Near-Future Traffic Evaluation based Navigation for Automated Driving Vehicles Considering Traffic Uncertainties

Kuen-Wey Lin[1], Masanori Hashimoto[2], Yih-Lang Li[1]
[1] Department of Computer Science, National Chiao Tung University, Hsinchu 300, Taiwan
[2]Department of Information Systems Engineering, Osaka University, Osaka 565-0871, Japan
[1]E-mail: kuenweylin.cs99g@nctu.edu.tw

Abstract

Because it is difficult to find empty space in a developed city to accommodate more transportation infrastructures, the development of an effective navigation system is a low cost option for mitigating traffic jam. Regarding a future world where automated driving technologies have become mature and most vehicles follow the pre-scheduled route suggested by a navigation system, it is likely to predict the traffic jam accurately if the navigation system can know the pre-scheduled route of each vehicle. Recently, a navigation algorithm is presented for automated driving vehicles with the assumption that all the navigating query requests are processed by a single system. However, the aforementioned algorithm does not consider any kind of uncertainty originating from accidents and destination change. To get close to the real world, we propose a navigation algorithm with near-future evaluation capability that also allows some kinds of uncertainties. We compare our algorithm with a dynamic-update based conventional navigation algorithm without near-future evaluation capability. We download some metropolitan maps from OpenStreetMap and utilize the data of traffic flow from official statistics to randomly generate many sets queries. Experimental results show that the total cruising time is improved for each case.

Keywords

Advanced Driver Assistance Systems, Self-Driving Vehicles, Planning and Decision

1. Introduction

The coordination and management of vehicles in a big city with a complex network of roads are critical to reduce traffic jam, which also reduces driving time, pollution, and noise. As compared to the high cost of building more transportation infrastructure in a developed city, the development of an effective navigation system is a low cost option for mitigating traffic jam. Nowadays, an on-board navigation device and a navigation service provider, such as Google Maps [1], can identify the shortest or fastest route based on the current or history traffic information for a vehicle. Each driver does not know the traffic situation that he will face along the shortest or fastest route and a traffic jam may still happen because each vehicle does not know the routes of the other vehicles. To plan routes for vehicles, previous works have focused on the traffic assignment problem or the multi-agent routing problem. For example, with a behavioral approach, a dynamic traffic assignment model can be used to represent the interaction between travel choices and traffic flows [2]. Multi-agent based vehicle routing systems focus on how these agents perform communication between them to improve decision making [3]. With route information sharing, [4] calculates the probability of traffic congestion of a road to facilitate re-routing for a driver. A traffic prediction system was demonstrated in [5] for a vehicle. For a set of tours with multiple destinations, [6] presented an algorithm to schedule the order of visiting. [7] proposed an adaptive in-vehicle routing system with real-time traffic information to get the least cost path. Regarding a future world where automated driving technologies have become mature and most drivers are willing to transfer the driving control to an automatic driving system that always follows the pre-scheduled route suggested by a navigation system in normal situations, it is likely to predict the traffic jam accurately if the navigation system can know the pre-scheduled route of each vehicle. Recently, a navigation algorithm based on near-future evaluation is proposed to effectively reduce traffic jam for hundreds of thousands of automated driving vehicles [8]. However, the aforementioned algorithm assumes that all the automated driving vehicles follow their planned routes, and it does not consider any kind of uncertainty, for example, traffic accidents or the change of route due to the change of destination or a temporary stop.

In this paper, we propose a navigation algorithm to guide vehicles based on the assumption that all the navigating query request are processed by a single system. To get close to the real world, some kinds of uncertainties are allowed, such as abrupt traffic jams caused by vehicle accident and prediction failure caused by changing the destination or a temporary stop. First, for a metropolitan map, we randomly generate traffic accidents on some roads. The frequency of traffic accidents is derived from official statistics [9][10][11]. We compare our navigation algorithm with a dynamic-update based conventional navigation algorithm without near-future evaluation capability. Secondly, as a person starts his journey by automatic driving, he may change the destination or temporarily park for something before reaching the destination and find a route by himself. For these situations, the navigating system has to update the future traffic flow of the roads on the pre-scheduled route. We download the maps of Kyoto, Osaka, and Taipei cities from OpenStreetMap [12] and utilize the data of traffic flow from official statistics to randomly generate many sets queries with different quantity (a query is issued by a vehicle). Experimental results demonstrate that, compared with dynamically updated navigating system, the total cruising time is improved for each case.

This paper is organized as follows: the next section presents the problem formulation. In Section 3 gives ours algorithm with the consideration of some kinds of uncertainties. Simulation and experimental results are shown in Section 4. Finally, we conclude this paper in Section 5.

2. Problem Formulation

First, we describe the graph to model traffic condition. A set of inputs for navigating queries and traffic uncertainties are defined. Then, we describe the output for routes and the objective of the problem. Finally, the assumptions are provided.

