ISSN 1 746-7233, England, UK
World Journal of Modelling and Simulation
Vol. 12 (2016) No. 1, pp. 15-23

Adaptive MDS-based algorithm for dynamic routing optimization in advanced traveler information systems

Mostafa K. Ardakani*

School of Engineering Technology, State University of New York, Farmingdale, NY, USA

(*Received October 11 2015, Accepted December 9, 2015*)

Abstract. By virtue of advances in technology, particularly in the presence of advanced traveler information systems (ATIS), dynamic (time-dependent) cost functions with frequent, instantaneous, and sometimes unpredictable changes need to be taken into account. An adaptive routing approach tackling this challenge is modeled and applied to find the shortest paths. A star (A^*) algorithm, which uses a best-first search, is also employed to speed up the optimization process. Moreover, in order to convert link cost functions to distances and construct the potential function in A^* algorithm, the weighted multidimensional scaling technique is utilized. Furthermore, the impact of A^* algorithm is evaluated and compared with Dijkstra's algorithm in different road networks.

Keywords: advanced traveler information system, A^* algorithm, dynamic transportation networks, dynamic shortest path, multidimensional scaling (MDS), real-time vehicle navigation

1 Introduction

Nowadays, traffic engineers are facing the challenge of alleviating traffic congestions. Traffic congestion in 2010 caused urban Americans to travel an additional 4.8 billion hours (equivalent to the time Americans spend relaxing and hiking in 10 weeks) and to consume an extra 1.9 billion gallons of fuel (equivalent to about 2 months of flow in the Alaska Pipeline) for a total congestion cost of $ 101 billion–an increase from $ 79 billion in 2000; and, this is only the amount of wasted time and fuel[29]. In order to relieve congestion, making best use of current infrastructure with advances in information technology is the underlying idea of intelligent transportation systems. Among the various sub-systems of intelligent transportation systems, advanced traveler information system aims to provide travelers with updated information about network conditions in hope that an informed traveler makes a better decision; consequently, better decisions would result in a relief from congestion. It is clear that real-life traffic never goes according to an a priori and static scheme.

This manuscript addresses a real and challenging problem that will become increasingly important as more advanced traveler information systems are deployed in large proportions. Generally, drivers have little knowledge of online traffic congestion on the paths they drive. Even when drivers use updated information, they normally do not know how to utilize that information to find better paths[23]. Accordingly, drivers often tend to drive sub-optimal routes. In order to lighten this problem, [5] proposed an adaptive routing approach using online information. The approach can be implemented in a vehicles navigation system employing the online traffic information to determine optimal turn-by-turn directions. Advanced traveler information systems can efficiently be utilized through the so-called intelligent vehicle navigation system. The manuscript deals with dynamic transportation networks. Here, the road network can be viewed as the shortest path problem considering travel times change dynamically. In this study, the weighted metric multidimensional scaling technique is utilized to convert travel times into distances. The result is subsequently applied to construct

* Corresponding author. *E-mail address*: Ardakam@farmingdale.edu

the potential function of A^* algorithm. The utilization of this technique in conjunction with the decremental approach can speed up the optimization process. The performance of the proposed methodology is evaluated through simulation.

The paper is organized as follows. In Section 2, dynamic (time-dependent) network literature review is presented. Section 3 provides a formal description of the adaptive routing and explains why the system is dynamic and how to tackle it. Section 4 applies the weighted multidimensional scaling technique in A^* algorithm to speed up the routing optimization. Section 5 demonstrates experimental results and comparisons. Conclusions and future works are presented in Section 6. In short, the paper outlines more realistic assumptions, models the problem in Section 3, provides an algorithm in Section 4 to solve the problem, and validates the statements in Section 5.

2 Literature review

The dynamic shortest path problem is a generalization of the shortest path problem assuming that network characteristics such as link costs change over time. Additional assumptions are usually considered to properly represent the problem or just reduce the complexity of the problem. An extensive review of the transportation routing literature with emphasis on contemporary algorithms has been reported in [9]. The dynamic shortest path problems can be categorized based on their characteristics such as discrete/ continuous representation of time, overtaking/ non-overtaking notion, waiting/ no-waiting time at nodes, etc. In continuous dynamic networks, time is treated as real numbers. [11] presented a detailed general framework for modeling and solving common variants of the discrete-time problem.

