Peiqun LIN, Ph.D.[1]
(Corresponding author)
E-mail: pqlin@scut.edu.cn
Chuhao ZHOU, Ph.D. candidate[1]
E-mail: 201810101700@mail.scut.edu.cn
Yang CHENG, Ph.D.[2]
E-mail: cheng8@wisc.edu
[1] School of Civil Engineering and Transportation
South China University of Technology
Guangzhou 510641, Guangdong, China
[2] Wisconsin Traffic Operations and Safety Laboratory
University of Wisconsin Madison
Madison, WI 53706, USA

Intelligent Transport Systems
Original Scientific Paper
Submitted: 5 July 2021
Accepted: 14 Dec. 2021.

A SYSTEMATIC COOPERATION METHOD FOR IN-CAR NAVIGATION BASED ON FUTURE TIME WINDOWS

ABSTRACT

Traffic congestion has become a severe problem, affecting travellers both mentally and economically. To alleviate traffic congestion, this paper proposes a method using a concept of future time windows to estimate the future state of the road network for navigation. Through our method, we can estimate the travel time not only based on the current traffic state, but the state that vehicles will arrive in the future. To test our method, we conduct experiments based on Simulation of Urban MObility (SUMO). The experimental results show that the proposed method can significantly reduce the overall travel time of all vehicles, compared to the benchmark Dijkstra algorithm. We also compared our method to the Dynamic User Equilibrium (DUE) provided by SUMO. The experimental results show that the performance of our method is a little better than the DUE. In practice, the proposed method takes less time for computation and is insensitive to low driver compliance: with as low as 40% compliance rate, our method can significantly improve the efficiency of the unsignalised road network. We also verify the effectiveness of our method in a signalised road network. It also demonstrates that our method can assign traffic efficiently.

KEYWORDS

navigation; greedy algorithm; future time windows; dynamic traffic assignment; simulation.

1. INTRODUCTION

Despite a lot of research on alleviating traffic congestion and significant advances in technologies applied to improve transportation, we still waste a large amount of time in traffic jams. Traffic congestion affects the efficiency and safety of people's lives. It is also a major cause of environmental pollution. Hence, it is urgent to find an effective method for congestion mitigation.

Speaking of research on congestion mitigation, Dynamic Traffic Assignment (DTA) is a popular research field. Many studies proposed their algorithms to efficiently arrange vehicles [1–4], among which many follow Wardrop's principles [5]: User Equilibrium (UE) and System Optimal (SO).

There are two main types of approaches to address DTA problems: optimality modelling and traffic simulation. Studies using optimality modelling attempt to seek an appropriate algorithm to minimise the travel time of every user. Han et al. [6] analysed DUE by modelling Bounded Rational Dynamic User Equilibrium (BR-DUE) and Variable Tolerance Bounded Rational Dynamic User Equilibrium (VT-BR-DUE) since drivers do not always seek the best route or departure time. Hoang et al. [7] proposed a linear framework to solve the UE problem; however, it is only implemented in a single Origin-Destination (OD) network. On the other hand, the SO principle is about minimising the overall travel time of all users. Zhang and Qian [4] presented heuristic algorithms to solve SO-DTA problems in general road networks with multiple OD demands; they also considered indifferentiable total system costs (TC), which is generally considered differentiable. Lu et al. [8], unlike the traditional SO problem to minimise the TC, aim to minimise the emission. To address this issue, they proposed

an agent-based model to distinguish vehicle speeds to calculate the emissions. Although these methods can acquire an effective routing arrangement for vehicles, they are difficult to implement due to the computational cost and compliance requirements.

