A Case for Time-Dependent Shortest Path Computation in Spatial Networks

Ugur Demiryurek, Farnoush Banaei-Kashani and Cyrus Shahabi
Department of Computer Science
University of Southern California
Los Angeles, CA 90089
{demiryur,banaeika,shahabi}@usc.edu

ABSTRACT

The problem of point-to-point shortest path computation in spatial networks is extensively studied with many approaches proposed to speed-up the computation. Most of the existing approaches make the simplifying assumption that weights (e.g., travel-time) 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 of the edge. With this paper, we study the applicability of existing shortest path algorithms to real-world large time-dependent spatial networks. In addition, we evaluate the importance of considering time-dependent edge travel-times for route planning in spatial networks. We show that time-dependent shortest path computation can reduce the travel-time by 36% on average as compared to the static shortest path computation that assumes constant edge travel-times.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications, Spatial Databases and GIS

General Terms

Algorithms

Keywords

Time-Dependent Shortest Path, Route Planning, Spatial Network

1. INTRODUCTION

With the ever-growing popularity of online map applications and their wide deployment in mobile devices and car-navigation systems, an increasing number of users search for point-to-point shortest paths. Accordingly, in the recent years there has been considerable interest by the research community to develop efficient algorithms to compute the optimum-path and the corresponding travel-time between a given source and destination in large road networks (e.g., [14, 11, 18, 15, 16]). All these studies as well as many existing commercial products make the simplifying assumption that the travel-time for each edge of the road network is fixed (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 current traffic congestion and, therefore, is a function of time (i.e., time-dependent).

To illustrate the difference between shortest path computation with static and time-dependent road networks, we show a simple example in Figure 1 where a spatial network is modeled as a graph and edge weights (i.e., travel-times) are time-dependent. Consider the snapshot of the network (i.e., a static network) with edge weights correspond to travel-time values at t=0. With classic shortest path computation approaches that disregard travel-time time-dependency, the shortest path from s to d goes through v_1, v_2, v_4 with a cost of 13 time units. However, by the time when v_2 is reached (i.e., at t=5), the cost of edge $e(v_2, v_4)$ changes from 8 to 12 time units, and hence reaching d through v_2 takes 17 time units instead of 13 as it was anticipated at t=0. In contrast, if the time-dependency of edge travel-times are considered and hence path going through v_3 was taken, the total travel-cost would have been 15 units which is the actual optimal shortest path. We call this shortcoming of the classic shortest path computation techniques as *no-lookahead* problem. Unfortunately, most of the existing online map applications suffer from the no-lookahead shortcoming and, hence, their shortest path recommendation a unique path regardless of the time the query is received. We remark that time-dependent shortest path computation [7] returns different optimum-paths for different departure-times from the source.

Meanwhile, we witness an increasing number of navigation service providers (such as Navteq [12] and TeleAtlas [17]) have started releasing their time-dependent travel-time datasets for road networks at high-resolution temporal granularity as fine as one sample for every five minutes. Considering the availability of high-resolution time-dependent travel-time data for road networks on the one hand, and the importance of time-dependency for accurate and useful route planning on the other hand, the need for efficient approaches to enable online computation of shortest paths in time-dependent networks is apparent and immediate. There are handful of studies [7, 1, 4, 5] that focus on efficient computation of time-dependent shortest path problem. With this paper, we study the applicability of these studies to online computation of time-dependent shortest path in real-world large road networks with time-varying travel-times. In addition, for the first time we show the importance of considering time-dependent edge travel-times in route planning as it improves the travel-time accuracy 36% on average as compared to the shortest path computation that assumes constant edge travel-times (see Section 5.2).

The remainder of this paper is organized as follows. In Sec-

ACM GIS '10 , November 2-5, 2010. San Jose, CA, USA (c) 2010 ACM
ISBN 978-1-4503-0428-3/10/11...$10.00.

tion 2, we review the related work on time-dependent shortest path algorithms. In Section 3, we define the time-dependent shortest path problem in spatial networks. In Section 4, we analyze the importance of time-dependency and the feasibility of existing time-dependent shortest path algorithms. In Section 5, we present the results of our experiments with a variety of spatial networks with real-world time-dependent edge weights. Finally, in Section 6, we conclude and discuss our future work.

2. RELATED WORK

In the last decade, numerous efficient shortest path algorithms with precomputation methods have been proposed (see [16] and [14] for an overview). However, there are a few studies that focus on efficient computation of time-dependent shortest path problem.