Perfect online information, which is particularly pertinent to transportation applications, assumes that travelers know the realizations of all link travel times up to the current time. In previous works such as [17], the optimal routing problem with perfect online information is studied. The value of information provided by the advanced traveler information systems is most evident when traffic conditions are not static. In this study, it is assumed that the perfect online information of link travel times is available up to the current time and decisions are made at nodes.

As the advanced traveler information system emerges within navigation systems, dynamic network routing becomes a crucial transportation problem. Using advanced traveler information technologies requires relaxing an important constraint in real problems. The approach advocated here allows for the incorporation of the most general travel time functions. It implies that the non-overtaking assumption, the so-called first-in-first-out (FIFO) property, can be relaxed. The non-overtaking assumption states that if A leaves node i of a link $i-j$ before B, B cannot arrive at node j before A. Although the FIFO property is useful for systems such as rail networks, it is not very applicable for road networks where overtaking occurs frequently. A non-FIFO assumption cannot be solved polynomially and makes the problem at least NP-hard.

A good review of the continuous-time dynamic shortest path problems can be found in [14]. However, its drawbacks are the FIFO property assumption and the linear approximation of link travel times, which to some extent make the problem less practical in real road traffic networks. In the discrete-time dynamic networks, studies such as [12] can be found with the FIFO assumption. Also, a fair body of research such as [15, 16] can be found on speedup time-dependent shortest path problems requiring the FIFO property. The continuous-time dynamic shortest path typically is employed in the network flow problems and has received a considerable attention in the literature, e.g., [21]. The problem formed here does not fall into any of mentioned categories.

3 Adaptive routing modeling

A road network is defined by a graph where links represent road segments and nodes correspond to intersections. 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. In this way, the time dependency of travel times is considered

in the problem. $\mu_{jk}(t)$ is the mean travel time (or 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 mean link travel times. Note that in the routing decision process, the origin node changes adaptively. In other words, the current node j is the origin node in dynamic optimization when determining the shortest path between j and the destination node.

The followings demonstrate an adaptive routing approach through a hypothetical example. It is assumed that drivers use the most updated information at each node to make a decision and chose the next link. Thus, at each node, the shortest path is formed using the last updated information. Fig. 1 shows a simple network that has 6 nodes and 8 links. Nodes 0 and 5 indicate the origin and destination nodes, respectively. Also, since it is assumed that there is no rest time at nodes, the arriving and departing times are the same. There are four paths for this network: path $1 = 0-1-3-5$, path $2 = 0-1-5$, path $3 = 0-2-4-5$, and path $4 = 0-2-5$.

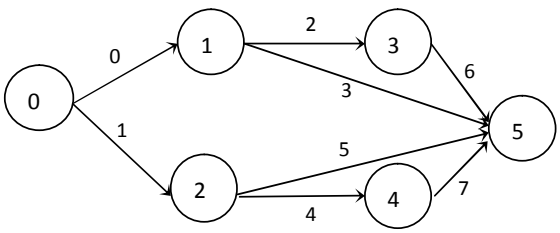

Fig. 1: A hypothetical network with 6 nodes and 8 links

The simulation period is set from 7 a.m. to 10 a.m. and time resolution is continuous all through the simulation. Suppose that the mean travel times are constant and equal to 12, 11, 15, 30, 20, 32, 14, and 13 min for nodes zero to 5, respectively. These link travel times are used to estimate the total travel times. In simulation, an arbitrary term of $0.03z$ where z is the standard normal distribution is added to the mean functions representing sample variations. This term shows the variant among travel time samples collected by sensors; however, there is no need of it in practice. It indicates that even if the mean travel times are constant, the mean of samples collected periodically is different. The mean travel time is assumed to be available at any time. Note that the mean travel time is not probabilistic. It is a time-dependent function representing the expected travel time. For instance, $\mu_{01}(10:30 \text{ a.m.}) = 12$ indicates that the expected travel time from node 0 to node 1 at 10:30 a.m. is 12 min. It is assumed that an intelligent system provides the expected travel times for any given link at any time.

Fig. 2 compares the total travel times from node 0 to node 5 for all four paths along with the adaptive routing approach. The total travel times are reported for drivers who depart node 0 between 7 a.m. and 9 a.m. The horizontal axis represents departure time from node 0 after 7 a.m. in minute scale. The vertical axis shows the total travel time to reach node 5.