For traffic simulation, a widely adopted method is the Cell Transmission Model (CTM) [9, 10]. Compared to the aforementioned methods, the computational complexity of this method is lower. Yazici et al. [11] utilised the CTM and SO-DTA formulation as an underlying network model to simulate different incident management strategies. Zhu and Ukkusuri [12] employed the CTM with a non-holding back linear programming formulation to solve the SO-DTA problem. They also proved that their formulation could completely resolve the holding-back issue, which is a well-known problem for SO models [13]. Different from the CTM, the Link Transmission Model (LTM) can also solve the DTA problem. Long and Szeto [14] applied the LTM to address SO-DTA problems with and without First-In-First-Out (FIFO) constraints. However, its solution is NP-hard, which is challenging to solve when networks are large. Besides, the states of all cells or links need to be precisely calculated, which means that vehicles have to obey their navigation.

Neither optimality modelling nor CTM is suitable for in-car navigation, because of high computation cost and full compliance, but they still play an important role in understanding how to reduce congestion. Many navigation systems, such as Google, Baidu, and Gaode, usually use the shortest path algorithm, such as A* and Dijkstra [15, 16]. Some of them only consider the current link impedance (link travel time). Some may use both historical data and user data to predict travel time for navigation, but they provide the same information for all vehicles at a certain time, which is reactive [17]. These methods are useful when the road network is small or has a small number of vehicles. However, if the demand for navigation is simultaneously high, problems will emerge because vehicles may enter the same links on their assigned routes, which leads to increased traffic on those links and therefore dramatically changes the traffic condition in the road network.

Therefore, in this study, we aim to develop an in-car navigation method with a low computation cost that can make vehicles reasonably disperse in a road network to mitigate congestion, not requiring full compliance. The proposed is a systematic cooperation method based on the concept of future time windows, called Systematic Navigation with Future Time Windows (SN-FTW). To validate the proposed method, a simulation network using SUMO was implemented, in which a logit model was deployed to analyse the compliance rate. Specifically, our contributions are listed as follows.

- We propose a systematic cooperation method based on future time windows that can effectively avoid the situation in which many vehicles are assigned a route that may lead to traffic congestion.
- The proposed method is iterative and of low computational complexity, therefore, easy to implement.
- The proposed method performs well for various compliance rates, which is validated by the experiment.
- Our method can consider both signalised intersections and unsignalised intersections.

The remainder of our paper is structured as follows: Section 2 introduces the related work and section 3 introduces the assumptions, describes the method, and how to use future time windows. In Section 4, we present various experiments to verify the performance of our method. In the last section, we conclude and provide future directions for improving the effectiveness of our method.

2. RELATED WORKS

For in-car navigation, the shortest path algorithm is the popular method of decades ago, which aims to find a route with minimum travel distance or time. However, it is no longer suitable for today's traffic environment.

In this big data era, we can acquire a large amount of data collected from vehicles, such as speed and position [18, 19]. Therefore, we can analyse the laws of traffic to assign a better route for vehicles. For example, Zhang et al. [20] analysed spatial-temporal congestion patterns from taxi GPS data. In their research, they first used vehicle trajectory data for map matching and acquired other information, like speed. On this basis, they used clustering methods and correlation analysis to find the cause of congestion. If we can know what causes congestion, we can avoid it by traffic assignments. In the study of Kyriakou et al. [21], they not only analysed spatiotemporal data but employed a traffic prediction model and developed a navigation system. First, they predicted the traffic state of different roads. Next, they employed Dijkstra to

find an optimum route for vehicles. It is a satisfying way to navigate by analysing historical traffic patterns, but there is a problem. If we navigate too many vehicles by prediction or historical patterns, it means we have changed traffic patterns, so the patterns might be wrong for navigation.

To avoid this contradiction, reinforcement learning may be a good choice. Reinforcement Learning (RL) has been used successfully in many situations [22–24]. In recent years, some studies also employed it for in-car navigation. For instance, Zhou et al. [25] implemented the Bush-Mosteller reinforcement learning scheme to seek optimal travel routes. Koh et al. [26] used an improved Deep Q-Learning Network (DQN) to construct real-time smart vehicle navigation. Because the agent randomly explores the environment when training, the RL structure can store many situations that may not be encountered in the real world, which means it can contain more traffic patterns than the data from the real world. However, we need to rely on traffic simulation for randomly exploring, which means if the road network is too large, it will take a long time to simulate when training and it is difficult to converge to the optimal. Besides, if we change the traffic environment, like closing a road, the trained model may become invalid.