Cooke and Halsey [2] first studied the time-dependent shortest path (TDSP) problem where they solved the problem by formulating it in discrete-time and using Dynamic Programming. Another discrete-time solution to TDSP is to utilize time-expanded networks [10]. In general, time-expanded network approaches assume that the edge weight functions are defined over a finite discrete window of time $t \in t_0, t_1, .., t_n$, where t_n is determined by the total duration of time interval under consideration. Therefore, the problem is reduced to the problem of computing, for each time window, minimum-weight paths through the static network where one can apply any of the well-known shortest path algorithms. Although these algorithms are easy to design and implement, they have numerous shortcomings. First, time-expanded models need to create a separate instance of network for each time instance hence leading substantial amount of storage. Second, such approaches can only provide approximate results because the model misses the state of the network between any two discrete-time instants. Finally, it is very hard to decide on the effective choice of discrete-time intervals for real-world applications. In [9], George and Shekhar proposed time-aggregated graph approach where the time-dependent travel-times of each edge are aggregated into time series. Even though this model requires less space than that of time-expanded networks, the results are still approximate.

In [7], Dreyfus showed that the time-dependent shortest problem can be solved by a generalization of Dijkstra's method as efficiently as for static shortest path problems. However, Halpern [8] proved that the generalization of Dijkstra's algorithm is only true for FIFO networks. If the FIFO property does not hold in a time-dependent network, then the problem is NP-Hard. In [13], Orda and Rom introduced a time-dependent shortest path approach based on Bellman-Ford algorithm. With their approach, they determine the path toward destination by refining the arrival-time functions on each node in the whole time interval T. In [9], Kanoulas et al. introduced Time-Interval All Fastest Path (allFP) approach in which they maintain a priority queue of all paths to be expanded instead of sorting the priority queue by scalar values. Therefore, they enumerate all the paths from the source to a destination node which incurs exponential running time in the worst case.

The ALT algorithm was proposed to accelerate shortest path computation in static road networks. With ALT, a set of nodes called landmarks are chosen and then the shortest distances between all the nodes in the network and all the landmarks are computed and stored. ALT employs triangle inequality based on distances to the landmarks to obtain heuristic function to be used in A* search. The time-dependent variant of ALT is studied in [5] where heuristic function is computed w.r.t lower-bound graph. The Contraction Hierarchies (CH) [1] and SHARC [4] methods (also developed for static networks) were augmented to time dependent road networks. We discuss the shortcomings of these approaches in Section 4.1.

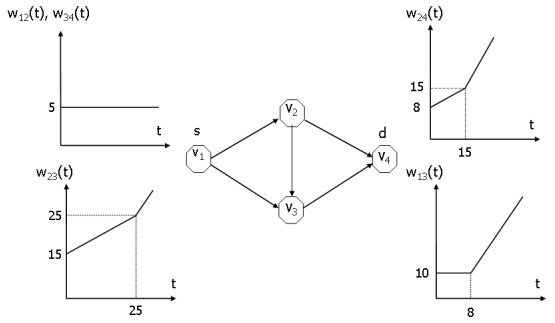


Figure 1: Time-dependent graph

3. PROBLEM DEFINITION

With our study we model the road network as a *time-dependent weighted graph* where the non-negative edge weights (i.e., travel-times) are time-varying. We assume that the time-dependent travel-times are provided as a piecewise linear functions (see Figure 1) which capture the typical congestion pattern for each road segment.

DEFINITION 1. **Time-dependent Graph.** *A Time-dependent Graph is defined as $G(V, E, T)$ where $V = \{v_i\}$ is a set of nodes and $E \subseteq V \times V$ is a set of edges representing the network segments each connecting two nodes. For every edge $e(v_i, v_j) \in E, and v_i \neq v_j$, there is a cost function $c_{v_i,v_j}(t)$, where t is the time variable in time domain T. An edge cost function $c_{v_i,v_j}(t)$ specifies the travel-time from v_i to v_j starting at time t.*

DEFINITION 2. **Time-dependent Travel Cost.** *Let $\{s = v_1, v_2, ..., v_k = d\}$ denotes a path which contains a sequence of nodes where $e(v_i, v_{i+1}) \in E$ and $i = 1, ..., k-1$. Given a $G(V, E, T)$, a path $(s \rightsquigarrow d)$ from source s to destination d, and a departure-time at the source t_s, the time-dependent travel cost $TT(s \rightsquigarrow d, t_s)$ is the travel-time it takes to travel the path. Since the travel-time of an edge varies depending on the arrival-time to that edge, the travel-time of a path is computed as follows:*

$TT(s \rightsquigarrow d, t_s) = \sum_{i=1}^{k-1} c_{v_i,v_{i+1}}(t_i)$ *where* $t_1 = t_s$, $t_{i+1} = t_i + c_{(v_i,v_{i+1})}(t_i), i = 1, .., k$.