Since the mean travel times are constant, the total travel times in Fig. 2 are expectedly the same regardless of departure times. This case can also be considered as an a priori problem. As a result, the adaptive optimal route must be identical to the optimal route obtained from executing the shortest path in node zero. Here, path $0-1-3-5$ has the least travel time with 41 minutes among the four paths. As it is shown, the adaptive method performs well in finding the shortest path in static conditions.

In order to illustrate the importance of the adaptive approach and evaluate its performance in a dynamic situation, the following occurrence on link 6 is assumed. Fig. 3 shows that an incident, maybe an accident at 8:20 a.m., happens. As a result, the mean travel time suddenly increases and then follows $\mu_6(t) = -3.7037 \times 10^{-4}t^2 + 0.0667t + 14$. In simulation, the recursive function of $T_5 = T_3 + \mu_6(T_5)$ is used to precisely estimate the travel time on link 6 where T_i is arrival/departing time at node i. Recall no rest time is assumed at nodes. The recursive function can be iterated until convergence with desired accuracy happens. Three iterations, which bring enough accuracy for this study, are executed. It should also be noticed that in a real situation there

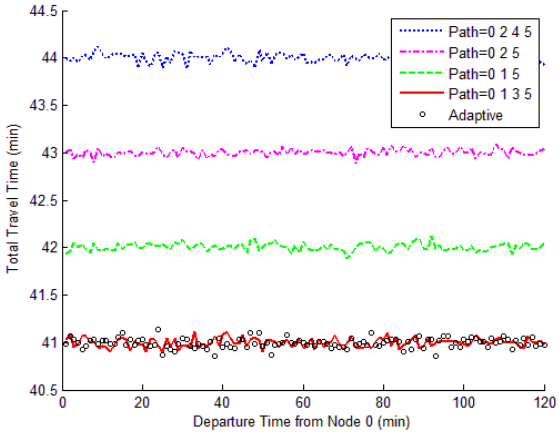

Fig. 2: Total travel times in static traffic condition

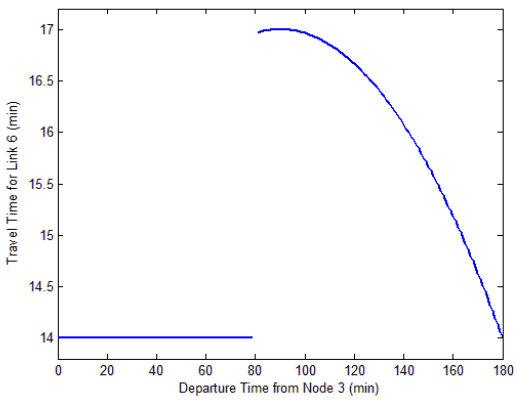

Fig. 3: The mean travel time with congestion on link 6

is no need of using the recursive function or assuming cost functions since the real values are provided by advanced traveler information systems.

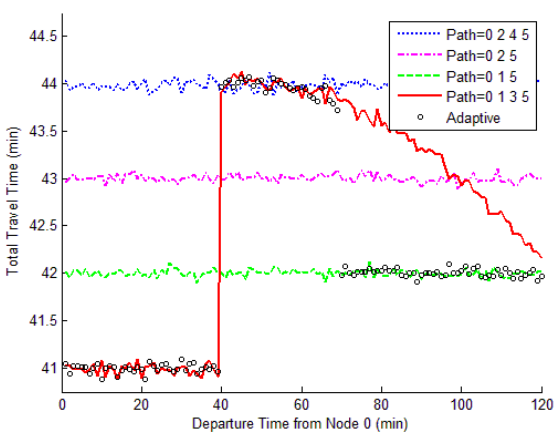

Fig. 4: Total travel times including the congestion of link 6

Fig. 4 compares the total travel times considering the congestion. Due to the incident on link 6, path $0-1-3-5$ and path $0-1-5$ are shortest paths with 41 and 42 minutes for drivers departing before and after 7:40, respectively. The Fig. 4 shows that the adaptive method performs quite well and finds the shortest

path before and after the occurrence. However, drivers departing node 0 from 7:40 to 8:10 experience longer travel time by choosing sub optimal path $0-1-3-5$. It is caused due to the fact that the travel time in path $1-3-5$ is shorter than that in path $1-5$ when drivers arrive at node 1. As a result, they select link 2 and later experience the congestion occurred on link 6. This is the limitation of all methods with no prediction ability. For instance, if the anticipatory information of mean travel time on link 6 were available, drivers could be informed and avoided the congestion. In other words, the shortest path is based on currently-observed travel times, and does not account for 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; more details can be found in [13].