2.1 Models and Objective

A weighted and directed graph $G = (V, E)$ consists of a set of vertices $V = \{v_1, v_2, \ldots, v_{|V|}\}$ and a set of edges $E = \{e_1, e_2, \ldots, e_{|E|}\}$. The number of vertices is $|V|$, and the number of edges is $|E|$. A vertex represents a junction, and an edge represents a lane. A weight is defined for each edge e_i: $w_i \in \{x \mid x \in \mathbf{Q}, x > 0\}$ and calculated based on Greenshield's V-K relationship [13] as follows:

$$w\left(e, d_q^e\right) = \frac{l_e}{s_e \times \left(1 - \frac{(n_e + 1) \times (l_v + g)}{l_e}\right)} \times k, \qquad (1)$$

where d_q^e is the number of vehicles on edge e when vehicle q entering e; s_e is the speed limit of e; n_e is the number of vehicles on e; l_v is the length of a vehicle; l_e is the length of an edge; g is the gap between two vehicles; K is a user-defined constant. In this paper, s_e, l_v, and g are given in advance. K is set to 1. We utilize equation (1) to calculate the cruising time of vehicle q to pass through edge e with traffic condition d_q^e.

A set of navigating queries $U = \{u_1, u_2, \ldots, u_{|U|}\}$ is requested by a vehicle. The number of queries is $|U|$. A query $u_i \in U$ consists of three attributes: $V_s^{u_i}$ is the location where u_i is requested; $V_d^{u_i}$ is the destination ($V_s^{u_i} \in V$, $V_d^{u_i} \in V$); $T_s^{u_i}$ is the time when u_i is requested. A vehicle can request many queries, but it has at most one query at the same time. Our model can allow $V_s^{u_i}$ and $V_d^{u_i}$ at any location on a map by storing $V_s^{u_i}$ and $V_d^{u_i}$ in a vertex or an edge.

To take into account traffic accidents, we newly annotate traffic accidents to edges. Each edge e has a set of traffic accidents $A^e = \{a^e_1, a^e_2, \ldots, a^e_{|Ae|}\}$ where a^e_i is a traffic accident occurred at edge e. The number of traffic accidents on edge e is $|A^e|$. Traffic accident a^e_i is a time period $[t_s^{a_i^e}, t_f^{a_i^e}]$ where $t_s^{a_i^e}$ is the starting time of a^e_i and $t_f^{a_i^e}$ is the finishing time of a^e_i. An edge can have many traffic accidents, but it has at most one traffic accident at the same time. We use $A = \{a^1, a^2, \ldots, a^{|E|}\}$ to refer to the union of all A^e. A set of pre-planned schedules $R = \{r_1, r_2, \ldots, r_{|R|}\}$ is used to demonstrate the functionality of a navigation algorithm considering manually driven vehicles. Pre-planned Schedule r_i consists of two attributes: $T_s^{r_i}$ is the starting time to traverse the next edge; E^{r_i} is an ordered list of edges which is a subset of E, and it is denoted by $(e_1^{r_i}, e_2^{r_i}, \ldots, e_{|r_i|}^{r_i})$ where $e_j^{r_i}$ is an edge traversed by the vehicle associated with r_i. The number of edges in E^{r_i} is $|r_i|$. Note that, one element of R is associated with one vehicle, and that vehicle is a manually driven vehicle.

The output of the problem is a set of schedules $S = \{s_1, s_2, \ldots, s_{|U|}\}$ where s_i denotes a schedule of query $u_i \in U$. Schedule s_i is an ordered list of edges $\{e_1^{u_i}, e_2^{u_i}, \ldots, e_{k_{s_i}}^{u_i}\} \subset E$. Elements in s_i are traversed in order by the vehicle requesting query u_i. $e_1^{u_i}$ is the edge connecting $V_s^{u_i}$ (the location where u_i is requested); $e_{k_{s_i}}^{u_i}$ is the edge connecting $V_d^{u_i}$ (the destination of u_i); k_{s_i} is the number of edges of s_i.

The objective of the problem is to reduce the total cruising time of all schedules. We define $ct(s_i)$ as the cruising time for schedule s_i and it is calculated by $\sum_{j=1}^{k_{s_i}} ct\left(e_j^{u_i}\right)$ where $ct\left(e_j^{u_i}\right)$ represents the cruising time to pass through edge e_j of query u_i and is calculated by equation (1). Thus, the total cruising time of all schedules is $\sum_{i=1}^{|U|} ct(s_i)$.

2.2 Assumptions

When a traffic accident occurs at an edge, we set the weight of the edge as a very big constant before the finishing time of the traffic accident; that is, the cruising time to pass through that edge will be very long. The proportion of pre-planned schedules R to navigating queries U is small because we consider a future world where automated driving technologies have become mature and most vehicles follow the instructions a navigation system.