DEFINITION 3. **Time-dependent Shortest Path (TDSP).** *Given a $G(V, E, T)$, s, d, and t_s, the time-dependent shortest path $TDSP(s, d, t_s)$ is a path with the minimum travel-time among all paths from s to d.*

In the rest of this paper, we assume that $G(V, E, T)$ satisfies the First-In-First-Out (FIFO) property. We also assume that objects do not wait at any node. In most real-world applications, waiting at a node is not realistic as it means that the moving object must interrupts its travel by getting out of the road (e.g., exit freeway) and finding a place to park and wait.

4. ANALYSIS OF TDSP

In this section, we analyze the time-dependent shortest path computation with respect to two categories. In particular, we i) discuss the applicability of existing algorithms for online computation of time-dependent shortest path in real-world large spatial networks, and ii) assess the importance of time-dependent shortest path by analyzing how much time-dependent planning improves the total travel-time as compared to static shortest path planning.

4.1 Online Computation of TDSP

The time-dependent shortest path problem was first shown by Dreyfus [7] to be polynomially solvable in FIFO networks by a trivial modification to any label-setting (e.g., Dijkstra) or label-correcting (e.g., Bellman Ford) shortest path 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 edge [1]. However, the modified versions of the standard shortest path computation algorithms are far too slow for online applications which are usually deployed on very large networks. For example, answering a time-dependent shortest path query based on the modified Dijkstra algorithm typically takes about 6 seconds for relatively large distances in California road network. Meanwhile, there are many efficient approaches that rely on precomputation to enable answering queries in real-time for online applications (e.g., [14]). These approaches are originally designed for static road networks and it is infeasible to extend them to time-dependent networks mainly due to very large storage and preprocessing requirements. This is because the input size (i.e., the number of shortest paths) increases drastically in time-dependent networks. In particular, since the length of a s-d path changes depending on the departure-time from s, the shortest path is not unique for any pair of nodes in time-dependent networks. It has been conjectured in [3] that the number of shortest paths between any pair of nodes in time-dependent road networks can be super-polynomial. Hence, an algorithm which precomputes the every possible path for any pair of nodes in large time-dependent networks and stores the corresponding path selections would suffer from exponential time and storage complexity. For example, the time-dependent extension of Contraction Hierarchies (CH) [1] and SHARC [4] speed-up techniques (which are proved to be very efficient for static networks) suffer from impractical precomputation times and intolerable storage complexity. Specifically, the main idea behind CH and SHARC is to remove unimportant nodes from the graph without changing the shortest path distances between the remaining (more important) nodes. However, unlike the static networks, the importance of a node can change throughout the time under consideration in time-dependent road networks, hence the importance of the nodes are time varying. Considering the super-polynomial input sizes with time-dependent networks [3], the main shortcomings of these approaches are impractical preprocessing times and extensive space consumption. For example, the precomputation time for SHARC in time-dependent road networks takes more than 2-3 days for relatively small road networks (e.g. Germany). While CH also suffers from slow preprocessing times, the space consumption for CH is 1000 bytes per node for relatively less varied edge-weights. Hence, CH cannot be used in real-world large networks with high traffic (i.e., large variation in edge-weights). On the other hand, the time-dependent ALT approach is more efficient than the time-dependent CH and SHARC in terms of preprocessing time and storage. However, with ALT, the landmark selection is very difficult (relies on heuristics) and the size of the search space and storage is severely affected by the choice of landmarks.

We conclude that the prohibitively high storage and preprocessing times make the existing time dependent shortest path algorithms impractical to be deployed on real-world large spatial networks (e.g., North America road network with approximately 15 million nodes).

[1]The shortest path computation problem is shown to be NP-hard in non-FIFO networks [8].Violation of the FIFO property rarely happens in real-world and hence is not the focus of this study.

4.2 Importance of TDSP

As we discussed there are handful of studies that focus on efficient computation of time-dependent shortest path. However, none of these studies investigate the practicality of time-dependent planning in real-world road networks. With our study, for the first time we assess the importance and the practical usefulness of time-dependent planning by comparing the results of time-independent shortest computation on a real-world spatial network with real time-varying edge travel-times. To this end, we focus on answering two specific questions: i) how much does time-dependent path planing reduce the travel-time as compared to static path planning, and ii) how different are the time-dependent shortest path and the static shortest path for a given source and destination. We answer these question in the following section based on our experimental evaluation with real-world datasets.