4 Speedup algorithm

The following recursive equation is formulated to solve the dynamic problem for a given node j.

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

where $g(k)$ is the estimated 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. Note that 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.

A straightforward approach is to simply update all links by arriving at each node and subsequently apply a single-pair algorithm to find the shortest path from the current node to the destination. This approach requires updating all links and optimizing the whole network from scratch. Instead, herein the following approach proposed by [5] is utilized.

Step 1. A straightforward approach is to simply update all links by arriving at each node and subsequently apply a single-pair algorithm to find the shortest path from the current node to the destination. This approach requires updating all links and optimizing the whole network from scratch. Instead, herein the following approach proposed by [5] is utilized.

Step 2. Delete (j) deletes nodes whose labels are smaller than the current node j.

Step 3. 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.

Step 4. Path (j) finds the shortest path from the current node j to the destination.

Step 5. Traverse to the next node and go to Step 2.

In Step 4, Path (j), the decremented network is optimized to find the shortest path. Any classical or speedup algorithm can be utilized in this step. Optimization algorithms applied in Path (j) directly affect the speed of the decremental approach. [5] evaluated Dijkstra's algorithm. The algorithm decrements the network size at each node prior to optimizing a single-pair short path problem. 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 optimal routes. In their approach, Dijkstra's algorithm is applied to optimize the problem. In this article, A^* algorithm is utilized in the decremental approach to enhance the optimization speed.

The idea of a goal-directed or an A^* search[20] is to push the search towards the destination. Dijkstra's algorithm is a special case of A^* when reduced costs are considered. A^* removes nodes based on a defined priority. The priority key for removing a given node v is made up of two parts: the length of the tentative shortest path from the origin to v (as in Dijkstra's algorithm), and an underestimation of the distance to reach the destination from v. 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 A^* to speed up or improve shortest path problems; see [18, 24]. Here, a variant of A^* algorithm is utilized for speedup purposes and compared to Dijkstra's algorithm. However, before applying A^* algorithm, its potential function needs to be delineated. The function that gives priority to nodes and estimates the distance between a node and the destination is called the potential function.

A statistical technique is adopted here to define the potential function using reconstructed networks. The technique converts the original network whose links are time (cost) to a new network whose links are distance. Subsequently, the potential function is defined based on the new network. Multidimensional scaling is a statistical technique that represents measurements of dissimilarity (or similarity) of objects as distances between points probably in a low dimensional space[10]. Multidimensional scaling has its origins in psychometric and obtained popularity in science and engineering. In transportation, there are several usages of multidimensional scaling technique specially concentrating on road network visualizations, e.g., [22]. In order to find the potential function, the weighted multidimensional scaling technique is applied to reconstruct a network whose link lengths are times (costs). In other words, the potential function is the Euclidian distance between a node and the destination in the reconstructed network; more discussion can be found in [6]. [19] suggested the following general least-squares loss function for the weighted multidimensional scaling technique.

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

This loss function sums the differences of squared dissimilarities δ_{ij}^2, and squared distances d_{ij}^2 of pair ij (link $i-j$). $f(z)$ 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. $\mathbf{X}$ represents a new configuration for the node coordinates in the reconstructed network. Here, data are in a two dimensional format, and dissimilarities are the link travel times. The weighted metric multidimensional scaling technique is applied to create a new network whose link lengths are equal to the corresponding times (costs) in the original network. The weight can include missing values (undefined δ_{ij}). $w_{ij} = 1$ indicates that for pair ij a dissimilarity is observed, whereas $w_{ij} = 0$ is used for pairs where a dissimilarity is missing. In other words, missing data are excluded from the optimization criterion. In our problem, if there is no link between node i and j, it is considered missing, $w_{ij} = 0$.