3. Proposed Algorithm

We introduce the idea of near-future traffic evaluation with uncertainties at first. Based on the idea, we propose our algorithm. Then, an illustrative example is used to show the effectiveness of the algorithm.

To evaluate near-future traffic condition with the consideration of traffic uncertainties, we maintain a timetable b_e for an edge $e \in E$. Two types of bar lines are utilized in a timetable. The first one represents a time period when a vehicle enters edge e and when that vehicle exits e. The second one refers to the starting time and the finishing time of a traffic accident. According to the overlapping of these bar lines, we can interpret the traffic condition of the corresponding edge at any time point. The insertion and removal of bar lines are performed by the proposed algorithm; however, the proposed algorithm cannot modify the bar lines of vehicles not following the instruction of the proposed algorithm. Moreover, the bar line of a traffic accident will not be included in a timetable until that traffic accident occurs because we cannot observe a traffic accident in advance.

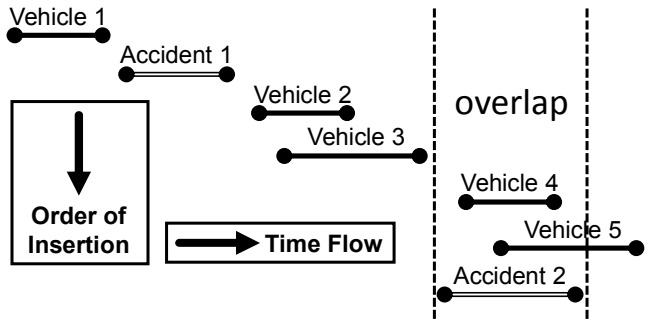


Fig. 1: A timetable for an edge. Each bar line is for a vehicle or a traffic accident. The proposed algorithm utilizes a timetable to evaluate the near-future traffic condition for an edge.

As shown in Fig. 1, there are five bar lines of vehicles and two bar lines of traffic accidents when the second bar line of traffic accident is just inserted into the timetable. The length of a bar line increases when the number of vehicles increases. The first bar line of traffic accident does not overlap any bar line of vehicle because the proposed algorithm has removed the overlapping to reduce cruising time. The bar lines of vehicle 4 and vehicle 5 overlap with the one of traffic accident 2 because we do not observe any traffic accident when we instruct that two vehicles. As soon as we observe traffic accident 2, we will arrange new paths for two vehicles whose cruising time periods overlap with that of traffic accident 2 to reduce cruising time.

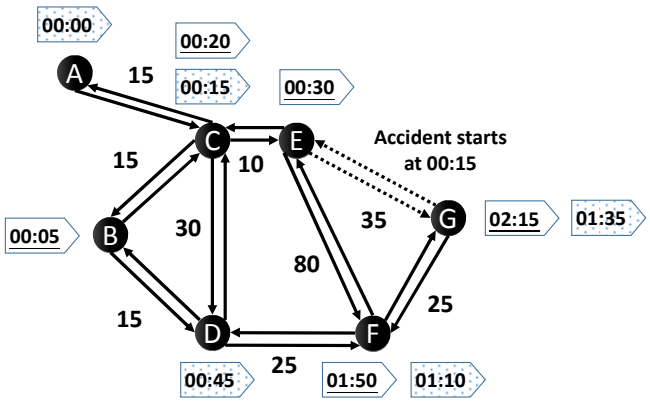
Now we describe the proposed algorithm. We utilize a queue Y_U to maintain the set U of navigating queries, a queue Y_A to maintain the set A of traffic accidents, and a queue Y_R to maintain the set R of pre-planned schedules. These queues are sorted according to $T_s^{u_i}$, $t_s^{a_i^e}$, and $T_s^{r_i}$ respectively. Note that, the system time is determined with $T_s^{u_i}$ of the earliest element of Y_U. That is, we can observe all events that occur before $T_s^{u_i}$, but we cannot observe any events that occur after $T_s^{u_i}$. At first, the earliest element (a query u_i with earliest $T_s^{u_i}$) of Y_U is extracted. A path with minimum cruising time is found on $G = (V, E)$ according to Equation 1, and the timetables on the newly found path are updated; then, $V_s^{u_i}$ is set to the next junction along the newly found path, and $T_s^{u_i}$ is updated accordingly. Next, we will check Y_A to see whether a traffic accident occurs or not. If a traffic accident occurs, we update the affected timetables and re-identify new paths for the affected vehicles. Finally, we check Y_R to see whether the starting time to traverse the next edge is earlier than the current system time. If $T_s^{r_i}$ of a pre-planned schedule r_i is earlier than the current system time, we will update its $T_s^{r_i}$ and the affected timetables. Note that, the proposed algorithm cannot know the future path of a pre-planned schedule in advance since the pre-planned schedule is determined by the driver of each manually driven vehicle; that is, it can only know where the vehicle of a pre-planned schedule is running.