5. EXPERIMENTAL EVALUATION

5.1 Experimental Setup

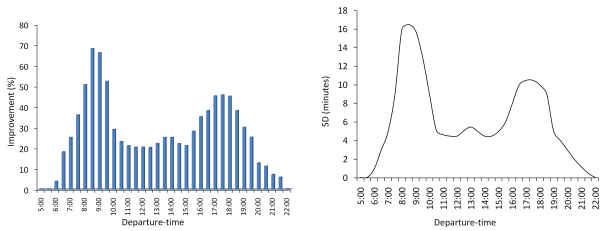
We conducted extensive experiments with different spatial networks to evaluate the practical usefulness of TDSP. As of our dataset, wee used California (CA) and Los Angeles (LA) road network data [12] with approximately 1,965,300 and 304,162 respectively. We conducted our experiments on a workstation with 2.7 GHz Pentium Core Duo processor with 12GB RAM memory.

In our lab, we maintain large-scale and high resolution (both spatially and temporally) traffic sensor (i.e., loop detector) dataset collected from the entire Los Angeles County highways and arterial streets. This dataset includes both inventory and real-time data (with update rate as high as every 1 minute) for 6300 traffic sensors covering approximately 3000 miles. The sampling rate of the streaming data is 1 reading/sensor/min. We have been continuously collecting and archiving the traffic sensor data past two years. We use this real-world dataset create time varying edge weights, we spatially and temporally aggregate sensor data by assigning interpolation points (for each 5 minutes) that depict the travel-times on the network segments. Based on our observation, all roads are un-congested between 9PM and 6AM, and hence we assume static edge weights between those times. In order to create time-dependent edge weights for the local streets in LA as well as the entire CA road network, we developed a traffic modeling approach (based on our observations from LA dataset) that synthetically generates edge travel-time profiles [6]. Our approach uses spatial (e.g., locality, connectivity) and temporal (e.g., rush hour, weekday) characteristics to generate travel-time of network edges that does not have readily available sensor data.

5.2 Results

In this section we report our experimental results from our shorest path queries in which we determine the source s and destination d nodes uniformly at random. We also pick our departure time randomly and uniformly distributed in time domain T. The average results are derived from 1000 random s-d queries.

With our experiment we investigate how much TDSP improves the total travel-time as compared to static shorest path (SP). We use Dreyfus's algorithm [7] to compute time-dependent shortest path for a given s and d. To compute static shortest path (with Dijkstra's algorithm), we use the maximum attainable speed (hence minimum travel-time) on the network edges. We conduct our experiments with the following settings. Given s-d path and for each 5 minutes from 6AM to 9PM, we first determine the time-dependent $P^* = (\{n_1, ..., n_l\}, t)$ and time-independent $P = \{n'_1, ..., n'_k\}$ optimum paths as well as their corresponding travel-times to d,



(a) TDSP vs SP (b) Standard deviation

Figure 2: TDSP and SP Comparison

i.e., $A^t_{P^*}(t)$ and $A_P(t)$, respectively. Next, we compute the actual time-dependent travel-time $A^t_P(t)$ of time-independent path P. Specifically, we take $P = \{n'_1, ..., n'_k\}$ and determine actual time-dependent cost of travel along P departing from n'_1 for a given t. Figure 2(a) plots the improvement gained by the TDSP over its static counter part for which we measure the relative percentage increase of SP's travel-time over that of TDSP computed as $A^t_P(t)/A^t_{P^*}(t) - 1$. As shown,the cost of the path found by the TDSP is on average 36% better than that of SP and the difference is more significant (i.e., 68%, 43%) during rush hours (i.e., 7-9:30AM, 4-6PM). The reason for significant difference during rush hours is that the edge weights change rapidly especially at the boundaries of the traffic peak periods and hence causing an overall increase in the cost of SP. However, TDSP avoids the congestion by selecting alternative segments and hence yields better travel-times. As expected, the paths found by TDSP and SP is often same before 6AM and after 9PM. Figure 2(b) depicts the standard deviation (in minutes) of $A^t_P(t) - A^t_{P^*}(t)$. As illustrated, the standard deviation is also more significant during rush hours.

