Measurement 60 (2015) 299–307

Contents lists available at ScienceDirect

Measurement

journal homepage: www.elsevier.com/locate/measurement

A decremental approach with the A^* algorithm for speeding-up the optimization process in dynamic shortest path problems

Mostafa K. Ardakani [a], Madjid Tavana [b,c,*]

[a] School of Engineering Technology, State University of New York, Farmingdale, NY, USA
[b] Business Systems and Analytics Department, Lindback Distinguished Chair of Information Systems and Decision Sciences, La Salle University, Philadelphia, PA 19141, USA
[c] Business Information Systems Department, Faculty of Business Administration and Economics, University of Paderborn, D-33098 Paderborn, Germany

ARTICLE INFO

Article history:
Received 31 July 2014
Received in revised form 16 September 2014
Accepted 1 October 2014
Available online 22 October 2014

Keywords:
A^* algorithm
Continuous-time dynamic network
Shortest path problem
Multidimensional scaling
Real-time travel-time

ABSTRACT

Dynamic (time-dependent) network routing has become important due to the deployment of advanced traveler information systems in navigation systems. We study the problem of speeding-up the shortest path in continuous-time dynamic networks without *a priori* knowledge of the link travel times. We apply the A^* algorithm using the decremental approach to reduce the network size and speed-up the optimization process in dynamic shortest path problems. We utilize the weighted metric multidimensional scaling technique to define the potential function in the A^* algorithm by converting the link travel costs (times) into distances. Moreover, we evaluate the performance of the decremental approach with respect to the CPU times and optimal costs by applying the A^* algorithm and Dijkstra's algorithm to different networks.

1. Introduction

Traffic congestion is a daily phenomenon caused by a growing amount of traffic and limited capacity of the road network. Traffic congestion is often triggered by predictable events (e.g., commuter traffic), less predictable events (e.g., weather), or unpredictable events (e.g., accidents). In order to relieve congestion, intelligent transportation systems apply up to date methods of information technology to the current information on the infrastructure. An intelligent transportation system refers to any type of information technology applied to transport infrastructure and vehicles for the purpose of improving transportation. An advanced traveler information system is perhaps the most recognized subgroup of intelligent transportation systems. An advanced traveler information system aims to provide travelers with updated information about the road network conditions with the expectation that an informed traveler makes better decisions, ultimately alleviating traffic congestion. The advanced traveler information systems have the greatest value during dynamic traffic conditions.

It is clear that real-life traffic never evolves according to an *a priori* or static scheme. As advanced traveler information systems and intelligent vehicle navigation systems are being increasingly used, tackling routing problems with realistic assumptions becomes increasingly important. A great deal of real-time traffic information is available to the drivers courtesy of the advances in information technology. This information can provide drivers with turn-by-turn directions and more realistic travel times. In this regard, Ardakani and Sun [2] proposed an adaptive approach to tackle the continuous dynamic shortest path problem. They developed a decremental algorithm to

* Corresponding author at: Business Systems and Analytics Department, Lindback Distinguished Chair of Information Systems and Decision Sciences, La Salle University, Philadelphia, PA 19141, USA. Tel.: +1 215 951 1129; fax: +1 267 295 2854.
E-mail addresses: Ardakam@farmingdale.edu (M.K. Ardakani), tavana@lasalle.edu (M. Tavana).
URL: http://tavana.us/ (M. Tavana).

http://dx.doi.org/10.1016/j.measurement.2014.10.014
0263-2241/

reduce optimization time and evaluated the impact of the proposed adaptive routing and the performance of the decremental approach in static and dynamic networks under different traffic conditions.

This article has two main objectives: (1) to apply the A^* algorithm in the decremental approach to speed-up the optimization of the dynamic shortest path problem; and (2) to utilize multidimensional scaling to convert costs (times) into distances. In other words, the main contributions of this study are: (1) employing the A^* algorithm in decremental algorithm to boost up the optimization speed; (2) applying the weighted metric multidimensional scaling technique to define the potential function of the A^* algorithm. Here, we assume that travelers know the realizations of all link travel times up to the current time. Also, link travel times change in an unpredictable fashion and the first-in-first-out (FIFO) property or other simplifications in link travel times are not applicable. These assumptions are particularly pertinent to road networks especially in the presence of intelligent vehicle navigation systems.

The paper is organized as follows. In Section 2, the literature and background on transportation routing networks is presented. A formal description of the problem and the proposed approach are described in Section 3. Section 4 discusses the weighted metric multidimensional scaling and its application in using the A^* algorithm. Section 5 demonstrates experimental results and comparisons. Concluding remarks are summarized in Section 6.

2. Literature review

Shortest path problems are classical network optimization problems with many applications such as communication networks, transportation systems, and computer networks among others. Several authors have studied the static shortest path problems in transportation networks. More recently, Bast et al. [6] reported on an extensive review of the transportation routing literature with emphasis on contemporary algorithms. They categorize these algorithms into basic techniques [16]; goal-directed techniques [21,18]; separator-based techniques [14,30]); hierarchical techniques [20,17]; bounded-hop techniques [13,1]; and combination techniques [19,8,15].

The dynamic (time-dependent) shortest path problem is a generalized shortest path problem with network characteristics such as link cost changes over time. Time dependencies can be represented by travel time functions on the links. In other words, the cost of a link depends on the time with which it is traversed. The dynamic shortest path problems can be categorized based on their characteristics such as discrete/continuous representation of time, overtaking/non-overtaking notion, and waiting/no-waiting time at nodes. These assumptions are usually incorporated either to properly represent the problem or to reduce the complexity of the problem. The non-overtaking assumption, FIFO, states that later departures cannot lead to earlier arrivals. Sometimes, if the FIFO condition is not met, the problem might be simplified by allowing waiting (resting) at nodes before the next departure. Also, sometimes, the FIFO property might be justified by assuming that each road lane is a separate arc and vehicles cannot overtake each other in such networks. However, due to advances in infrastructure and transportation technology, fewer assumptions should be considered in network optimization; e.g., vehicles can maneuver freely on roads. Orda and Rom [27] considered a general analysis of time-dependent shortest paths under several waiting constraints. They showed that the shortest-path problem can be efficiently solved if travel time functions are nonnegative and follow the FIFO assumption. In addition, they showed that the non-FIFO assumption cannot be solved polynomially since it makes the problem at least NP-hard. A fair body of research such as Nannicini [25], Delling [12], and Batz et al. [7] can be found on the speed-up time-dependent shortest path problems requiring the FIFO property.