To be a more practical navigation system, rerouting is a useful strategy. Pan et al. [17] employed a traffic guidance system that can monitor traffic state and reroute vehicles when there are signs of congestion. The method of Lin et al. [27] is similar to the method of Pan et al. [17], but they used a different way to estimate traffic state and they required that all vehicles are automated. However, when a vehicle has been driving for some time, it may not be able to avoid driving through congested roads, because of no choice to reroute.

3. METHODOLOGY

3.1 Modelling for navigation issues

For a good traffic navigation strategy, vehicles should disperse in the road network to avoid traffic congestion, which means that the travel times of all vehicles are as low as possible. Therefore, we can formulate the problem as

$$\min S = \sum_{i=1}^{n} T\left(p_i^j\right), \quad j = 1,2,3,...,m_i \tag{1}$$

where S denotes the sum of all vehicles' travel time; p_i^j is the jth feasible route that is selected by the ith vehicle, which just enters the road network; n denotes the number of vehicles entering the road network; $p_i^j \in \boldsymbol{P_i} \subseteq \boldsymbol{P_{all}}$; $\boldsymbol{P_i}$ denotes all feasible routes that the ith vehicle can select; m_i denotes the number of routes in $\boldsymbol{P_i}$ and $\boldsymbol{P_{all}}$ is the set of all feasible routes between all ODs. Function T can calculate the travel time of the ith vehicle that selects the route p_i^j.

For every vehicle, they always want to select a route that can minimise its travel time, and if it can choose the optimal route, the traffic flow would distribute reasonably over the network. The optimal route of a vehicle can be described as

$$p_i^* = \underset{p_i^j}{\operatorname{argmin}}\, T\left(p_i^j\right) \tag{2}$$

where p_i^* denotes the optimal route for the ith vehicle. We convert our issue into finding the optimal path for each car. To reduce the complexity of our issue, we make some assumptions [28]:

1) If an intersection in the network is unsignalised, vehicles adopt the priority rule that a vehicle passes through this intersection first when the link it uses has higher priority; if an intersection is signalised, it must be fixed-time signals and we need to know its phases.
2) First-in first-out (FIFO) [29, 30]: The vehicle that enters a road link first leaves it first.
3) Causality [31, 32]: (I) For a vehicle on a road link, its speed and travel time are only related to the vehicles that enter the link before it. (II) For a vehicle that just enters the network, its optimal route is only affected by the vehicles that have entered the network. We can formulate this assumption as follows.

$$p_i^* = \underset{p_i^j}{\operatorname{argmin}}\, T\left(p_i^j \middle| p_1^*, p_2^*, \ldots, p_{i-1}^*\right) \tag{3}$$

4) The route for a vehicle, which obeys our navigation, is calculated when it enters the network and will not change during the trip.
5) We can obtain the number of vehicles on all links of the network from the detectors.
6) When a vehicle enters the road network, the compliance, if it follows the navigation, is determined; if not, its route choice follows the logit model according to the current network state.

The travel time of a vehicle is largely determined by the link impedance of its whole trip. However, the impedance of all links changes from the moment it enters to the moment it leaves. If we only use the link impedance when the vehicle enters the network

to calculate the optimal route, the traffic flow distribution may become uneven as vehicles gradually enter the network, which means that traffic congestion would emerge. Therefore, we need to predict the link impedance, which means that we want to know what the link impedance of a link at which a vehicle will arrive will be when that vehicle enters the network. We use e_l to denote a link in the road network, where $e_l \in \boldsymbol{E}$, $l \in [1,L]$, while $\boldsymbol{E}$ is the set of all links and L denotes the number of links in the road network. We denote this variable as t_l^h (link impedance of the e_l at time h). The link impedance is largely determined by the number of vehicles on the link and the properties of this link, such as the length and width of this link. We describe t_l^h as

$$t_l^h = R(\hat{x}_l^h | \theta_l) \tag{4}$$

where $\hat{x}_l^h$ denotes the estimated number of vehicles on the e_l at time h; and θ_l denotes the properties of the e_l, which is constant. The function R can calculate the link impedance according to $\hat{x}_l^h$ and θ_l. To calculate t_l^h more precisely, we need to divide a long road into many short links, because vehicles on a road may have an uneven distribution; if we disregard this possibility, we may calculate the wrong travel time for the traffic assignment. Therefore, e_l is a small section of a road.