We also compared the path similarity (number of identical edges in P^* and P) of TDSP and SP. Our results showed that the path found by the TDSP deviates on average 28% with maximum recorded deviation of 87% (where TDSP finds almost completely different path than that of SP) from SP. One interesting observation from this experiment is that although different departure times return different optimum paths, there exists only a limited number of different paths during a day for a given s and d. In particular, we used 180 different departure times, and on average the number of *distinct* optimum-time path computed by time-dependent shortest path algorithm was on average 7, and at most 12.

In sum, we observe that the use of time-dependent information can significantly reduce the travel-times especially during peak hours when the faster travel-time routes needed the most. In addition, our experimental results show that while the paths found by TDSP can deviate largely from SP during rush hours, the number of distinct paths computed by TDSP is relatively small (i.e, on average 7).

6. CONCLUSION AND FUTURE WORK

With this paper, we study the applicability of existing algorithms for computation of time-dependent shortest path in real-world large spatial networks with time-varying edge travel-times. We also present the first comprehensive study on the impact of considering time-varying edge costs for route planning in spatial networks. Our experimental evaluation with real-world spatial networks and traffic data shows that taking the time-dependent edge travel-times into account can reduce the travel-time by 36% on average as compared to static shortest path computation that assumes constant edge weights. We also observe that although time-dependent shortest path computation returns different optimal-paths for different departure-times, there are only a limited number of distinct paths (i.e., 7 on average) for a given source and destination. We believe that these observations open a door to several practical directions for future work.

We intend to pursue this study in two different directions. First, we intend to investigate new data models for effective representation of time-dependent spatial networks. This is critical in support of developing efficient and accurate time-dependent algorithms, while minimizing the storage and computation cost. Second, given that existing time-dependent shortest path algorithms are impractical with large scale spatial networks, we plan to develop efficient and scalable preprocessing techniques that can be used to accelerate the time-dependent shortest path computation.

7. ACKNOWLEDGMENTS

This research has been funded in part by NSF grant CNS-0831505 (CyberTrust), the NSF Integrated Media Systems Center (IMSC), unrestricted cash and equipment gift from Google, and in part from the METRANS Transportation Center, under grants from USDOT and Caltrans. Any opinions, findings, and conclusions expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

8. REFERENCES

[1] G. V. Batz, D. Delling, P. Sanders, and C. Vetter. Time-dependent contraction hierarchies. In *ALENEX*, 2009.

[2] L. Cooke and E. Halsey. The shortest route through a network with timedependent internodal transit times. In *Journal of Mathematical Analysis and Applications*, 1966.

[3] B. C. Dean. Algorithms for minimum-cost paths in time-dependent networks with waiting policies. In *Networks*, 2004.

[4] D. Delling. Time-dependent sharc-routing. In *ESA*, 2008.

[5] D. Delling and D. Wagner. Landmark-based routing in dynamic graphs. In *WEA*, 2007.

[6] U. Demiryurek, B. Pan, F. B. Kashani, and C. Shahabi. Towards modeling the traffic data on road networks. In *SIGSPATIAL-IWCTS*, 2009.

[7] S. E. Dreyfus. An appraisal of some shortest-path algorithms. In *Operations Research Vol. 17, No. 3*, 1969.

[8] J. Halpern. Shortest route with time dependent length of edges and limited delay in nodes. In *MMO*, 1969.

[9] E. Kanoulas, Y. Du, T. Xia, and D. Zhang. Finding fastest paths on a road network with speed patterns. In *ICDE*, 2006.

[10] E. Köhler, K. Langkau, and M. Skutella. Time-expanded graphs for flow-dependent transit times. In *ESA*, 2002.

[11] U. Lauther. An extremely fast, exact algorithm for finding shortest paths in static networks with geographical background. In *Geoinformation and Mobilitat*, 2004.

[12] NAVTEQ. http://www.navteq.com, Accessed in May 2010.

[13] A. Orda and R. Rom. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *J. ACM*, 1990.

[14] H. Samet, J. Sankaranarayanan, and H. Alborzi. Scalable network distance browsing in spatial databases. In *SIGMOD*, 2008.

[15] P. Sanders and D. Schultes. Highway hierarchies hasten exact shortest path queries. In *ESA*, 2005.

[16] P. Sanders and D. Schultes. Engineering fast route planning algorithms. In *WEA*, 2007.

[17] TELEATLAS. http://www.teleatlas.com, Accessed in May 2010.

[18] D. Wagner and T. Willhalm. Geometric speed-up techniques for finding shortest paths in large graphs. *In ESA*, 2003.