The discrete-time dynamic shortest path problems have been applied especially in public transit networks, which are inherently time-dependent. In public transit networks, certain parts of the network need to be traversed at particular, discrete points in time (see Müller-Hannemann et al. [24] for more details). In continuous-time dynamic networks, time is treated as real numbers. Accordingly, a timetable, which is usually used as the input to the time-dependent algorithms, cannot be constructed unless time is discretized. Another approach for dealing with the continuous-time dynamic problems is to approximate the time function with a piecewise linear function which still requires the FIFO property. For instance, Dean [11] provided a good review of the continuous-time dynamic shortest path problems and proposed the linear approximation of link travel times with the FIFO assumption. He also considered a more general setting. The FIFO assumption makes the problem less practical in real road traffic networks. Although the FIFO property is useful for systems such as train networks, it is not applicable for road transportation where overtaking occurs frequently. The continuous-time dynamic network has been primarily used in the network flow problems [23]. The problem formulated here is related to the continuous-time dynamic shortest path problem with the assumption of unpredictable link travel times where the link travel times cannot be discretized or piecewise approximated.

The majority of the research in transportation networks assumes that changes occur in a predictable way with the non-overtaking assumption. However, by virtue of advances in technology, particularly during the use of intelligent vehicle navigations using advanced traveler information systems, time-dependent cost functions with frequent, instantaneous, and sometimes-unpredictable changes need to be taken into consideration. In accordance with the adaptive routing notion, these assumptions require solving a series of static shortest path problems. The adaptive approach adopted here is developed by Ardakani and Sun [2]. It allows incorporating the most general travel time functions and relaxing the non-overtaking constraint in the continuous-time dynamic shortest path problem. It should be mentioned that although the decremental approach is not guaranteed to find the optimal routes, it provides a fast computation speed with negligible cost increases.

3. Adaptive routing and decremental algorithm

A road network is defined by a graph where links represent road segments and nodes signify intersections with an origin O and a destination D. A random network $G = (N, A, P)$ is described by a triple notation consisting of a set of nodes N ($|N| = n$), a set of links A ($|A| = m$), and a link characterization P specifying the mean link travel times, which are time-dependent random variables. This way, the time dependency of travel times is considered in the problem. The network can be assumed directed or undirected depending on the topology of the road network. For example, a one-way street can be represented in a direct network. The general setting assumes a single-vehicle origin and destination. However, due to the overtaking assumption, driver B can arrive at the destination node before driver A even if driver B leaves the origin node later. This assumption relaxes the FIFO constraint. $\mu_{jk}(t)$ is the mean travel time (cost) of link j_k at time t. It is assumed that decisions are made at nodes. The decision determines what link must be chosen next at each node based on the current state $x = \{j, t, I\}$ where j is the current node, t is the current time, and I is the current information. The current information I contains the mean link travel times. The mean travel time from node j to node k, $\mu_{jk}(t)$, is assumed to be available at any time. Note that $\mu_{jk}(t)$ is not probabilistic. It is a time-dependent function representing the expected travel time from node j to node k. For instance, $\mu_{12}(10{:}30\ \text{a.m.}) = 5$ indicates that the expected travel time from node 1 to 2 at 10:30 a.m. is 5 min. It is assumed that an intelligent vehicle navigation system estimates and provides the expected travel times for any given link at any time. Also, in the routing decision process, the origin node changes adaptively. In other words, the current node j is also the origin node when determining the shortest path between j and the destination node using dynamic optimization.

Fig. 1 illustrates a hypothetical network with six nodes (i.e., $|N| = 6$) and eight links (i.e., $|A| = 8$). There are four paths for this network from the origin node 0 to the destination node 5: path 1 = 0_1_3_5, path 2 = 0_1_5, path 3 = 0_2_4_5, and path 4 = 0_2_5. The left figure shows the expected link travel times in minutes at $t = 0$. The dashed lines indicate optimal paths. Path 0_1_3_5 is suggested as the shortest path according to the link travel times available at $t = 0$. Following the suggested path, the driver arrives at node 1 after t', although 12 min was the expected time for link 0_1. In practice, this discrepancy is commonly due to new traffic situations and/or inaccurate link travel time estimates. Now, consider the right network demonstrating the new position of the driver at t'. The driver needs to find the optimal path from 1 to 5 according to the newly updated link travel times. For example, the travel time for link 1_5 is $\mu_{15}(t') = 29$, which is different from $\mu_{15}(0) = 30$. Based on updated link travel times at t', 1_5 is chosen as a new optimal path. The reason 1_3_5 was not considered as the optimal path could be due to a congestion occurrence on the link 3_5 within t'. Note that anticipatory information is not available. In other words, the shortest path is based on currently-observed travel times, and does not account for the changes that may occur between the present moment and the time a traveller actually arrives at the next node. It emphasizes the distinction between *instantaneous* and *experienced* travel times (see Yildirimoglu and Geroliminis [31] for more details).

This adaptive routing scheme can be expressed by a recursive equation. The following dynamic problem is defined for a given node j.

$$k^* = \arg\min\left\{g(k) + \mu_{jk}(t) | \forall k \in B(j)\ \&\ x\right\}, \tag{1}$$

where $g(k)$ is the expected travel time from node k to the destination node D; the boundary condition is $g(D) = 0$. $B(j)$ represents the outgoing adjacent nodes from node j, that is, $B(j) = \{(j, k) \in A;\ k \in N\}$. x is the current state containing information I. The decision is made at node j to determine which adjacent node k must be selected to have a minimum total travel time from node j to the destination. There is no rest time assumed at nodes; therefore, arriving and departing times are identical. Here, the goal is to heuristically solve a time-dependent shortest path problem using a dynamic shortest path algorithm.

