For each light traffic change, the edge weight is set to 20% of the current weight. For each heavy traffic change, the weight is set to a large value by multiplying a weight factor $v$ (which is set to 5 by default). For each road maintenance, the weight is set to 1. We reset the edge weight to its initial value if the edge weight is updated by heavy traffic or road maintenance after 10 iterations. Implementation and evaluation platforms. All tested methods except CH [30] were implemented in Java. Experiments on the service provider were conducted on an Intel Xeon E5620 2.40 GHz CPU machine with 18 GBytes memory, running Ubuntu 10.10; and experiments on the client were performed on an Intel Core2Duo 2.66 GHz CPU machine with 4GBytes memory, running Windows 7. Table 3 shows the ranges of the investigated parameters, and their default values (in bold). In each experiment, we vary a single parameter, while setting the

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-CSP</td>
<td>Storage-cloud service provider</td>
</tr>
<tr>
<td>POW</td>
<td>Proof of Ownership</td>
</tr>
<tr>
<td>(PKU,skU)</td>
<td>User’s public and secret key pair</td>
</tr>
</tbody>
</table>
others to their default values. For each method, we measure its performance in terms of tune-in size, query response time, broadcast size, and index maintenance time for all tested methods, and report its average performance over 2,000 shortest path queries.

VI. CONCLUSION

In this paper, the Online computation of shortest path is computed on the basis of every vehicle for communication with the live traffic. Online system will store the live information five min before so that if any communication problem arises our system gives the alerts to the users. Now it is easy to identify the shortest path with in small fraction of time. This is the important information used in our solution.

REFERENCES