According to the assumption of causality, we can also conclude that $\hat{x}_l^h$ is only related to the vehicles that have entered the road network, which means that when the kth vehicle enters the road network, we can calculate $\hat{x}_l^h(k)$ based on the k-1 vehicles that have entered the road network. We formulate this process as

$$\hat{x}_l^h(k) = M(l,h | p_1^*, p_2^*, \ldots, p_{k-1}^*) \tag{5}$$

where the function M can calculate $\hat{x}_l^h$ for the kth vehicle to acquire the optimal route according to the arrangement of the k-1 vehicles that have entered the road network.

3.2 Algorithm

Although we have formulated our calculation, it is difficult to calculate $\hat{x}_l^h$ in an analytical way. Nevertheless, according to our assumption, we can apply the greedy algorithm to calculate $\hat{x}_l^h$. The procedure can be described as follows:

1) Suppose there are k-1 vehicles on the road network at the time h_{k0}; the kth vehicle just enters, and we need to calculate the optimal route p_k^*.
2) Assuming that we already know $\hat{x}_l^h(k)$, we take a route p_k^1, which we assume is $(e_{l_1}, e_{l_2}, e_{l_3}, e_{l_4})$ from $\boldsymbol{P_k}$. Because we know $\hat{x}_{l_1}^{h_{k0}}(k)$, we can obtain the travel time $t_{l_1}^{h_{k0}}$ of e_{l_1} according to the function R. We can calculate the time that the kth vehicle enters e_{l_2}, which is $h_{k1} = h_{k0} + t_{l_1}^{h_{k0}}$. We can also obtain h_{k_2}, h_{k_3}, h_{k_4} in the same way, where h_{k_4} denotes the time when the vehicle leaves e_{l_4}. Therefore, we can obtain the travel time of the route p_k^1, which is $T(p_k^1) = h_{k_4} - h_{k_0}$. We can obtain p_k^* by calculating $T(p_k^j)$ for all $p_k^j \in \boldsymbol{P_k}$, which means that we can calculate the optimal route p_k^* for the kth vehicle according to $\hat{x}_l^h(k)$.
3) Update $\hat{x}_l^h$ according to the optimal route of the kth vehicle. Assuming that the optimal route is $(e_{l_1}, e_{l_2}, e_{l_3}, e_{l_4})$ and h_{k_0}, h_{k_1} h_{k_2}, and h_{k_3} denote the time that the vehicle enters the link, and h_{k_4} denotes the time that the vehicle leaves e_{l_4}. We know that the kth vehicle is on link e_{l_1} during $[h_{k_0}, h_{k_1}]$, so we can easily calculate $\hat{x}_{l_1}^{h^*}(k+1) = \hat{x}_{l_1}^{h^*}(k) + 1$ where $h^* \in [h_{k_0}, h_{k_1}]$. In the same way, we can calculate $\hat{x}_{l_2}^{h^*}(k+1)$, $\hat{x}_{l_3}^{h^*}(k+1)$, and $\hat{x}_{l_4}^{h^*}(k+1)$.
4) There are no vehicles on the road network when the first vehicle enters, so $\hat{x}_l^h(1) = 0, (l \in [1,L], h \in \boldsymbol{R}^+)$, which means that the vehicle can use a free-flow speed for all links.