In order to find the shortest path from the current node to the destination, a straightforward approach is to simply update all of the links by arriving at each node and then applying a single-pair algorithm. This approach requires updating all the links and optimizing the whole network from scratch. Instead, we herein advocate another approach decrementing the network size at each node prior to applying a single-pair algorithm. As the driver travels toward the destination, the network size becomes smaller, and consequently the optimization process becomes faster. The proposed decremental approach is capable of reducing the network size by deleting noncritical nodes. Noncritical refers to nodes that are unlikely to be part of the optimal route. The logic behind the decremental approach is that the chance of finding better routes through backward nodes is small and drivers try to travel toward the destination located at the end of the x-axis.

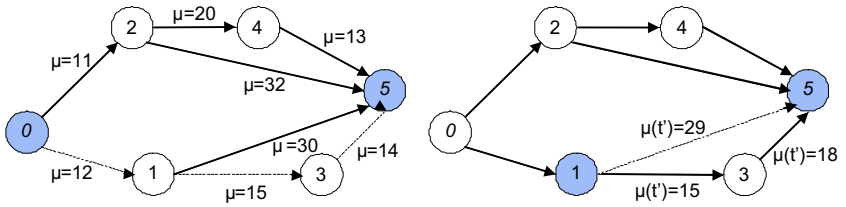

Fig. 1. Two consecutive steps of an adaptive routing algorithm.

The decremental approach proposed by Ardakani and Sun [2] consists of the following notable features:

i. *Node labeling* is based on the Cartesian coordinate system; x (or y) axis. The origin node label is 1 and the destination node label is n (sorted topology).
ii. *Delete* (j) deletes nodes whose labels are smaller than the current node j.
iii. *Update* (j, k, t) returns updated costs (link travel times) from the current node j to node k at time t; k belongs to the decremented network.
iv. *Path* (j) finds the shortest path from the current node j to the destination.
v. Traverse to the next node and repeat from *ii*.

In *Path* (j), the decremented network is optimized to find the shortest path. Any classical or speed-up algorithm can be utilized in this step. Optimization algorithms applied in *Path* (j) directly affect the speed of the decremental approach. Ardakani and Sun [2] evaluate Dijkstra's algorithm. Here, a variant of the A^* algorithm is utilized for speed-up purposes and compared to Dijkstra's algorithm.

The idea of a goal-directed technique such as the A^* search [21] is to push the search toward the destination. Dijkstra's algorithm is a special case of A^* when reduced costs are considered. The A^* search removes nodes based on a defined priority. The priority key for removing a node v is made up of two parts: the length of the tentative shortest path from the origin to node v (as in Dijkstra's algorithm), and an underestimation of the distance to reach the destination from node v. The function that gives priority to nodes and estimates the distance between a node and the destination is called the potential function. The A^* algorithm searches no more nodes than Dijkstra's algorithm, particularly if the potential function significantly underestimates distances to the destination. Different algorithms have been embedded or developed based on the A^* concept to speed-up or improve shortest path problems. For instance, the A^* search has been applied to the time-dependent model on public transit networks [28]; multimodal journey planning [5]; obtaining bounds on the search space size [29]; and speed-up [26]. The following statistical technique is adopted here to define the potential function based on the reconstructed network.

4. Weighted metric multidimensional scaling

Multidimensional scaling is a statistical technique that represents measurements of dissimilarity (or similarity) of objects as distances between points most likely in a low dimensional space [9]. Multidimensional scaling has its origins in psychometric studies and obtained popularity in science and engineering. In transportation, there are several usages of multidimensional scaling particularly concentrating on road network visualizations [10,22]. Multidimensional scaling can be categorized into classical scaling, metric scaling, nonmetric scaling, and generalized scaling. In regular classical scaling, also known as Torgerson–Gower scaling, the input are dissimilarities between pairs of items, weights are not allowed, and Strain is chosen as a loss function (see [9]. If dissimilarities are measured in metric such as distances, more flexible loss functions and weights can be employed. In non-metric scaling, the data are at the ordinal level of measurement. Generalized scaling, which might also be categorized in metric scaling, is recently developed and mainly used to recognize deformable objects.

In order to fit the data in a new configuration (structure), a loss function needs to be minimized. Borg and Groenen [9, pp. 253] mention the following general loss function:

$$\sigma(\mathbf{X}) = \sum_{i<j} w_{ij}\left[f(\delta_{ij}^2) - f(d_{ij}^2(\mathbf{X}))\right]^2 \quad (2)$$

This loss function sums the differences of squared dissimilarities δ_{ij}^2, and squared distances d_{ij}^2 of pairs ij. $f(z)$ is an increasing scalar function; for instance, $f(z)$ equal to $\sqrt{z}$, z, and $log(z)$ are known as Stress, SStress, and Multiscale loss functions, respectively. $\mathbf{X}$ denotes a new configuration of points, which is the coordinates of nodes in the reconstructed network. Also, link travel costs are assumed as dissimilarities. In multidimensional scaling, weights play important roles such as underlining different aspects of the data, or taking the reliability of the data into account. For instance, $w_{ij} = 1$ indicates that for pair ij dissimilarity is observed, whereas $w_{ij} = 0$ can be used when dissimilarity is missing (undefined δ_{ij}). The missing data are excluded in optimization. If there is no link between node i and node j, w_{ij} is considered to be a missing value and is set equal to zero.

