Journal of Applied Mathematics and Physics, 2016, 4, 1-7
Published Online January 2016 in SciRes. http://www.scirp.org/journal/jamp
http://dx.doi.org/10.4236/jamp.2016.41001

Backward Dijkstra Algorithms for Finding the Departure Time Based on the Specified Arrival Time for Real-Life Time-Dependent Networks

Gelareh Bakhtyar[1], Vi Nguyen[2], Mecit Cetin[1], Duc Nguyen[1,3]

[1]Department of Civil and Environmental Engineering, Old Dominion University, Norfolk, USA
[2]Department of Mechanical and Aerospace Engineering, Old Dominion University, Norfolk, VA, USA
[3]Department of Modeling, Simulation, and Visualization Engineering, Old Dominion University, Norfolk, VA, USA
Email: gbakh001@odu.edu, vnguy025@odu.edu, mcetin@odu.edu, dnguyen@odu.edu

Received 9 November 2015; accepted 5 January 2016; published 12 January 2016

Abstract

A practical transportation problem for finding the "departure" time at "all source nodes" in order to arrive at "some destination nodes" at specified time for both FIFO (*i.e.*, First In First Out) and Non-FIFO "Dynamic " Networks is considered in this study. Although shortest path (SP) for dynamic networks have been studied/documented by various researchers, contributions from this present work consists of a sparse matrix storage scheme for efficiently storing large scale sparse network's connectivity, a concept of Time Delay Factor (TDF) combining with a "general piecewise linear function" to describe the link cost as a function of time for Non-FIFO links' costs, and Backward Dijkstra SP Algorithm with simple heuristic rules for rejecting unwanted solutions during the backward search algorithm. Both small-scale (academic) networks as well as large-scale (real-life) networks are investigated in this work to explain and validate the proposed dynamic algorithms. Numerical results obtained from this research work have indicated that the newly proposed dynamic algorithm is reliable, and efficient. Based on the numerical results, the calculated departure time at the source node(s), for a given/specified arrival time at the destination node(s), can be *non-unique*, for some Non-FIFO networks' connectivity.

Keywords

Backward Dijkstra, Dynamic Networks, Piece-Wise Linear Function, Specified Arrival Time

1. Introduction

For most people who have to commute from their homes to their work-places, they want to have the answers for either of the following questions: if we leave our home at a specified time, what time we will arrive at the office?

How to cite this paper: Bakhtyar, G., Nguyen, V., Cetin, M. and Nguyen, D. (2016) Backward Dijkstra Algorithms for Finding the Departure Time Based on the Specified Arrival Time for Real-Life Time-Dependent Networks. *Journal of Applied Mathematics and Physics*, **4**, 1-7. http://dx.doi.org/10.4236/jamp.2016.41001

Or what time we should depart from home in order to arrive at the office at a specified time? Similar questions have been asked by long distance travelers, etc.

The vast majority of Shortest Path Problem (SPP) publications have dealt with *static* (*i.e.*, non-time-dependent) networks that have fixed topology and constant link costs. In recent years, there has been a renewed interest in the study of Time-Dependent Shortest Path Problems (TDSPP). Thus, one of the fundamental network problems in such applications is the computation of the shortest paths from all nodes to a set of destination nodes for all possible departure time, in a given time-dependent network.

Orda and Rom [1] have presented an algorithm for finding the shortest path and minimum delay under various waiting constraints, and for all instances of time. The properties of the derived path (under arbitrary functions for link delays) are also investigated in their studies. Daganzo [2] have solved the backward SSP on a network with FIFO links. The FIFO property means "First In First Out" and states that if a vehicle leaves node *i* at time t_1 and the other one leaves the same node at time $t_2 > t_1$, then the second vehicle cannot arrive at node *j* before the first one. Chabini and Ganugapati [3] have proposed an efficient dynamic solution algorithm, call algorithm DOT, and prove that no sequential algorithm with a better worst-case computational complexity can be developed. Wuming and Pingyang [4] introduced an algorithm to solve the shortest paths in time-dependent network by converting Non-FIFO network to a FIFO network and solve the problem using the traditional SSP algorithms. Ding, Yu, and Qin [5] have proposed a new Dijkstra-based algorithm by decoupling *path-selection* and *time-refinement* in the starting-time interval *T*. They have also established/proved the time complexity and space complexity based on their proposed 2 step approached. Through extensive numerical studies, they have also concluded that their dynamic algorithm outperforms existing solution algorithms in terms of efficiency.