Therefore, we can obtain $\hat{x}_l^h$ of any time and any link via the iteration. Besides, if an intersection is unsignalised, we have to consider its influences. Because the unsignalised intersection delay only occurs when there are conflicts, we can estimate it by calculating the conflicting flow rate. The modified formula can be described as

$$t_l^h = R(\hat{x}_l^h, v_{l_x}^h | \theta_l) \tag{6}$$

where $v_{l_x}^h$ denotes the conflicting flow rate at time h and l_x denotes the links that have traffic flows that conflict with e_l. If an intersection is signalised, we can calculate t_l^h as below.

$$t_l^h = (\varphi(x_l^{h_0} | \theta_l) + 1) R(\hat{x}_l^h | \theta_l) + \tau_l^h \tag{7}$$

where $\varphi(x_l^{h_0} | \theta_l)$ means the penalty which is between 0 and 1. If too many vehicles on the e_l at the current time h_0, vehicles may wait a long time and miss the green light, so we set a penalty to let them not choose the e_l as much as possible; τ_l^h denotes how much time vehicles need to wait on the e_l at the time h because of the red traffic light.

We can statically calculate a route for each vehicle, but if the estimate is wrong, we may cause congestion because of accumulative errors. Therefore, we need to adjust our estimate according to the current road network state. We can add a procedure for adjustment of estimation to our method because we can acquire the current network state.

5) Assuming that we have estimated $\hat{x}_{l_1}^{h}(k)$ and obtained the actual value $\hat{\hat{x}}_{l_1}^{h_c}(k)$ at the current time h_c, we can calculate $\delta = \hat{x}_{l_1}^{h_c}(k) - \hat{\hat{x}}_{l_1}^{h_c}(k)$. We can recalculate $\hat{x}_{l_1}^{h}(k)$ by δ. For instance, if δ=2, we can calculate how much time these two extra vehicles take on the e_{l_1} and adjust our time windows using a similar method, as shown in step 3.

The previously mentioned method is continuous, and inconvenient to realise for a computer. Therefore, we need to discretise it. We divide time into small time windows with the size t_w [40].

In *Figure 1*, we demonstrate how to use future time windows to calculate the best route for a vehicle. When Veh2 enters the network, there are two feasible routes (dashed line and solid line). According to the future time windows of these links, we should choose the solid line. We can observe that Veh1 will also drive through e_4, but when Veh2 arrives at e_4, Veh1 may have left this link at this time, which means that it will not influence Veh2.

Next, we need to consider that vehicles may not obey our navigation. For this issue, we use the following equation to simulate the vehicles that disobey our navigation according to assumption 6:

$$P\left(p_i^j = p_i^j\right) = \frac{\exp\left(T'\left(p_i^j\right)^{-1}\right)}{\sum_{1}^{m_i} \exp\left(T'\left(p_i^j\right)^{-1}\right)} \tag{8}$$

where P denotes the probability that the ith vehicle selects the route $p_i^{j'}$. The function T' is similar to the function T, but T' only uses $\hat{x}_l^{h_0}$ to calculate the travel time, which means that it only considers the network state when a vehicle enters. Therefore, we can acquire the routes that the vehicles select that disobey our navigation (these routes may be different from our system calculation). We can calculate $\hat{x}_l^{h}$ considering the vehicles that obey our navigation and those not.

Because the route set $\boldsymbol{P}_{all}$ is only related to the road network structure and OD, we can generate it offline to reduce the calculation cost for online navigation. However, if the road network is too large, it is also difficult to search on all feasible routes. Considering that few drivers want to detour too far, to reduce the complexity of calculation, we propose a method that is based on Depth-First-Search (DFS) with a break condition. The procedure is described as follows:

1) Calculate the shortest distance d_{min} of an OD using Dijkstra.