Weighted metric multidimensional scaling is applied to form a new network whose link lengths are equal to the corresponding costs (times) in the original network. In other words, the basic idea is to find the coordinates of nodes given link travel costs. Also, dimension reduction is not considered here; thus, the data remain in two dimensions. The approach is analogous to laying out a map for a city if the only available information is the distances between all pairs of intersections. After the new network is formed using weighted metric multidimensional scaling, the potential function in the A^* algorithm can be defined based on the Euclidian distance between a node and the destination in the reconstructed network.

The following example shows how weighted metric multidimensional scaling can be applied to reconstruct a network. Fig. 2 demonstrates a random network with 10 nodes. The optimal route with the minimum distance from the origin node to the destination is 1_3_7_10 with the total distance of 243.7 + 281.7 + 252.7 = 778.1.

Now, suppose distances in Fig. 2 change to costs and a visualization of a network whose link lengths are costs is assumed. Fig. 3 presents two reconstructed networks after applying the weighted metric multidimensional scaling technique. In the left figure, the link costs are 1.5 times of distances for all links, which leads to the optimal distance of 1.5×778.1. In the right figure, the costs of links 1_2 and 9_10 are twice as much as their distances and the rest of the costs are identical to their corresponding distances. The optimal path has the distance of 778.1. Note that in the reconstructed networks, link lengths represent

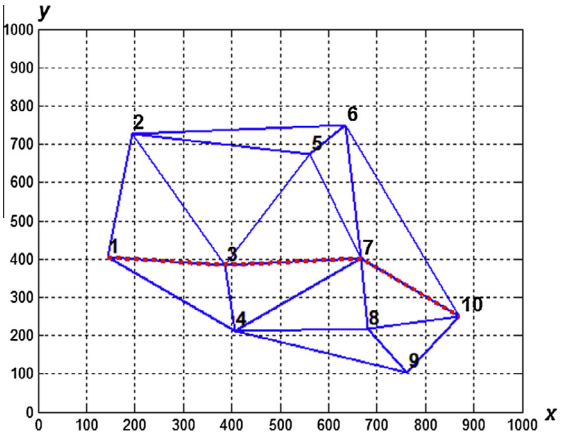

Fig. 2. A random network with 10 nodes and 21 links along with its optimal route.

the desired costs, although the node coordinates are different from Fig. 2.

In simulation, weighted metric multidimensional scaling is utilized using the *mdscale* command in the MATLAB statistics toolbox. It assumes the traveling cost matrix as input and generates a new configuration of nodes. The two-dimensional link cost matrix is assumed to be the dissimilarity matrix. The command treats NaN's as missing values, and ignores them in the minimization of the loss function. The metric scaling criterion *metricsstress* is set to apply the SStress loss function that is normalized with the sum of 4th powers of the dissimilarities. The initial node locations are considered to be starting points in optimization. The outputs are the Cartesian coordinates of nodes and the optimal value of the SStress loss function.

5. Simulation case study

This section evaluates the decremental approach using the A^* algorithm in different dynamic networks. It is assumed that traffic information is fully available up to the current time. The most updated information is utilized at each node to make a decision and choose the next link. Thus, at each node, the shortest path problem is executed using the last updated information. It is assumed that link cost functions are contingent upon time and link length. The following quadratic functions are used to generate the mean travel times (costs). This time-dependent function and its parameters are arbitrarily chosen and can be in other forms. These functions do not affect the computational aspects of algorithms and only concerns the accuracy of estimated travel times. Note that in a real situation there is no need of assuming these functions since the real values are provided by the intelligent navigation system. Nonetheless, the sensitivity analyses of estimated travel times are performed in Ardakani and Sun [2].

$$\mu_{ij}(t, d_{ij}) = d_{ij}\left(-\frac{U}{(TR)^2}t^2 + \frac{2U}{TR}t + 1\right), \tag{3}$$

where U and R are random variables following uniform distributions with range of $(-0.3, 0.3)$ and $(0.5, 1)$, respectively to generate different quadratic functions. The d_{ij} is the distance between node i and node j. Also, T is a parameter adjusting travel times according to the length of the network, which is here assumed to be 1000.

In simulation, networks are randomly generated in a two-dimensional Cartesian system. Random points representing nodes are generated in a predefined area of 1000×1000. In order to generate links, Delaunay triangulations using random points as vertices are employed. Node labeling is formed based on sorted nodes along the x-axis. As a result, the origin and destination nodes are labeled 1 and n, which have minimum and maximum values on the x-axis respectively (e.g., see Fig. 2). All analyses and simulations are conducted using the MATLAB software with a Dual-Core 3.0 GHz CPU. It should be mentioned that using C++ might increase the optimization speed; particularly if we do not call a series of built-in functions in MATLAB. However, we are more interested in relative comparisons regardless of language programing.

At each run, two approaches are considered before applying A^* or Dijkstra's algorithm. The first approach applies all nodes with the original network size. The second one, hereafter called Decremental, utilizes the

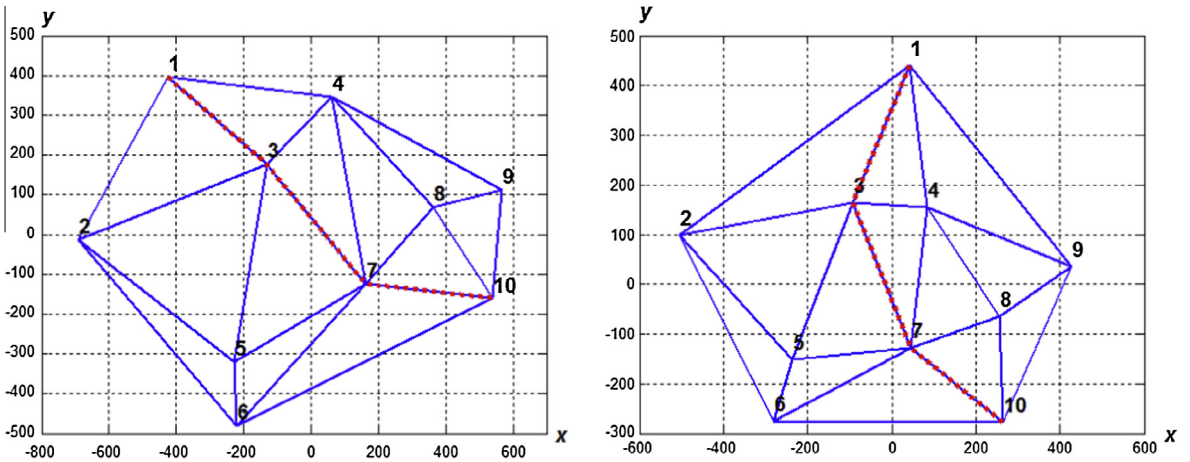

Fig. 3. Two reconstructed networks using weighted metric multidimensional scaling.