Algorithm: Near-future traffic evaluation based navigation considering traffic uncertainties
Input: The map $G=(V, E)$, the weight function $w(e, d_q^e)$, a set U of queries, a set A of traffic accidents, and a set R of pre-planned schedules
Output: A set S of schedules
Begin
1. Initialize queue $Y_U/Y_A/Y_R$ with $U/A/R$;
2. **while** $Y_U \neq \varnothing$
3. Extract a query u_i with minimum $T_s^{u_i}$ from Y;
4. **for** each traffic accident a^e_j
5. **if** $t_s^{a_j^e} < T_s^{u_i}$
6. Extract a^e_j from Y_A;
7. Update affected timetables;
8. Re-find paths for affected vehicles;
9. **for** each pre-planned schedule r_k
10. **if** $T_s^{r_k} < T_s^{u_i}$
11. Update $T_s^{r_k}$;
12. Update affected timetables;
13. Re-insert r_k to Y_R;
14. Initialize map G;
15. Use the enhanced Dijkstra's algorithm to find a path with minimum cruising time for u_i;
16. Assign u_i's $V_s^{u_i}$ to the junction which is next to the original source on the found path;
17. Update $T_s^{u_i}$ of u_i;
18. Update affected timetables;
19. Re-insert u_i to Y_U;
end

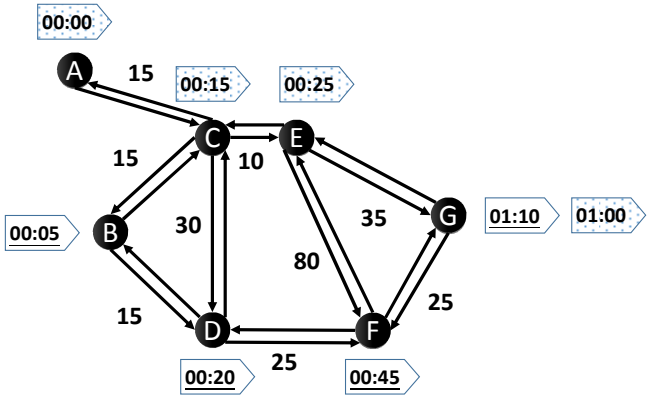
Fig. 2: The Proposed near-future traffic information evaluation Algorithm.

We show the pseudo code in Fig. 2. Note that, the weight of an edge e on G is the cruising time for u_i to pass through e. Line 3 determines the current system time. Lines 4-8 handle traffic accidents and pre-planned schedules are handled at lines 9-13. Lines 14-15 utilize the extended Dijkstra's algorithm [14] to find a path with minimum cruising time on G. Line 16 updates $V_s^{u_i}$ according to the newly found path. Lines 17-18 update $T_s^{u_i}$ and affected timetables. Finally, the updated u_i is re-inserted into Y_U if the vehicle of u_i has not arrived at its destination at Line 19.

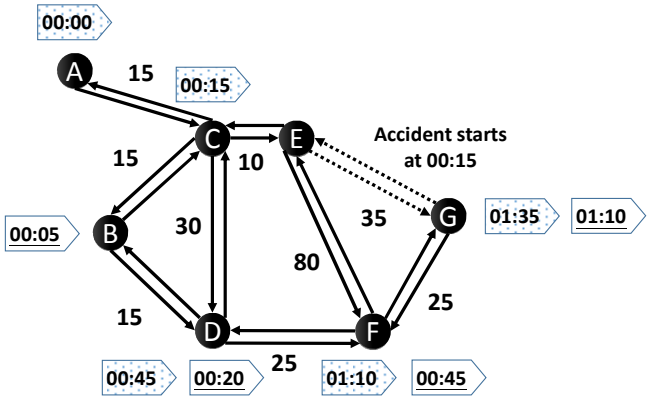
An example with seven junctions (vertices) and eighteen directed edges (roads) is shown in Fig. 3.The number next to an edge is the cruising time to pass through that edge if there is no traffic congestion. For simplicity, we assume that a traffic congestion occurs if the number of vehicles on an edge exceeds 10. In this case, there are eleven queries. Ten queries from A to G are named as Q_{AG1} to Q_{AG10}. One query from B to G is named as Q_{BG}. Q_{BG} starts at 00:05 (mm:ss) while the other queries start at 00:00. We assume that there is a traffic accident occurring at 00:15 between E and G.



(a)



(b)



(c)

Fig. 3: (a) The result of a dynamic-update based traditional navigating algorithm without near-future traffic evaluation. (b) When there is no traffic accident, the proposed algorithm will guide vehicles to detour the roads where traffic congestion are going to happen. (c) The final result of the proposed algorithm with a traffic uncertainties.

As shown in Fig. 3(a), we introduce a dynamic-update based traditional navigating algorithm without near-future traffic evaluation.