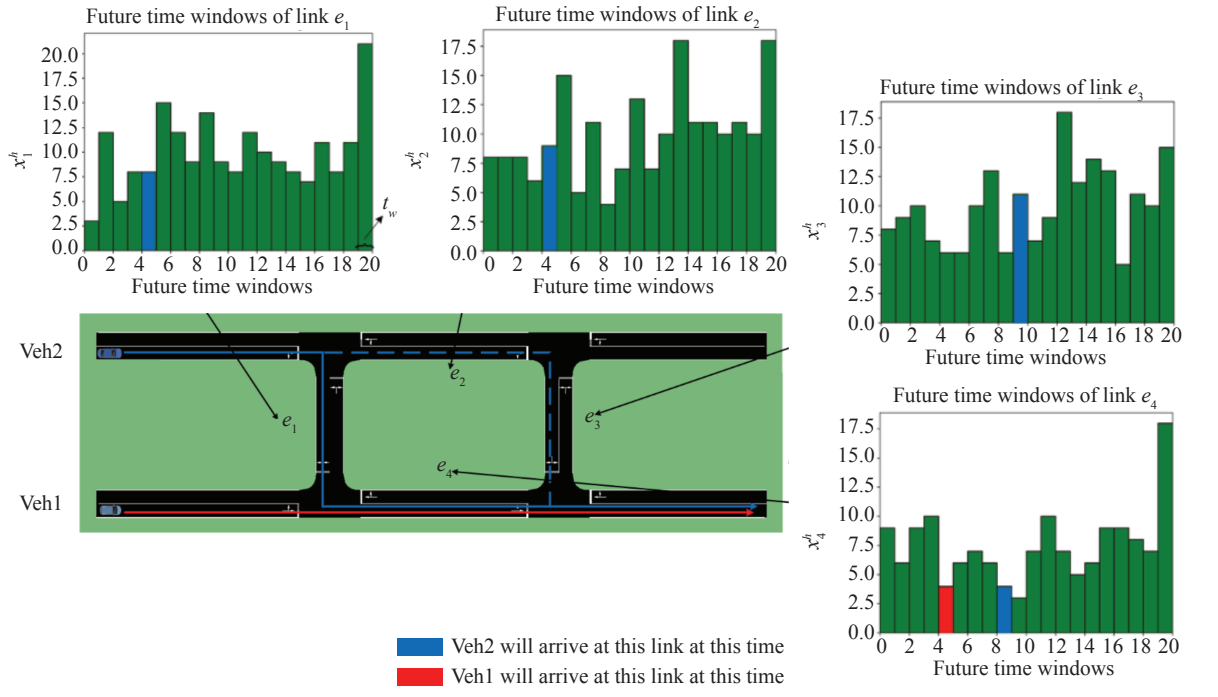


Figure 1 – Mechanism of the future time windows

2) Set the hyperparameter. The driving distance of a feasible route cannot exceed $\gamma \cdot d_{min}$.
3) Run the DFS. Assuming that the links that have been searched are e_{l_1},e_{l_2}, the driving distance has exceeded $\gamma \cdot d_{min}$; we then give up all routes that include e_{l_1},e_{l_2}.
4) Repeat steps 1–3 for all ODs

4. CASE STUDY

To verify the effectiveness of our method, we use SUMO to simulate our traffic navigation method [33, 34]. It can provide us with a substantial amount of information, such as vehicle speed and the number of halting vehicles (speed < 0.1m/s). Due to the traffic control interface (TraCI), which is a major feature of SUMO, we can easily use Python to implement our experiments. The experiments were conducted on a 64-bit Windows 10 machine with Intel Core i7-3770 (3.4GHz) and 16GB of memory.

4.1 Road impedance function and simulation parameters

In this paper, based on the Bureau of Public Roads (BPR) function, the road impedance function to calculate the travel time of a road [35] is expressed as

$$t_i = t_{i0} \cdot \left(1 + \alpha\left(\frac{N}{N_m}\right)^{\beta}\right) \tag{9}$$

where t_{i0} denotes travel time under free flow; N is the number of vehicles on a link at a certain time and N_m denotes the maximum number of vehicles that can exist on a link. According to our experiments, α=0.15 and β=4 are reasonable values. For unsignalised intersections, we need to adjust our road impedance because of the conflicting flow. According to the Highway Capacity Manual 2000 (HCM 2000), we can calculate the potential capacity via the conflicting flow rate. For simplicity, the final impedance function is expressed as

$$t_i = t_{i0} \cdot \left(1 + \alpha\left(\frac{N + \mu \sum l_x N_{l_x}}{N_m}\right)^{\beta}\right) \tag{10}$$

where μ, which falls between 0 and 1, denotes the proportion of vehicles from other links that will add to N. In this paper, we set μ=0.2, which is a suitable value obtained through our experiment. N_{l_x} is the number of vehicles on the e_{l_x}.