following method to reduce the network size. According to the decremental approach, all nodes whose labels are smaller than the current label are removed from the network. Thus, four approaches, namely Dijkstra (D), Decremental-Dijkstra (DD), A^*, and Decremental-A^* (DA^*), are considered to optimize the adaptive routing problem. The Dijkstra approach applies Dijkstra's algorithm in a dynamic network using the most updated data at each node to select the next link. The Decremental-Dijkstra approach is similar to the Dijkstra approach except that it implements the decremental approach to reduce the network size before applying Dijkstra's algorithm. The A^* approach uses the weighted multidimensional scaling technique to define the potential function for the whole network and optimizes the network based on prioritized nodes. The Decremental-A^* approach utilizes the A^* algorithm, after it reduces the network size by applying the decremental approach at each iteration. Note that since it is assumed that no waiting time is allowed at the nodes, the arriving and departing times are the same. Also, the weighted multidimensional scaling technique is re-run at each iteration to update the potential functions to employ the A^* algorithm.

Different networks with sizes of 500, 1000, 1500, 2000, 2500, and 3000 nodes are selected. In order to increase the reliability of results, each set of simulations is executed 11 times. For instance, Table 1 presents 11 runs of random networks with size of $n = 3000$ in the Decremental-A^* approach. It illustrates runs that are sorted based on the CPU time, number of links in network (m), number of traversed links on optimal routes, CPU times in second scale to find the optimal routes, the optimal values of the SStress loss function, and total cost. Note that setup times such as matrix preparations and other initializations are not included in the CPU times. They just signify the required times to find the optimal routes by applying the shortest path algorithms with the available inputs.

Further analyses are based on the median of CPU times chosen from 11 random samples. For instance, in Table 1 2.15 is the median of the CPU times; accordingly, the 9th sample is selected to represent the results for the Decremental-A^* approach with a size of 3000. This network has 8980 links and the optimal route is obtained by traversing 52 links from the origin node to the destination node with the total cost of 1116.4. The SStress value is approximately zero, which indicates that the loss function is properly minimized in the weighted multidimensional scaling technique. This column indicates that in all runs, the new networks are accurately reconstructed.

Fig. 4 displays 24 median sample runs obtained from four approaches and six networks. It is assumed that the median sample runs can represent the corresponding networks. Fig. 4 indicates that the decremental approach reduces the CPU times significantly and that the A^* algorithm is superior to Dijkstra's algorithm in all cases. However, reconstructing the network to define the potential function may be expensive either due to preprocessing time or memory consumption, depending on the link travel times and the topology of the network. Also, Fig. 4 shows that regardless of network size, the performance of Decremental-A^*, A^*, Decremental-Dijkstra, and Dijkstra approaches are in decreasing order, i.e., $DA^* > A^* > DD > D$.

CPU time reductions illustrated in Fig. 4 are quantitatively compared in Table 2. For instance, for the network with 3000 nodes, A^* and Decremental-A^* approaches require 4.07 and 2.15 seconds CPU time, respectively. Thus, compared to A^*, DA^* has the advantage of $((4.07 - 2.15)/4.07)100\% = 47\%$ reduction in CPU time, which is shown in column $DA^* > A^*$. As a whole, the results in each column indicate quite homogeneous reductions in different networks. The last row shows the average reductions of six networks. The columns $DA^* > A^*$ and $DD > D$ indicate that the decremental approach saves approximately 40–45% of CPU time when compared to its non-decremental counterparts. Also, columns $DA^* > DD$ and $A^* > D$ signify about 70% CPU time reduction for A^* in comparison to the Dijkstra approach. Moreover, A^* is superior to the Decremental-Dijkstra approach; note that $A^* > DD$. As is specified in $DA^* > D$, the Decremental-A^* approach outperforms the Dijkstra approach.

The decremental approach decreases the number of evaluated links, which eventually leads to CPU time reductions. Fig. 5 illustrates CPU times and the number of evaluated links in detail for networks with $n = 3000$. The x-axis represents iterations or traversed nodes. The left figure shows that the CPU times in the Decremental approaches are less than their non-decremental counterparts. Also, A^* speeds up the optimization time significantly. The right figure exemplifies the decreasing number of evaluated links using the decremental approach. The number of evaluated links is always about

Table 1
The Decremental-A^* approach for 11 random networks with $n = 3000$.

Run	m	Traversed links	CPU time (s)	SStress	Cost
6	8975	50	1.13	2.87×10^{-15}	1070.3
2	8979	53	1.82	2.72×10^{-15}	1183.4
4	8974	50	1.91	3.38×10^{-14}	1052.3
11	8974	55	1.98	2.97×10^{-15}	1065.4
1	8973	58	2.03	3.26×10^{-15}	1115.8
9	8980	52	2.15	2.10×10^{-15}	1116.4
3	8979	56	2.16	1.15×10^{-15}	1144.0
10	8977	56	2.29	2.35×10^{-15}	1057.4
8	8969	56	2.34	2.52×10^{-15}	1118.2
7	8976	65	2.5	4.33×10^{-15}	1187.7
5	8977	60	3.13	1.06×10^{-15}	1244.5

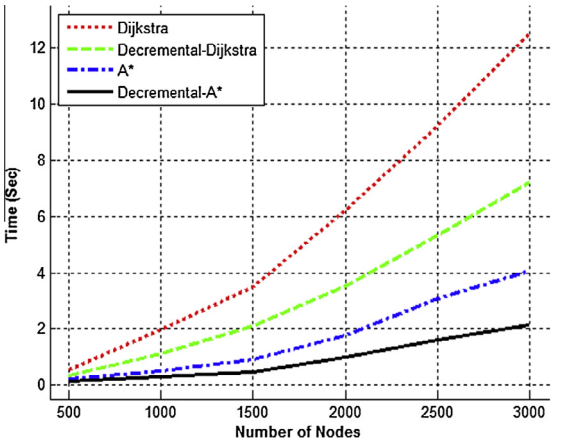

Fig. 4. CPU time of median sample runs in four approaches.