Bidirectional Dijkstra search is a standard technique to speed up computations on *static* networks. However, since the arrival time at the destination is unknown, the cost of time-dependent links around the target node cannot be evaluated, thus bidirectional search cannot be directly applied on time-dependent networks. Nannicini [6] has proposed a solution to the above problem by using a time-independent lower bounding function in the backward search.

Computational strategies for families of Frank-Wolfe (FW), Conjugate FW, Bi-conjugate FW Deterministic User Equilibrium (DUE) algorithms for static networks have also been reported by Allen [7].

The focus of this paper is to find the departure time at the source node(s) for a given (or specified) arrival time at the destination node(s) in FIFO, and Non-FIFO networks. This present work consists of a sparse matrix storage scheme for efficiently storing large scale sparse network's connectivity, a concept of Time Delay Factor (TDF) combining with a general piece-wise linear function to describe the link cost as a function of time (for Non-FIFO links' costs), and Backward Dijkstra SP Algorithm with simple heuristic rules for rejecting unwanted solutions during the backward search algorithm.

The remaining of this paper is organized as following. Dynamic networks are discussed in Section 2, where the concept of TDF in conjunction with piece-wise linear time function for the links' costs are also introduced in this section. A small numerical example of a dynamic network (with 5 nodes and 9 links) is used in Section 3 to facilitate the discussions of the Polynomial LCA and Forward Dijkstra algorithms for finding the arrival time at the destination node, based on the known departure time at the source node. Furthermore, this same dynamic network will also be used in Section 3 for finding the departure time at the source node in order to arrive at the destination node at a given time. The issue of unique (or non-unique) solution for this focused problem (*i.e.* finding the departure time at the source node for a specified arrival time at the destination node) is also discussed in Section 3. Real-life (large-scale) dynamic transportation networks are investigated, using the proposed time-dependent Backward Dijkstra algorithm, and the numerical results are presented in Section 4 to validate the proposed dynamic algorithm. Finally, conclusion is summarized in Section 5.

2. Time Delay Factor and Piece-Wise Linear Time Function in Dynamic Networks

For dynamic networks, the time to travel from node "*i*" to node "*j*" of a particular link "*k*" is no longer a constant. The actual travel time on link "*k*" will depend on the departure time at node "*i*". In this work, the following formulas are employed for a typical link "*k*", connected by node "*i*" to node "*j*":

$$AT = DT + CST * TDF(DT) \tag{1}$$

where $AT =$ Arrival Time at node "*j*" for a typical link "*k*", $DT =$ Departure Time at node "*i*" for a typical

link "k", CST = Constant "Static" Time for a typical link "k", $TDF(DT)$ = Time Delay Factor (TDF), which is dependent on DT and can be defined as Equation (2), and $y(DT)$ is the appropriated time function for a typical link "k".

$$TDF(DT) = 1 + y(DT) \tag{2}$$

In this work, the time function (which depends on DT) can be represented as shown in **Figure 1** where piece-wise linear time function is used. For typical travel time in real dynamic networks, travel time will be increased during certain hours of the day, say during 6 am - 8 am in the morning (due to morning rush, since travelers drive to work), and say during 16 hours-18 hours (or 4:00 pm - 6:00 pm, since travelers leave their offices for heading homes).