At first, the traditional algorithm suggests path *A-C-E-G* for Q_{AG1}~Q_{AG10} because no traffic accident occurs at that moment. The traditional algorithm suggests path *B-C-E-G* for Q_{BG} because it does not know there will be a traffic congestion on the edge between *C* and *E* at 00:20. When Q_{AG1}~Q_{AG10} arrive at *C* at 00:15, the traditional algorithm observes a traffic accident occurring on edge *EG* and suggests path *C-D-F-G* for Q_{AG1}~Q_{AG10}. When Q_{BG} arrives at *C*, the traditional algorithm suggests path *C-E-F-G* for Q_{BG} because a traffic congestion will occur as Q_{BG} enters edge *CD* (the number of vehicles exceeds 10). As a result, the cruising time is 95 seconds for Q_{AG1}~Q_{AG10} and 135 seconds for Q_{BG}.

We discuss the performance of the proposed algorithm using Fig. 3(b) and Fig. 3(c). At first, we do not know a traffic accident occurring at 00:15; thus the proposed algorithm suggests path *A-C-E-G* for Q_{AG1}~Q_{AG10} and path *B-D-F-G* for Q_{BG} to avoid any potential traffic congestion. When Q_{AG1}~Q_{AG10} arrive at *C*, as shown in Fig. 3(c), the proposed algorithm observes the traffic accident and suggests path *C-D-F-G* for Q_{AG1}~Q_{AG10} because Q_{AG1}~Q_{AG10} will not meet Q_{BG} on that path. Compared the result of the traditional algorithm (shown in Fig. 3(a)), the proposed algorithm (shown in Fig.3 (c)) can save 48% cruising time for Q_{BG}.

4. Experimental Results

We implemented the proposed algorithm using C/C++ language on a 3.0-GHz Intel machine with 48-GB memory under CentOS 5.11 operating system. The maps for Kyoto, Osaka, and Taipei metropolitan maps are downloaded from OpenStreetMap [12]. The data of these cities are shown in Table 1. We use the data of traffic flow from Department of Transportation, Taipei City Government [9] to generate a set of queries. Based on the trend of traffic flow during one day, we randomly generate some sets of queries with different quantity for comparison. We compared the proposed algorithm with a dynamic-update based traditional navigating algorithm without near-future traffic evaluation (DNXNFE). When arriving a junction, DNXNFE will re-find a path with minimum cruising time based on the traffic condition at that moment for a vehicle. Traffic accidents account for 0.00011% of the number of queries. The duration of each traffic accident is set to be thirty minutes. The number of vehicles changing their destinations, making a temporary stop, or discontinuing following the pre-planned routes accounts for 0.1% of the number of queries.

We list the number of queries, the total reduced cruising time (in hour), and the total reduced mileage (in kilometer) in Table 2. We can observe that the total reduced cruising time and total reduced mileage can be up to 134 thousand hours and 1.6 million kilometers, respectively, for the case with over 0.5 million vehicles (Taipei_523k). The objective of the proposed algorithm is to reduce cruising time; thus, in some cases with few queries such as Osaka_289k, the total mileages of all vehicles increase. However, for the cases with many queries, the saving of mileages is good. More figures are used to detail the performance of the proposed algorithm in the next paragraph. Note that, for all cases, the total cruising time is improved. We also list the improvement in cruising time and mileage in Table 2.

At first, we discuss case Osaka_289k, a case with nearly 0.3 million vehicles. As shown in Table 2, we can observe

that the improvement in mileage is negative. We utilize Fig. 4 to explain the phenomenon. On the whole, the numbers of vehicles with cruising time and mileage improvement are larger than those with deterioration in both. Therefore, even though few vehicles with nearly 5% deterioration in mileage, the average deterioration is only 0.02%. We observe the routes of the vehicles with long mileage in detail. We find that the proposed algorithm can guide some vehicles to avoid traffic congestion and reduce cruising time using the roads with high speed limit such as expressway. We utilize Fig. 5 to describe the difference in speed for case Osaka_289k. At each time period, we calculate the numbers of vehicles at that speed range for the proposed algorithm and DNXNFE. Then, we subtract the number of DNXNFE from the one of the proposed algorithm. The difference between the numbers of vehicles for each time period is shown in *z*-axis. The speed range is shown in *x*-axis and the data per hour is shown in *y*-axis. In other words, a peak at a speed range means that the proposed algorithm has more vehicles than that of DNXNFE at that speed range. We can see that there is a peak at [80-90+]. It implies that most of our queries are with the average speed between 80km/h and 90km/h. On the other hand, a hollow at a speed range means that DNXNFE has more vehicles than that of the proposed algorithm at that speed range. We can observe that there is a hollow at [30-40]; thus, we can conclude that many vehicles of DNXNFE are with the average speed between 30km/h and 40km/h. In brief, the proposed algorithm can provide good traffic flow. The abbreviations used in line charts or straight bar charts are shown in Table 3.