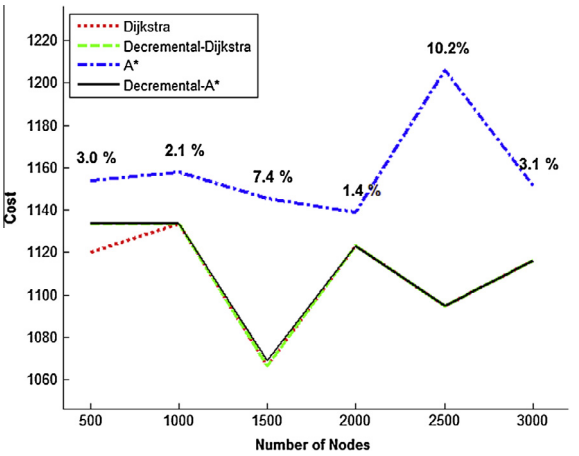

Fig. 6. Corresponding costs in Fig. 4.

Table 2
Rounded CPU time reduction percentages.

N	$DA^* > A^*$	$DA^* > DD$	$DA^* > D$	$A^* > DD$	$A^* > D$	$DD > D$
500	39	66	78	43	64	37
1000	44	75	86	55	74	43
1500	48	78	87	57	74	40
2000	43	71	84	50	71	43
2500	47	70	82	42	67	42
3000	47	70	83	44	67	42
Average	45	72	83	49	70	41

9000 for the non-decremental approaches. As a whole, CPU times and evaluated links favor the Decremental-A^* approach consuming the lowest CPU times.

Another issue that should be addressed is the quality of obtained optimal routes. Fig. 6 compares the total travel costs using Dijkstra (D), Decremental-Dijkstra (DD), A^*, and Decremental-A^* (DA^*) approaches. The costs are chosen from the median samples sorted based on the CPU time improvements. The costs of optimal routes obtained from A^* are compared to the costs of the Dijkstra approach. The percentage amount of cost increases are shown on the plot. For instance, for the network with size of 500 nodes, the optimal costs using the A^* approach and the Dijkstra approach are 1154.2 and 1120.1, respectively, which leads to $(1154.2 - 1120.1)/1120.1 = 3\%$ additional cost. Fig. 6 signifies that roughly a 5% increase in total cost occurs when the A^* approach is used. The optimal routes have the same costs in both the Decremental-Dijkstra and Decremental-A^* approaches. Note that costs using these two approaches are similar to costs using the Dijkstra approach except when $n = 500$. It indicates that the decremental approach is not guaranteed to find the optimal routes. However, this occurs rarely, and when it does occur, cost differences are not significant. Table 3 demonstrates traversed routes in detail to better explain the cost discrepancies.

Table 3 illustrates the optimal routes and the associated costs for a random run with $n = 500$. The last column shows the associated costs of the corresponding approaches displayed in the first column. The first row shows the number of required iterations to reach the destination node 500 from the origin node 1. At each iteration, the shortest path from the current node to the destination

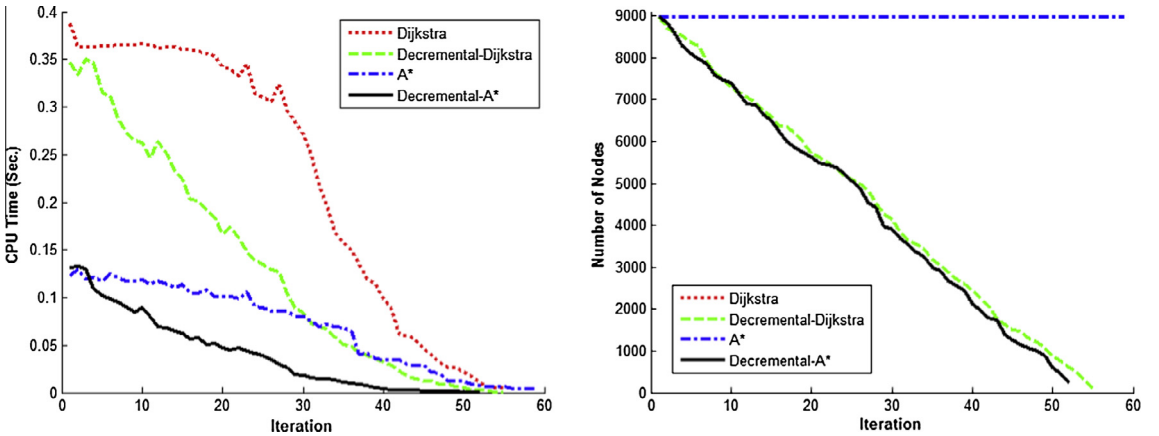

Fig. 5. CPU times (left) and evaluated links (right) with $n = 3000$.

Table 3
Traversed nodes and associated total costs for a random run with $n = 500$.

Iterations	1	2	3	4	5	6	7	8	9	10	...	20	21	22	23	Cost
D	1	35	55	82	102	118	142	141	187	215	...	485	498			1120.1
DD	1	35	55	82	102	118	142	179	212	239	...	485	498			1133.7
A^*	1	35	55	82	102	118	142	125	139	141	...	430	473	485	498	1154.2
DA^*	1	35	55	82	102	118	142	179	212	239	...	485	498			1133.7

node is executed. All approaches traverse 21 nodes on their optimal routes to reach the destination except the A^* approach which requires 23 iterations.