In **Figure 1**, the coordinates ($DT, y(DT)$) of such points O, A, B, C, D, E, F, G, H, and I are defined as the input parameter (provided by the software user). Thus, this piece-wise linear time function can be (conveniently, and appropriately) provided to take into account of different local traffic congested time. In general, one may have different function $y(DT)$ for different links. However, in our research work, we assume that all links (see **Figure 2(a)**) will have the same travel behavior as the one shown in **Figure 1**.

The value of $y(DT)$ can be varied (say, from 0.00 to 1.00 as indicated in **Figure 1**). Thus, for static networks, the TDF defined in Equation (2) is equal to 1 (by setting $y(DT)$ = 0.00), while for dynamic networks, the value of TDF could be any where within the range $[1.00 - 2.00]$, depending on the value of $y(DT)$. The following 2 important observations can be made:

1) On a typical link "k", if the **departure time** at node "i" is known, then the arrival time at node "j" can be uniquely and easily computed (by using Equation (1-2), and **Figure 1**).

2) On a typical link "k", if the **arrival time** at node "j" is known, then the departure time at node "i" can also be computed (by using Equation (1)-(2), and **Figure 1**). However, in this case, the computed departure time at node "i" may NOT be unique. Some sorts of elimination (heuristic) rules need be developed in order to find an acceptable single solution.

3. Finding the Departure Time at the Source Node(s) Based on the Specified Arrival Time at the Destination Node(s)

In this section, a dynamic network with 5 nodes and 9 links, shown in **Figure 2(a)**, will be analyzed. For convenience, all links will be assumed to have the same time function as illustrated in **Figure 1**. The following problem's cases will be investigated.

Problem 1. Use the polynomial LCA (time dependent) method to find the time dependent shortest path from any source node, say $s = 5$ to any destination node, say $t = 2$ at the following three possible departure time:

Case (a): 9 hrs. = 9:00 am (to simulate right after rushed /busy hours, see **Figure 1**).

Case (b): 15 hrs. = 3:00 pm (to simulate right before rushed /busy hours, see **Figure 1**).

Case (c): 16.75 hrs. = 4:45 pm (to simulate during rushed /busy hours, see **Figure 1**).

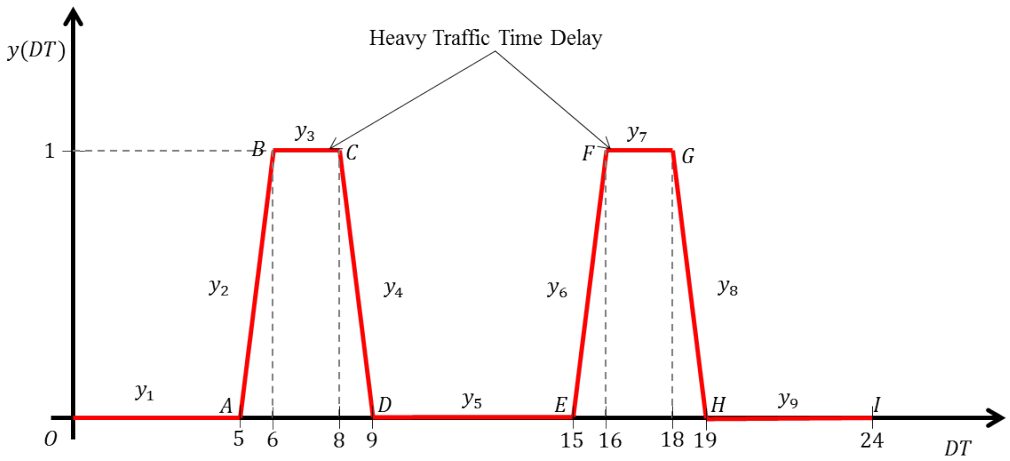

Figure 1. Piece-wise linear time function for a typical link "k".