The *mdscale* command in Matlab (statistics toolbox) is utilized to solve the weighted multidimensional scaling problem. It treats NaNs as missing values, and ignores them in the minimization of the loss function. The two-dimensional link cost matrix is assumed to be a dissimilarity matrix input, and the metric scaling criterion metricsstress is set for the SStress loss function, which is normalized with the sum of the 4^{th} powers of the dissimilarities. The initial locations are considered to be the starting point in the optimization. The outputs are the Cartesian coordinates of the nodes and the optimal value of the SStress loss function. Note that in the reconstructed networks, the node coordinates are different from the original ones; however, link lengths still represent the desired costs.

5 Experimental simulation

This section evaluates the performance of the decremental approach using the A^* algorithm in dynamic networks. It is assumed that traffic information is fully available up to the current time. The most updated information is utilized in adaptive routing. Thus, at each node, a single-pair shortest path problem is executed using the last updated information. It is assumed that link cost functions are time-dependent without a priori knowledge. All analyses and simulations are conducted using MATLAB software with a Dual-Core 3.0 GHz CPU. In simulation, networks are randomly generated in a two-dimensional Cartesian system. Random points representing nodes are generated in a predefined area of $1,000 \times 1,000$ in the first quadrant. 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.

At each run, two approaches are considered before applying A^* or Dijkstra's algorithm. The first approach includes all of the network nodes with the original network size. The second one, hereafter called Decremental, utilizes the 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 is similar to the Dijkstra approach except that it implements the decremental approach to reduce the network size before applying Dijkstra's algorithm. In A^* approach, the weighted multidimensional scaling technique is used to define the potential function for the whole network and prioritize the nodes. The Decremental-A^* approach reduces the network size at each iteration by applying the decremental approach, and subsequently utilizes A^*.

Different networks with 500, 1,000, 1,500, 2,000, 2,500, and 3,000 nodes are selected. In order to increase the reliability of results, each set of simulations is executed 11 times. For instance, Tab. 1 presents 11 runs of random networks with size of $n = 3,000$ for the Decremental-A^* approach. It illustrates sample runs sorted based on the CPU times. The table displays number of links in network (m), number of traversed links on optimal routes (i.e., required iterations to arrive to the destination), the CPU times in second scale to find the optimal routes, and the optimal values of the SStress loss function. 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 when applying the shortest path algorithms considering available inputs.

Table 1: The Decremental-A^* approach for 11 random networks with $n = 3,000$

Run	m	Traversed Links	CPU Time (Sec.)	SStress
6	8,975	50	1.13	2.87×10^{-15}
2	8,979	53	1.82	2.72×10^{-15}
4	8,974	50	1.91	3.38×10^{-15}
11	8,974	55	1.98	2.97×10^{-15}
1	8,973	58	2.03	3.26×10^{-15}
9	**8,980**	**52**	**2.15**	$\mathbf{2.10 \times 10^{-15}}$
3	8,979	56	2.16	1.15×10^{-15}
10	8,977	56	2.29	2.35×10^{-15}
8	8,969	56	2.34	2.52×10^{-15}
7	8,976	65	2.5	4.33×10^{-15}
5	8,977	60	3.13	1.06×10^{-15}

For further analyses, runs are chosen based on the median of the CPU times from 11 random samples. For instance, in Tab. 1, 2.15 is the median of the CPU times; accordingly, the 9$^{\text{th}}$ sample is selected to represent the results for the Decremental-A^* approach for the network with size of $3,000$. This network has $8,980$ links and the optimal route is obtained by traversing 52 links from the origin to the destination node. The SStress column indicates the loss function is properly minimized in the weighted multidimensional scaling technique. In other words, it confirms that the new networks are accurately reconstructed in all runs.

Fig. 5 displays 24 median sample runs (four approaches and six networks). It is assumed that the median sample runs can characterize the corresponding networks. Fig. 5 indicates the decremental approach reduces the CPU times significantly and A^* is superior to Dijkstra's algorithm. However, estimating the potential function, i.e., reconstructing the network, may be extensive either due to pre-processing time or memory consumption depending upon the link travel times and the topology of the network. Fig. 5 shows that the performance of Decremental-A^*, A^*, Decremental-Dijkstra, and Dijkstra are always in this order, $DA^* > A^* > DD > D$.