For the signalised intersections, we use the sigmoid function to calculate the penalty. The formula is expressed as

$$\varphi(N) = sigmoid\left(\frac{N}{N_m} \cdot a - b\right) \tag{11}$$

where a and b are used to adjust the input range to (-10, 10), so a=20 and b=10.

In our simulation, the free-flow speed of a vehicle at every link is 13 m/s, and the length of a vehicle is 5 m. We define N_m as the length of a link divided by the length of a vehicle. We use the Poisson distribution to model the vehicle's arrival [36, 37].

We built two road networks for experiments, one of which does not employ traffic lights and the other does. Vehicles adopt the priority rule in the road network without traffic lights.

4.2 Experiment 1: Road network without traffic lights

Introduction of road network

The road network (*Figure 2*) is based on a Manhattan road network, in which some roads are added as the entrances (X0, X1, X2, and X3) and exits (X4, X5, X6, and X7).

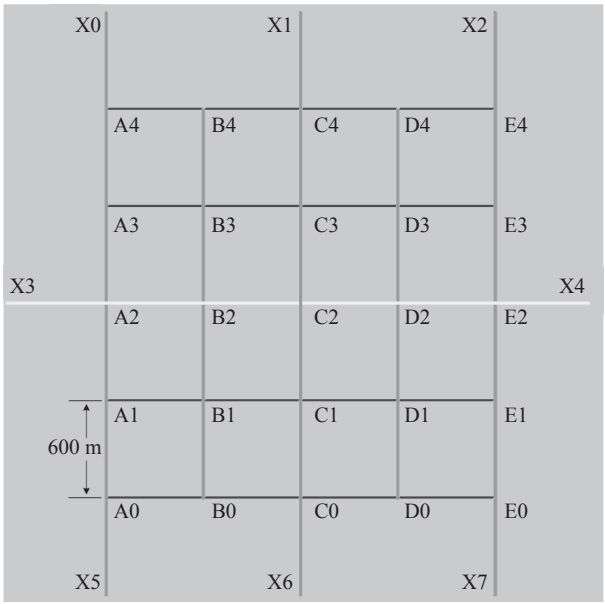


Figure 2 – Road network structure

All the intersections in this road network are unsignalised. The length of all roads is 600 m, and all roads are two-way four-lane roads. At an intersection, a vehicle can make turning movements in all directions except for a U-turn. It is a many-to-many network, which means that vehicles can drive from any of the entrances to any of the exits. The nodes (Ai, Bi, Ci, Di, and Ei) are intersections. We use two nodes to name a road, such

as A2A1, which means that a vehicle can drive from A2 to A1 via A2A1. The priority of the roads (yellow line) that straightly connect X3 and X4 is 3 (the larger the priority is, the more important this road is), while the priority of the blue lines, such as X0 to X5, is 2 and the priority of other lines is 1.

The total number of vehicles that will enter this road network is 8000 (approximately 2000 vehicles from each entrance), and the probability of a vehicle driving from an entrance to any exit is equal, which means, for example, that there will be approximately 500 vehicles from X1 to X5.

According to the distance between different entrances and exits, we use four different γ to calculate all the routes that we need. We set γ=2 for the 2400 m driving distance (disregarding intersections), such as X3 to X5, γ=1.4 for 3600 m, γ=1.2 for 4800 m and γ=1 for 6000 m.