As for the case Taipei_523k, in Fig. 6, we can observe that the numbers of queries with cruising time and mileage improvement are also larger than those with deterioration in both. Few vehicles have deteriorated cruising time and mileage because some vehicles that lead to or pass by congested region are scheduled to detour congested region in advance. Especially, the improvement in cruising time can be up to 52% at a time period (7 p.m.) because the proposed algorithm can effectively consider some traffic uncertainties to avoid traffic congestion. The proposed algorithm can evaluate the near-future traffic condition and find some alternative routes to solve traffic congestion in advance. Fig. 7 shows that most of our vehicles are with the average speed between 40km/h and 50km/h; however, most of DNXNFE's vehicles are with the average between 30km/h and 40km/h. Furthermore, there is a hollow at (0-10), which means that a big traffic jam occurs by DNXNFE.

5. Conclusion

In this paper, we have proposed a near-future traffic evaluation based navigation algorithm considering traffic uncertainties. Compared with a traditional navigation algorithm without near-future traffic evaluation, even with some traffic uncertainties, the proposed algorithm can still guide a vehicle to detour the roads where traffic congestion are going to happen in the near future. The distribution of the cruising speeds of vehicles confirms that the proposed algorithm can provide smooth traffic flows.

6. References

[1] Google Maps [Online]. Available: https://maps.google.com/

[2] Y.-C. Chiu, J. Bottom, M. Mahut, A. Paz, R. Balakrishna, T. Waller, and J. Hicks, "Dynamic Traffic Assignment: A Primer," *Transportation Research Circular E-C153*, June 2011.

[3] R.R. Negenborn, B. De Schutter, and J. Hellendoorn, "Multi-Agent Model Predictive Control for Transportation Networks: Serial versus Parallel Schemes," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 3, pp. 353–366, Apr. 2008.

[4] T. Yamashita, K. Izumi, and K. Kurumatani, "Car Navigation with Route Information Sharing for Improvement of Traffic Efficiency," in *Proc. of the IEEE International Conference on Intelligent Transportation Systems*, 2004, pp. 465-470.

[5] B. S. Kerner, H. Rehborn, M. Aleksic, and A. Haug, "Traffic Prediction Systems in Vehicles," in *Proc. of the IEEE International Conference on Intelligent Transportation Systems*, 2005, pp. 251-256.

[6] H. Kuriyama,, Y. Murata,, N. Shibata, K. Yasumoto, and M. Ito, "Congestion Alleviation Scheduling Technique for Car Drivers Based on Prediction of Future Congestion on Roads and Spots," in *Proc. of the IEEE International Conference on Intelligent Transportation Systems*, 2007, pp.910-915.

[7] C. Li, S. G. Anavatti, and T. Ray, "Adaptive Route Guidance System with Real-Time Traffic Information," in *Proc. of the IEEE International Conference on Intelligent Transportation Systems*, 2012, pp. 367-372.

[8] K.-W. Lin, Y.-L. Li, and M. Hashimoto, "Near-Future Traffic Evaluation based Navigation for Automated Driving Vehicles," in *Proc. of the IEEE Intelligent Vehicle Symposium*, 2017, pp. 1465-1470.

[9] Department of Transportation, Taipei City Government [Online]. Available: http://english.dot.gov.taipei/

[10] Kyoto Prefectural Police Headquarters [Online]. Available:http://www.pref.kyoto.jp/fukei/kotu/koki_j/jiko/hassei28.html

[11] City of Osaka [Online]. Available: hhttp://www.city.osaka.lg.jp/shimin/page/0000370379.html

[12] OpenStreetMap [Online]. Available: https://www.openstreetmap.org/

[13] T. Yamashita, K. Izumi, K. Kurumatani, and H. Nakashima, "Smooth Traffic Flow with a Cooperative Car Navigation System," *ACM AAMAS'05*, pp. 478-485.

[14] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA: MIT Press, 2009.

Table 1: The data of the maps of Kyoto, Osaka, and Taipei.

Name	Kyoto	Osaka	Taipei
Num. of Roads	164,349	347,449	212,289
Num. of Junctions	58,252	116,986	78,656
Population	1,468,879	2,694,731	2,703,829
Area	828 km^2	223 km^2	272 km^2

Table 2: Comparison between our algorithm and DNXNFE. Cruising time (hour) and mileage (kilometer) are shown.

Case	Num. of Query	Total Reduced Cruising Time (Improvement)	Total Reduced Mileage (Improvement)
Taipei_523k	523,053	134,605(53.99%)	1,691,270(25.17%)
Taipei_348k	523,053	6,574(8.13%)	170,063(4.87%)
Taipei_290k	290,821	4,115(6.25%)	117,599(4.08%)
Taipei_174k	174,225	8,588(19.21%)	58,888(3.44%)
Kyoto_342k	342,842	2,457(2.59%)	58,246(1.39%)
Kyoto_285k	285,755	1,839(2.33%)	40,040(1.13%)
Kyoto_171k	171,445	446(0.96%)	13,215(0.63%)
Osaka_347k	347,825	2,573(2.76%)	1,697(0.03%)
Osaka_289k	289,773	2,101(2.73%)	-1,409(-0.02%)
Osaka_173k	173,928	790(1.74%)	666(0.02%)

Table 3: Abbreviations used in line charts or straight bar charts.