Tab. 2 compares CPU time reductions in detail. For instance, for the network with 3,000 nodes A^* and Decremental-A^* require 4.07 and 2.15 seconds, respectively. Thus, compared to A^*, DA^* has $((4.07 - 2.15)/4.07)\,100\% = 47\%$ improvement in CPU time shown in the second column. Overall, the results indicate homogeneous reductions for different networks. The last row shows the average reductions for all six comparisons. The columns $DA^* > A^*$ and $DD > D$ indicate that the decremental approach saves approximately 40% to 45% of CPU time when compared to their non-decremental counterparts. Also, columns $DA^* > DD$ and $A^* > D$ signify about 70% CPU time reduction by using A^* in place of the Dijkstra algorithm. Moreover, A^* is superior to the Decremental-Dijkstra approach, see the $A^* > DD$ column. As it

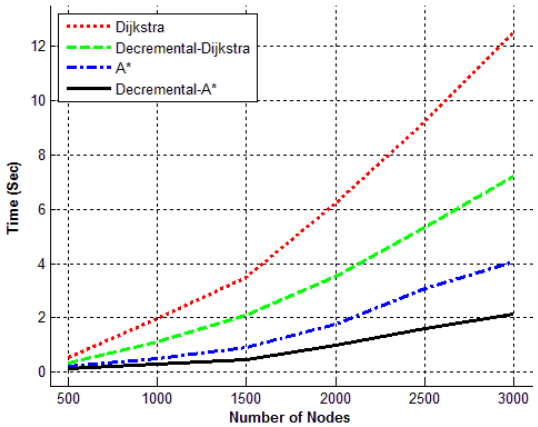

Fig. 5: Medians of CPU times in four approaches

is specified in $DA^* > D$, the Decremental-A^* approach clearly outperforms the Dijkstra's algorithm. As a whole, among all approaches, the Decremental-A^* approach has the minimum CPU times.

Table 2: Rounded CPU time reductions in percentage

N	$DA^* > A^*$	$DA^* > DD$	$DA^* > D$	$A^* > DD$	$A^* > D$	$DD > D$
500	39	66	78	43	64	37
1,000	44	75	86	55	74	43
1,500	48	78	87	57	74	40
2,000	43	71	84	50	71	43
2,500	47	70	82	42	67	42
3,000	47	70	83	44	67	42
Average	**45**	**72**	**83**	**49**	**70**	**41**

6 Conclusion

This article considers real-time travel times provided by advanced traveler information technologies in a shortest path problem context. This advancement challenges the problem with more practical assumptions in road traffic networks such as relaxing the FIFO property. Specifically, the continuous-time dynamic shortest path problem without a priori knowledge of link travel times needs to be tackled. In this effort, the adaptive routing approach is applied, which can be emerged within vehicle navigation systems and provide drivers with turn-by-turn directions. The existing adaptive approach uses Dijkstra's algorithm to find the shortest path. A^* algorithm is applied to enhance the optimization speed. Moreover, the weighted multidimensional scaling technique is employed to convert link cost functions to distances, which outlines the potential function in A^* algorithm. The impact of A^* algorithm is evaluated and compared with Dijkstra's algorithm in different time-dependent networks. The simulation results indicate that using A^* algorithm decreases the CPU times significantly (about 70%). Among the four evaluated approaches, the Decremental-A^* approach is eventually advocated.

As a direction for future research, the performance of stochastic-based search methods can be investigated in comparison with A^* algorithm. More specifically, the simulated annealing algorithm or some of its extensions such as annealing-based particle swarm algorithm are appropriate, see [26, 27]. The simulated annealing-based methods are capable of escaping local optima with a high probability. Furthermore, another future research is to employ multi-objective methods to find Pareto optimal solutions when the accuracy and CPU time need to be trade-off. These methods can be found in [1–3, 7, 25]. It is also important to pay attention to the assumptions when model problems; [4, 8, 28] are instances emphasizing this issue.

References

[1] M. Ardakani, R. Noorossana. A new optimization criterion for robust parameter designłthe case of target is best. *The International Journal of Advanced Manufacturing Technology*, 2008, **38**(9-10): 851–859.

[2] M. Ardakani, R. Noorossana, et al. Robust parameter design using the weighted metric methodłthe case of 'the smaller the better'. *International Journal of Applied Mathematics and Computer Science*, 2009, **19**(1): 59–68.

[3] M. K. Ardakani. The impacts of errors in factor levels on robust parameter design optimization. *Quality and Reliability Engineering International*, 2015.

[4] M. K. Ardakani, D. Das, et al. Estimation in second-order models with errors in the factor levels. *Communications in Statistics-Theory and Methods*, 2011, **40**(9): 1573–1590.