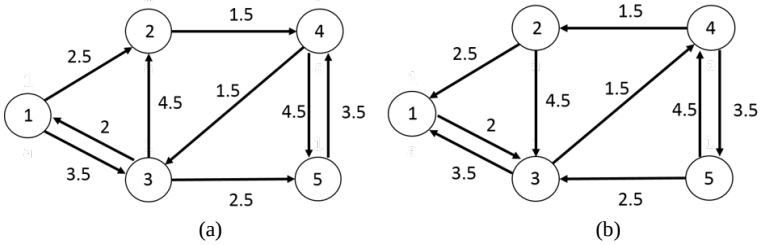

Figure 2. (a) A dynamic network topology; (b) A dynamic reversed network topology with 5 nodes and 9 links.

This problem is rather straight forward, since the departure time (DT) is known at the source node 5. For any subsequent links $i-j$, since the departure time of node "*i*" for links $i-j$ is known, hence:

The function $y(DT)$ can be easily/uniquely determined (from **Figure 1**), the Time Delay Factor (TDF) can be easily/uniquely determined (from Equation (2)), and the arrival time (AT) at node "*j*" can be easily/uniquely determined (from Equation (1)). Eventually, the AT at node 2 (for all cases a, b, and c) were found and presented in **Table 1**.

Problem 2. Re-do problem 1 (for all cases a, b, and c), but using the regular forward time dependent Dijkstra algorithm.

The final results are identical to the one obtained in Problem 1 (using Polynomial LCA time-dependent algorithm (see **Table 1**)).

Problem 3. Find the departure time for the known arrival time using dynamic backward Dijkstra algorithm for all three cases of the previous problem. Based on the numerical results obtained in problems 1 and 2, we knew that if the driver *departs* from the *source* node 5 at 9:00am (case a), or at 3:00pm (case b), or at 4:45pm (case c), then he/she will *arrive* at the *destination* node 2 at 16.00 (or 4:00pm), or at 24.00 (or mid-night), or at 26.25 (or 2:15am next day), respectively.

To find the solutions for the above questions, our proposed modified dynamic backward Dijkstra algorithms can be summarized in the following major steps:

Step 1. Revised the links' direction of the given network (see **Figure 2(b)**). The given arrival times (obtained from problems 1 and 2) can be used as the known departure time at the source node 2.

Step 2. In this step, we would like to find "what time the driver should *depart* from the *source* node *i* (for link $i-j$), in order to *arrive* at the *destination* node *j* at a specified time?". For this situation, Equation (1) can also be used. However, the known variables are AT and CST, and the unknown variable is DT. This is completely different from the defined problems 1 and/or 2, where the known variables are DT and CST, and the unknown variable is AT. While the unknown variable AT can be easily (and uniquely) found from Equation (1) for Problems 1 and 2, the unknown variable DT for Problem 3 may not be easily (and/or uniquely) found from Equation (1). Combining Equation (1) and Equation (2), one obtains:

$$AT = DT + CST \times [TDF = 1 + y(DT)] \tag{3}$$

The only unknown in Equation (3) is Departure time (DT). To illustrate this point, the following numerical details are provided and explained for Problem 3, case (b), where we start with the known AT at node 2 as 24.00 (or mid-night). Starting from node $j=2$, find all the connected out-going links $j-i$ (based on **Figure 2(b)**).

Start first iteration, when $distance = \{Inf \quad 0 \quad Inf \quad Inf \quad Inf\}$, $predecesor = \{0 \quad 0 \quad 0 \quad 0 \quad 0\}$, $S = \{2\}$ (the array of explored nodes), $AT = 24.00$ (Given arrival time at destination node). For Outgoing link 2-1, we have $jnode = 1, AT = 24.00$, and $CST(\,for\ link\ 2-1) = 2.5$, Equation (3) will give the 9 computed DT values (corresponding to the 9 time functions $y_1, y_2, y_3, \ldots, y_9$ shown in **Figure 1**) as following:

$$\begin{aligned} &\{DT_1 \quad DT_2 \quad DT_3 \quad DT_4 \quad DT_5 \quad DT_6 \quad DT_7 \quad DT_8 \quad DT_9\} \\ &= \{21.5 \quad 9.71 \quad 19 \quad 0.67 \quad 21.5 \quad 16.9 \quad 19 \quad 17.3 \quad 21.5\} \end{aligned} \tag{4}$$

However, **Figure 1** implies that the time function $y_1(DT)$ is only valid if DT is within the range [0.00 - 5.00 hours], the time function $y_2(DT)$ is only valid if DT is within the range [5.00 - 6.00 hours], the time

Table 1. Numerical Results for Dynamic Network in **Figure 2**.