Abbreviation	Meaning
Avg_Time_Pos_Imp	Average cruising-time improvement rate
Avg_Time_Neg_Imp	Average cruising-time deterioration rate
Avg_Mile_Pos_Imp	Average mileage improvement rate
Avg_Mile_Neg_Imp	Average mileage deterioration rate
Time_Pos_Num	Number of queries with improved cruising time
Time_Neg_Num	Number of queries with deteriorated cruising time
Mile_Pos_Num	Number of queries with improved mileage
Mile_Neg_Num	Number of queries with deteriorated mileage
Total	Total number of queries

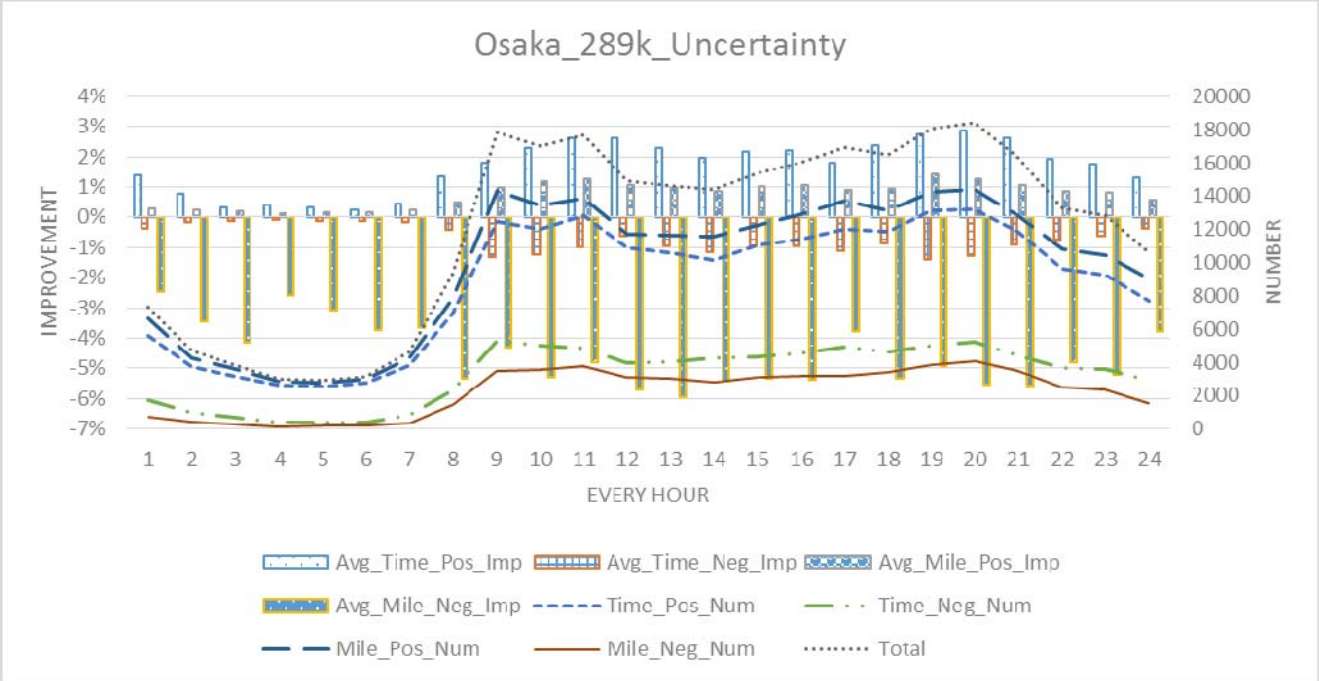


Fig. 4: Comparison between proposed algorithm and DNXNFE – The improvement in cruising time and mileage.

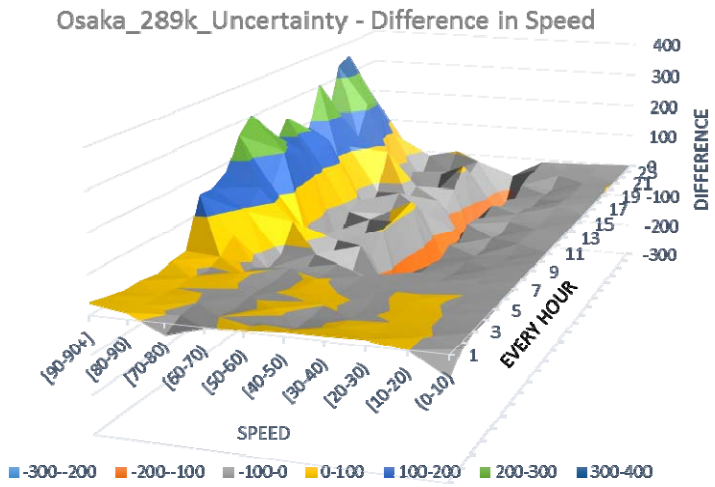


Fig. 5: Comparison between the proposed algorithm and DNXNFE - The difference in speed.