The key point of route differences lies in the direction of traveling against or toward the destination located at the last point of the *x*-axis. Recall that nodes are labeled based on their locations on the *x*-axis coordinate. Generally, it is expected that traveling against the destination does not reduce the travel cost. However, in rare occasions, as is observed in Dijkstra approach, re-routing at the 7th iteration from node 142 to node 141 and then traversing a different route leads to a lower travel cost, 1120.1. The traversed nodes are identical in the Decremental-Dijkstra and Decremental-A^* approaches. The A^* approach tends to have more backward traversing; e.g., see re-routing from node 142 to node 125 at the 7th iteration. This occurs because of the poor potential function obtained from an imprecise reconstructed network. To be more specific, the link lengths in the reconstructed network do not represent the cost links in the original network. In other words, inaccurate results obtained from the weighted multidimensional scaling technique lead to inaccurate coordinates of nodes, which eventually cause the backward tendency. It also explains why the cost of the A^* algorithm in Fig. 6 is higher. This shortcoming can be remarkably improved by embedding the A^* approach in the decremental approach. Since the decremental approach removes nodes whose labels are smaller than the current node, it eliminates all backward possibilities. However, the optimality of routes still depends on the accuracy of node coordinates in the reconstructed network.

6. Conclusion and future research directions

This study addresses a real and challenging problem that will become increasingly important as more advanced traveler information systems are employed. In this respect, the adaptive routing approach with practical assumptions is utilized in continuous-time dynamic shortest path problem. The adopted assumptions consider general travel time functions and relax the non-overtaking constraint. Also, the decremental approach deleting noncritical nodes is employed to reduce the network size. The decremental approach is heuristic and is not guaranteed to find the optimal routes. However, as the results confirm, it provides an appropriate trade-off between CPU time reductions and cost increases. The adaptive routing along with the decremental approach can be applied in an intelligent vehicle navigation system to provide drivers with turn-by-turn directions, minimum travel times, and more realistic estimates.

The A^* algorithm is embedded in the decremental approach to speed-up the optimization of the shortest path problem. Moreover, the weighted multidimensional scaling technique is employed to define the potential function in the A^* algorithm. It reconstructs a network whose link lengths are equivalent to travel costs. The A^* and Dijkstra's algorithms are applied in the decremental approach to find the shortest paths in continuous-time dynamic networks. As a result, four approaches, namely, Dijkstra, Decremental-Dijkstra, A^*, and Decremental-A^*, are formed. Their performances are then evaluated in time-dependent networks with six different sizes. The simulation results indicate that using the decremental approach significantly decreases the average of evaluated links, which consequently reduces CPU times up to 45%. Also, the A^* approach reduces CPU time significantly (about 70%) and even outperforms the Decremental-Dijkstra approach.

In addition to CPU time, the performance of the different approaches is examined with respect to the quality of obtained routes. The total travel cost is chosen to measure this feature. The simulation results indicate that the Dijkstra, Decremental-Dijkstra, and Decremental-A^* approaches have almost the same performances. The costs of the optimal routes in A^* are roughly 5% higher than costs in the other three approaches. The cost increase is due to the inaccurate potential function obtained from an imprecise reconstructed network. This shortcoming cannot impact the Decremental-A^* approach since the decremental approach eliminates backward traversing. As a whole, among the four evaluated approaches, the Decremental-A^* approach, which consumes the least CPU time and achieves relatively high quality routes, is recommended for the defined problem.

We are currently considering several extensions of the method proposed in this study:

Travel cost: As is discussed in Section 3 with regards to experienced and instantaneous travel times, the total cost may increase due to later variations in link travel times; conceivably, more variations would be expected farther from the destination. Thus, the methods using perfect real-time information with no prediction ability have a limitation in this situation. In order to alleviate this shortcoming, it is helpful to develop techniques to predict link travel times utilizing historical data, which improves the accuracy of travel time estimates and alleviate the suboptimality of routes.

Different topology: In simulation, networks are randomly generated in a two-dimensional Cartesian system. Delaunay triangulations using random points as vertices are employed to generate links. Although it is common to generate random networks using Delaunay triangulations for computational purposes, it is recommended to test

other topologies, especially ones representing the real transportation networks. As a result, the proposed decremental approach needs to be evaluated using new topologies.

Multi-objective optimization: Algorithm designs in routing optimization for the minimum expected travel time problems are different from those where variance is taken into consideration. In multi-objective optimization, the standard deviation and mean link travel times can be treated as two individual objective functions. The goal is then to minimize these two objective functions simultaneously. Accordingly, instead of optimal routes, Pareto optimal routes need to be obtained. Ardakani and Wulff [3] briefly review multi-objective decision making for multi-response optimization problems. Those techniques can also be applied in this subject. In case of constructing responses, measurement errors should be taken into account [4].

Evolutionary algorithms: In simulations, the scale of optimization is a network with 3000 nodes. At this scale, any traditional optimization method seems appropriate. However, in real world problems or large scale networks, applying evolutionary approaches can provide significant advantages regarding the computational aspect of optimization. For instance, evolutionary approaches can be employed in shortest path optimization or multidimensional scaling optimization. Another justification to use evolutionary algorithms is to include the time required for optimization of multidimensional scaling. Recall that estimating the potential function (i.e., reconstructing the network) may be expensive either due to preprocessing time or memory consumption depending upon the link travel times and the topology of the network.

Acknowledgement

The authors would like to thank the anonymous reviewers and the editor for their insightful comments and suggestions.