Case	Source Node	Destination Node	Departure Time	Arrival Time	Shortest Time (Cost)	Path	Number of Explored Nodes
Polynomial LCA & Forward Dijkstra							
a	5	2	9	16	7	5 → 3 → 2	5
b	5	2	15	24	9	5 → 3 → 1 → 2	5
c	5	2	16.75	26.25	9.5	5 → 3 → 2	5
Backward Dijkstra							
a	2	5	9	16	7	2 → 3 → 5	4
b	2	5	15.5714	24	8.4286	2 → 3 → 5	4
c	2	5	19.25	26.25	7	2 → 3 → 5	4

function $y_3(DT)$ is only valid if DT is within the range [6.00 - 8.00 hours], the time function $y_4(DT)$ is only valid if DT is within the range [8.00 - 9.00 hours], the time function $y_5(DT)$ is only valid if DT is within the range [9.00 - 15.00 hours], the time function $y_6(DT)$ is only valid if DT is within the range [15.00 - 16.00 hours], the time function $y_7(DT)$ is only valid if DT is within the range [16.00 - 18.00 hours], the time function $y_8(DT)$ is only valid if DT is within the range [18.00 - 19.00 hours], and the time function $y_9(DT)$ is only valid if DT is within the range [19.00 - 24.00 hours].

For the above reasons/restrictions, out of the 9 computed DT (shown in Equation (4)), we can only accept the value $DT = DT_9 = 21.5$ hours, with the value $y(DT_9) = 0.00$, which correspond to the $TDF = 1.0$. We can update our information as below:

$$distance(jnode) = distance(S(end)) + link\ cost \times TDF,$$

$$distance = \{2.5 \quad 0 \quad Inf \quad Inf \quad Inf\},\ predecesor = \{2 \quad 0 \quad 0 \quad 0 \quad 0\}.$$

For outgoing link 2-3, we have $AT = 24.00$, and CST (for link 2 - 3) = 4.5, Equation (3) will give the 9 computed DT values (corresponding to the 9 time functions: $y_1, y_2, y_3, \ldots, y_9$, shown in **Figure 1**) as following:

$$\begin{aligned}&\{DT_1 \quad DT_2 \quad DT_3 \quad DT_4 \quad DT_5 \quad DT_6 \quad DT_7 \quad DT_8 \quad DT_9\}\\ &= \{19.5 \quad 7.6 \quad 15 \quad 6 \quad 19.5 \quad 15.8 \quad 15 \quad 18.9 \quad 19.5\}\end{aligned} \tag{5}$$

Based on the restrictions imposed on the 9 functions $y(DT)$, shown in **Figure 1**, out of the 9 computed DT (shown in Equation (5)), there were 3 possible solutions for $DT = DT_6 = 15.8$, or $DT = DT_8 = 18.9$, or $DT = DT_9 = 19.5$ hours. The corresponding values for $\{y_6 \quad y_8 \quad y_9\} = \{0.8182 \quad 0.1429 \quad 0.00\}$, and $\{TDF_6 \quad TDF_8 \quad TDF_9\} = \{1.82 \quad 1.14 \quad 1.00\}$. Among the 3 possible solutions for DT (such as $DT = DT_6$, or $DT = DT_8$, or $DT = DT_9$), we select the *largest* $DT = DT_9 = 19.5$ hours, since this choice will correspond to the *smallest* $TDF = TDF_9 = 1.00$. In other words, our selected choice will give the smallest TDF which will give the smallest travel cost for this particular link. We can update our information as below:

$$DT = 19.5,\quad distance(jnode) = distance(S(end)) + link\ cost \times TDF,$$

$$distance = \{2.5 \quad 0 \quad 4.5 \quad Inf \quad Inf\},\ predecesor = \{2 \quad 0 \quad 2 \quad 0 \quad 0\}.$$

The next node to explore is node 1 (*i.e.*, $next = 1$), so the second iteration can start by searching toward all the outgoing links from node 1 in which the arrival time at node 1 is 21.5 ($AT = 21.5$), and $S = \{2 \quad 1\}$. The algorithm will stop when the next node to explore is the destination node.