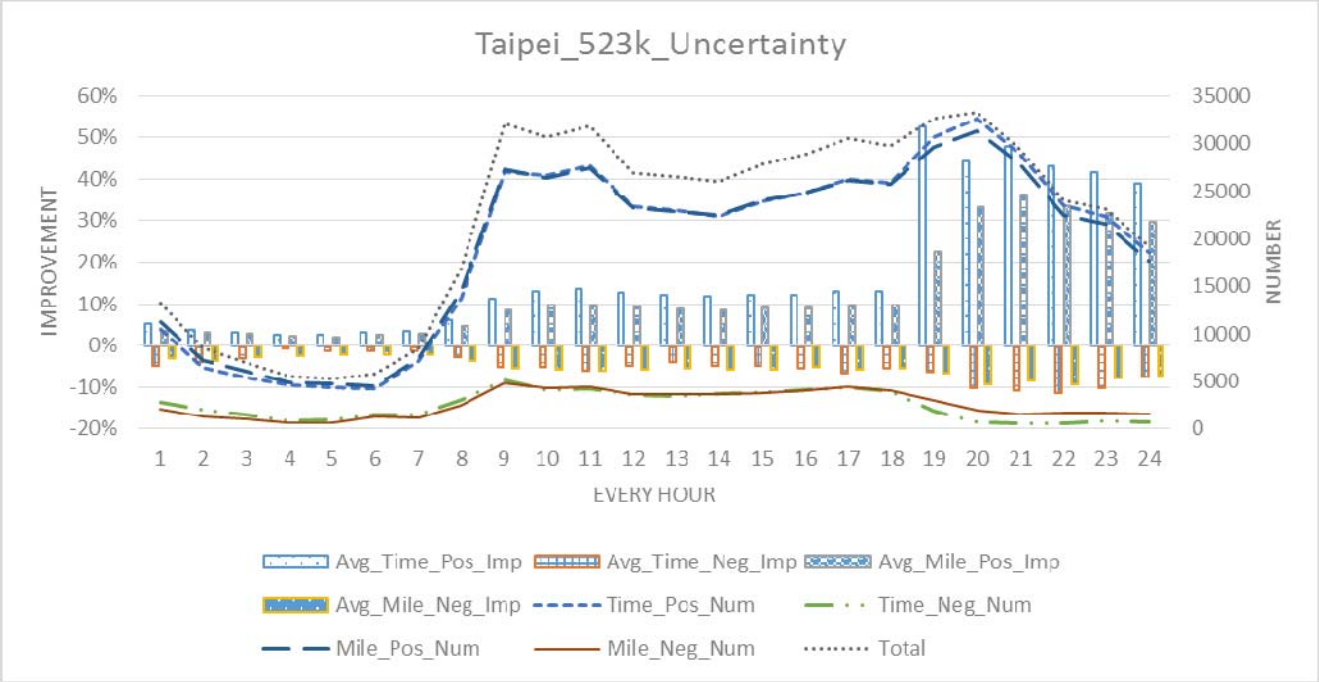


Fig. 6: The improvement in cruising time and mileage – for the biggest case with over 0.5 million vehicles.

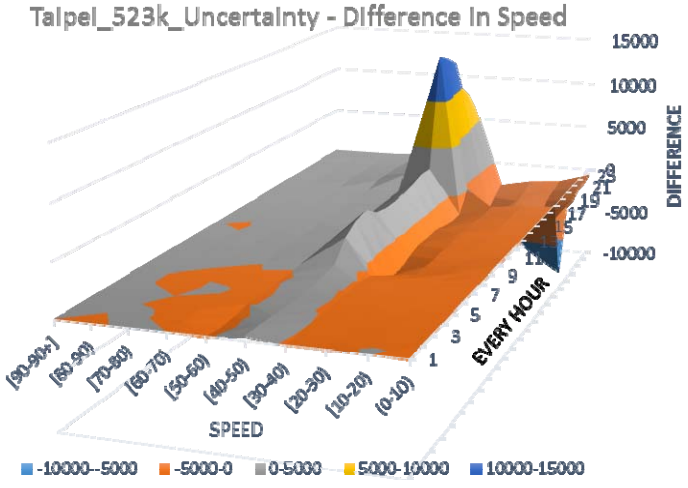


Fig. 7: The difference in speed – for the biggest case with over 0.5 million vehicles.