References

[1] T. Akiba, Y. Iwata, K. Kawarabayashi, Y. Kawata, Fast shortest-path distance queries on road networks by pruned highway labeling, in: Proceedings of the 16th Meeting on Algorithm Engineering and Experiments (ALENEX'14), SIAM, 2014, pp. 147–154.
[2] M.K. Ardakani, L. Sun, Decremental algorithm for adaptive routing incorporating advanced traveler information, Comput. Oper. Res. 39 (12) (2012) 3012–3020.
[3] M.K. Ardakani, S.S. Wulff, An overview of optimization formulations for multi-response surface problems, Qual. Reliab. Eng. Int. 29 (1) (2013) 3–16.
[4] M.K. Ardakani, S.S. Wulff, D. Das, T.J. Robinson, Estimation in second-order models with errors in the factor levels, Commun. Statist.: Theory Methods 40 (9) (2011) 1573–1590.
[5] C. Barrett, K. Bisset, M. Holzer, G. Konjevod, M.V. Marathe, D. Wagner, Engineering label-constrained shortest-path algorithms, in: The Shortest Path Problem: Ninth DIMACS Implementation Challenge, vol. 74 of DIMACS Book, American Mathematical Society, 2009, pp. 309–319.
[6] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, R. Werneck, Route Planning in Transportation Networks, Technical Report: Microsoft, 2014.
[7] G.V. Batz, R. Geisberger, P. Sanders, C. Vetter, Minimum time-dependent travel times with contraction hierarchies, ACM J. Exp. Algorithmics 18 (1.4) (2013) 1–43.
[8] R. Bauer, D. Delling, P. Sanders, D. Schieferdecker, D. Schultes, D. Wagner, Combining hierarchical and goal-directed speed-up techniques for Dijkstra's algorithm, ACM J. Exp. Algorithmics 15 (2.3) (2010) 1–31. Special Section devoted to WEA'08.
[9] I. Borg, P. Groenen, Modern Multidimensional Scaling: Theory and Applications, second ed., Springer, New York, 2005.
[10] U. Brandes, F. Schulz, D. Wagner, T. Willhalm, Generating node coordinates for shortest path computations in transportation networks, ACM J. Exp. Algorithmics 9 (1) (2004) 1–16.
[11] B. Dean, Continuous-Time Dynamic Shortest Path Algorithms, Master Thesis, Massachusetts Institute of Technology, MA, 1999.
[12] D. Delling, Time-dependent SHARC routing, Algorithmica 60 (1) (2011) 60–94.
[13] D. Delling, A.V. Goldberg, A. Nowatzyk, R.F. Werneck, PHAST: hardware-accelerated shortest path trees, J. Parallel Distrib. Comput. 73 (7) (2013) 940–952.
[14] D. Delling, A.V. Goldberg, I. Razenshteyn, R.F. Werneck, Graph partitioning with natural cuts, in: 25th International Parallel and Distributed Processing Symposium (IPDPS'11), IEEE Computer Society, 2011, pp. 1135–1146.
[15] D. Delling, G. Nannicini, Core routing on dynamic time-dependent road networks, Informs J. Comput. 24 (2) (2012) 187–201.
[16] E.W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1959) 269–271.
[17] L. Fu, D. Sun, L.R. Rilett, Heuristic shortest path algorithms for transportation applications: state of the art, Comput. Oper. Res. 33 (11) (2006) 3324–3343.
[18] A.V. Goldberg, C. Harrelson, Computing the shortest path: A^* search meets graph theory, in: Proceedings of the 16th Annual ACM (SODA'05), 2005, pp. 156–165.
[19] A.V. Goldberg, H. Kaplan, R.F. Werneck, Reach for A^*: shortest path algorithms with preprocessing, In the Shortest Path Problem: Ninth DIMACS Implementation Challenge, AMS, 2009, pp. 93–139.
[20] R.J. Gutman, Reach-based routing: a new approach to shortest path algorithms optimized for road networks, in: Proceedings of the 6th Workshop on Algorithm Engineering and Experiments (ALENEX'04), SIAM, 2004, pp. 100–111.
[21] E. Hart, N. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Trans. Syst., Sci. Cybernetics 4 (2) (1968) 100–107.
[22] C. Kaiser, F. Walsh, C.J.Q. Farmer, A. Pozdnoukhov, User-centric time-distance representation of road networks, in: Proceedings of the 6th International Conference GIScience, 2010, pp. 85–99.
[23] R. Koch, E. Nasrabadi, Flows over time in time-varying networks: optimality conditions and strong duality, Eur. J. Oper. Res. 237 (2) (2014) 580–589.
[24] M. Müller-Hannemann, F. Schulz, D. Wagner, C. Zaroliagis, Timetable information: models and algorithms, in: Algorithmic Methods for Railway Optimization, vol. 4359 of Lecture Notes in Computer Science, Springer, 2007, pp. 67–90.
[25] G. Nannicini, Point-to-Point Shortest Paths on Dynamic Time-Dependent Road Networks. Ph.D. Thesis, Ecole Polytechnique, Italy, 2009.
[26] G. Nannicini, D. Delling, D. Schultes, L. Liberti, Bidirectional A^* search for time-dependent fast paths, Networks 59 (2) (2012) 240–251.
[27] A. Orda, R. Rom, Shortest-path and minimum delay algorithms in networks with time-dependent edge-length, J. ACM 37 (3) (1990) 607–625.
[28] E. Pyrga, F. Schulz, D. Wagner, C. Zaroliagis, Efficient models for timetable information in public transportation systems, ACM J. Exp. Algorithmics 12 (24) (2008) 1–39.
[29] M. Rice, V. Tsotras, Bidirectional A^* search with additive approximation bounds, in: Proceedings of the 5th International Symposium on Combinatorial Search (SoCS'12), AAAI Press, 2012.
[30] P. Sanders, D. Schultes, Engineering highway hierarchies, ACM J. Exp. Algorithmics 17 (1) (2012) 1–40.
[31] M. Yildirimoglu, N. Geroliminis, Experienced travel time prediction for congested freeways, Transp. Res. Part B 53 (2013) 45–63.