Eventually, the AT at node 5 for all cases a, b, and c of the problem were found, and presented in **Table 1**. Thus, for certain *dynamic* networks, there may be more than one solution for the *departure time* at node *i* (say

$i = 5$) which still give the same *specified arrival time* at node *j* (say $j = 2$). By using the suggested criterion to select the value of DT, the resulted path will also *often* corresponds to the SP as well.

4. Numerical Result for Large Scale Real-Life Networks

In this section, 12 large-scale examples based on real-life networks data have been solved using the regular forward Dijkstra, and backward Dijkstra algorithms. The regular forward Dijkstra algorithm is employed to find the *arrival time* at the *destination node j*, based on the *known departure time* at the *source node i*. The backward Dijkstra algorithm is employed to find the *departure time* at the *source nodei*, based on the *known* (*specified*) *arrival time* at the *destination node j*. For cases where *multiple solutions* for DT do exist, we will select the DT which gives the smallest value of $y(DT) = y_{\min}$, which corresponds to the smallest value for $TDF = TDF_{\min}$. This is the criterion which has been used in Section 3.

For convenient purposes, the arrival time at the destination node *j* of the Forward Dijkstra algorithm will be used as the departure time for the destination *j* of the Backward Dijkstra algorithm, for the same network with reversed links' directions. All numerical results are compiled and tabulated in **Table 2**.

For the problem of finding the departure time at the source node(s) based on the specified/given arrival time at the destination node(s), and based on the numerical results presented in **Table 2**, the following major observations can be made:

a) Unique solution has been obtained in all examples except example 2 and 11.

b) Multiple (or non-unique) solutions have been found in examples 2, and 11. For these examples, different *departure time* at the *source node* can lead to the same *specified arrival time* at the *destination node*. In example 2, if the driver departs at the source node 25 at either 6.00 hours, or at 7.236 hours, he/she still arrives at the destination node 110 at the specified time 21.7647 hours. Of course, if the driver departs at the source node 25 at 7.9032 hours, then not only he will arrive at his destination node on time (at the specified time 21.7647 hours), but this selected path will also be the shortest path.

Table 2. Comparisons of forward and backward Dijkstra results for real networks.

Example	Network Name	Source w.r.t. Forward Search	Destination w.r.t. Forward Search	Forward Search			Backward Search (Y_{min})	
				Departure Time	Arrival Time	Cost	Back Calculated Departure Time	Cost
1	Winnipeg	5	100	6	16.494	10.494	6	10.494
2	Winnipeg	25	110	6	21.764	15.764	7.236	14.528
2	Winnipeg	25	110	7.236	21.764	14.528		
3	Barcelona	5	400	6	10.587	4.5876	6.0002	4.587
4	Barcelona	15	400	5	11.954	6.954	5	6.954
5	Austin	56	1800	1	22.855	21.855	1	21.855
6	Austin	156	1500	6	18.735	12.735	6.0007	12.734
7	Austin	5	6100	23	53.041	30.041	23	30.041
8	Austin	1	7388	6	22.797	16.797	5.9993	16.797
9	Philadelphia	6	560	1	13.481	12.481	1	12.481
10	Philadelphia	36	510	7	22.7	15.7	6.9996	15.700
11	Philadelphia	48	1415	1	63.352	62.352	1.5262	61.826
11	Philadelphia	48	1415	1.526	63.352	61.826		
12	Philadelphia	100	1429	6	57.165	51.165	6.0001	51.165
13*	Winnipeg	25	110	6	25.020	19.020	6	19.020
14*	Philadelphia	48	1415	1	199.32	198.32	1	198.32