[5] M. K. Ardakani, L. Sun. Decremental algorithm for adaptive routing incorporating traveler information. *Computers & Operations Research*, 2012, **39**(12): 3012–3020.

[6] M. K. Ardakani, M. Tavana. A decremental approach with the A^* algorithm for speeding-up the optimization process in dynamic shortest path problems. *Measurement*, 2015, **60**: 299–307.

[7] M. K. Ardakani, S. S. Wulff. An overview of optimization formulations for multiresponse surface problems. *Quality and Reliability Engineering International*, 2013, **29**(1): 3–16.

[8] M. K. Ardakani, S. S. Wulff. An extended problem to bertrand's paradox. *Discussiones Mathematicae: Probability & Statistics*, 2014, **34**.

[9] H. Bast, D. Delling, et al. Route planning in transportation networks. *Tech. Rep.*, Microsoft, 2015.

[10] I. Borg, P. J. Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

[11] I. Chabini, B. Dean. Shortest path problems in discrete-time dynamic networks: complexity, algorithms and implementations. *Tech. Rep.*, Massachusetts Institute of Technology, MA, 1999.

[12] I. Chabini, S. Lan. Adaptations of the A^* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks. *Intelligent Transportation Systems, IEEE Transactions on*, 2002, **3**(1): 60–74.

[13] Y.-C. Chiu, J. Bottom, et al. A primer for dynamic traffic assignment. *Transportation Research Board*, 2010, 2–3.

[14] B. C. Dean. *Continuous-time dynamic shortest path algorithms*. Ph.D. Thesis, Massachusetts Institute of Technology, 1999.

[15] D. Delling. *Engineering and augmenting route planning algorithms*. Ph.D. Thesis, Karlsruhe Institute of Technology, 2009.

[16] D. Delling. Time-dependent sharc-routing. *Algorithmica*, 2011, **60**(1): 60–94.

[17] S. Gao, I. Chabini. Optimal routing policy problems in stochastic time-dependent networks. *Transportation Research Part B: Methodological*, 2006, **40**(2): 93–122.

[18] A. V. Goldberg, H. Kaplan, R. F. Werneck. Reach for A^*: Shortest path algorithms with preprocessing. *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, 2009, **74**: 93–139.

[19] P. J. Groenen, J. de Leeuw, R. Mathar. Least squares multidimensional scaling with transformed distances. **in:** *From data to knowledge*, Springer, 1996, 177–185.

[20] P. E. Hart, N. J. Nilsson, B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 1968, **4**(2): 100–107.

[21] S. M. Hashemi, E. Nasrabadi. On solving continuous-time dynamic network flows. *Journal of Global Optimization*, 2012, **53**(3): 497–524.

[22] C. Kaiser, F. Walsh, et al. User-centric time-distance representation of road networks. **in:** *Geographic information science*, Springer, 2010, 85–99.

[23] S. Lim. *Traffic prediction and navigation using historical and current information*. Ph.D. Thesis, Massachusetts Institute of Technology, 2008.

[24] G. Nannicini, D. Delling, et al. Bidirectional A^* search for time-dependent fast paths. **in:** *Experimental Algorithms*, Springer, 2008, 334–346.

[25] R. Noorossana, M. K. Ardakani. A weighted metric method to optimize multi-response robust problems. *Journal of Industrial Engineering International*, 2009, **5**: 10–19.

[26] N. Safaei, M. Saidi-Mehrabad, M. Jabal-Ameli. A hybrid simulated annealing for solving an extended model of dynamic cellular manufacturing system. *European Journal of Operational Research*, 2008, **185**(2): 563–592.

[27] N. Safaei, R. Tavakkoli-Moghaddam, C. Kiassat. Annealing-based particle swarm optimization to solve the redundant reliability problem with multiple component choices. *Applied Soft Computing*, 2012, **12**(11): 3462–3471.

[28] V. Sarhangian, A. Vaghefi, et al. Optimizing inspection strategies for multi-stage manufacturing processes using simulation optimization. **in:** *Proceedings of the 40th Conference on Winter Simulation*, Winter Simulation Conference, 2008, 1974–1980.

[29] D. Schrank, B. Eisele, T. Lomax. The 2011 urban mobility report. *Texas A&M Transportation Institute. The Texas A&M University System*, 2012.