*FIFO Network (example 1 through 12 correspond to Non-FIFO Network).

c) In example 11, the driver departs from source node 48 at 1.00 hour (=1:00am), and he/she will arrive at the destination node 1415 at 63.3523 hours (=63.3523 − 48 = 15.3523 hours, two days later), based on the Forward Dijkstra search. Based on the Backward Dijkstra search, the driver should depart (at the same source node) at 1.5262 hours (=1:5262 am hours), if he/she wishes to arrive at the same destination node at the specified time 63.3523 hours.

5. Conclusions

In this paper, the well-known polynomial LCA, and the Regular Forward Dijkstra algorithms have been conveniently applied to dynamic (time dependent) networks, through the concept of piece-wise linear function and Time Delay Factor (*TDF*) which is a function of the departure time (*DT*) at node "*i*" for a typical link $i - j$.

The practical problems of finding the departure time at the source node(s) based on the specified/given arrival time at the destination node(s) can be efficiently solved by using the proposed Backward Dijkstra algorithm, which basically employs the Forward Dijkstra algorithm on the same dynamic network with all links' direction are reversed. Extensive numerical results based on a small-scale (academic) dynamic network (with 5 nodes, and 9 links), as well as using 12 real-life (large-scale) dynamic networks, seem to indicate that:

i) The proposed Backward Dijkstra (time dependent) algorithms always find the correct departure time at the source node "*i*" that will guarantee to arrive at the destination node "*j*" at the specified/given arrival time.

ii) For FIFO dynamic networks, the computed paths correspond to the shortest paths, and the solution is unique.

iii) For certain NON-FIFO dynamic networks, the computed paths often correspond to the shortest paths, although SP is not a requirement for the type of time-dependent problems considered in this work.

iv) Depending on the particular NON-FIFO dynamic network, the computed solution(s) might be unique or non-unique where multiple solutions do exist.

Acknowledgements

This paper was in part funded by Mid-Atlantic Transportation Sustainability University Transportation Center (MATS UTC, for the third author), and by the NSF grant#1440673 (for the last author).

References

[1] Orda, A., and Rom R. (1990) Shortest Path and Minimum Delay Algorithms in Networks with Time-Dependent Edge Length. *J. ACM*, **37**, 607-625. http://dx.doi.org/10.1145/79147.214078

[2] Daganzo, C.F. (2002) Reversibility of the Time-Dependent Shortest Path Problem. *Transportation Research Part B*: *Methodological*, **36**, 665-668. http://dx.doi.org/10.1016/S0191-2615(01)00012-1

[3] Chabini, I. and Ganugapati, S. (2002) Parallel Algorithms for Dynamic Shortest Path Problems. *International Transactions in Operational Research*, **9**, 279-302. http://dx.doi.org/10.1111/1475-3995.00356

[4] Luo, W.M. and Han, P.Y. (2007) Study on Non-FIFO Arc in Time-Dependent Networks. *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*. http://dx.doi.org/10.1109/SNPD.2007.445

[5] Ding, B., Yu, J.X. and Qin, L. (2008) Finding Time-Dependent Shortest Paths over Large Graphs. *EDBT*'2008 *Proceedings of the* 11*th International Conference on Extending Database Technology*: *Advances in Database Technology*, 205-216. http://dx.doi.org/10.1145/1353343.1353371

[6] Nannicini, G. (2009) Point-to-Point Shortest Paths on Dynamic Time-Dependent Road Networks. Ph.D. Dissertation, Ecole Polytechnique, France.

[7] Allen, S.E. (2013) Parallel Implementations of the Frank-Wolfe Algorithms for the Traffic Assignment Problem. M.Sc. Thesis, MSVE Department, Old Dominion University, Norfolk, USA.