Simulation results with different hyperparameters

As we previously mentioned, vehicles will enter following a Poisson distribution, and we conduct different simulations, where λ is 0.333, 0.25, and 0.2. We do not conduct our simulation for excessively large values of λ, such as 0.5, because vehicles that enter the network at 1 veh/s or 0.5 veh/s on average will cause the road network to become overburdened, and traffic congestion will be almost impossible to alleviate by our navigation. We test how hyperparameters influence our calculation results. *Figure 3* shows the average travel times of all OD pairs for different hyperparameters.

In *Figure 3*, labels in the legend denote the λ value in the Poisson distribution. t_w (unit is in seconds) in the x-axis denotes the size of the time window and n_l means how many links of a road we divide. The dashed lines across the figure denote the travel times for all vehicles that choose the shortest route (using Dijkstra), and different colours indicate different λ.

From *Figure 3*, we can observe that the travel times of vehicles via our navigation are lower than those of vehicles that choose the shortest route, and the performance is much better than that of Dijkstra when λ=0.333. The red dashed line shows that if vehicles always choose the shortest route, traffic congestion can easily occur, and our method can save more than 300 s. When λ is 0.25 or 0.2, our method is also better than Dijkstra, but not substantially better, because few vehicles run on the network, and almost all vehicles can drive in free-flow. The blue and green bars indicate that the effects of our hyperparameters are not obvious. The red bars demonstrate that the network may be overburdened in this situation, which is not easy for navigation, compared to the blue and green bars. We can discover that there are appropriate hyperparameters that can make our navigation more efficient and save driver time. We choose t_w=2 and n_l=3 for other experiments and then verify the performance with different λ.

To demonstrate the effectiveness of our method, we also compared it to the DUE from SUMO [38]. From *Figure 4*, we can observe that our method is a little better than the DUE, which means that our method is also similar to the DUE to some extent.

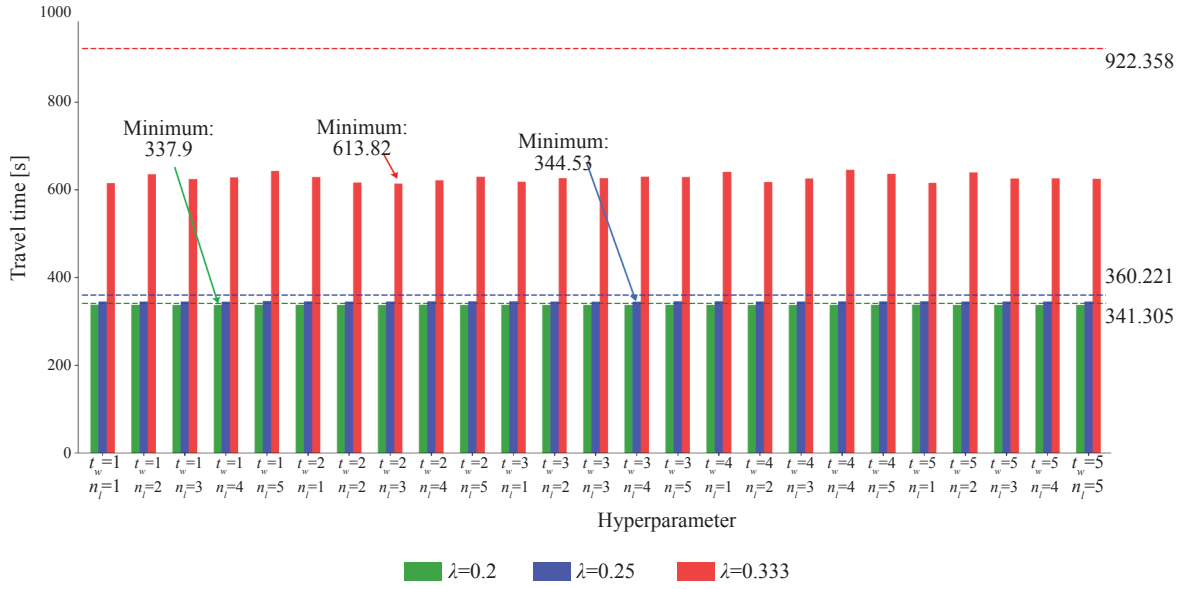
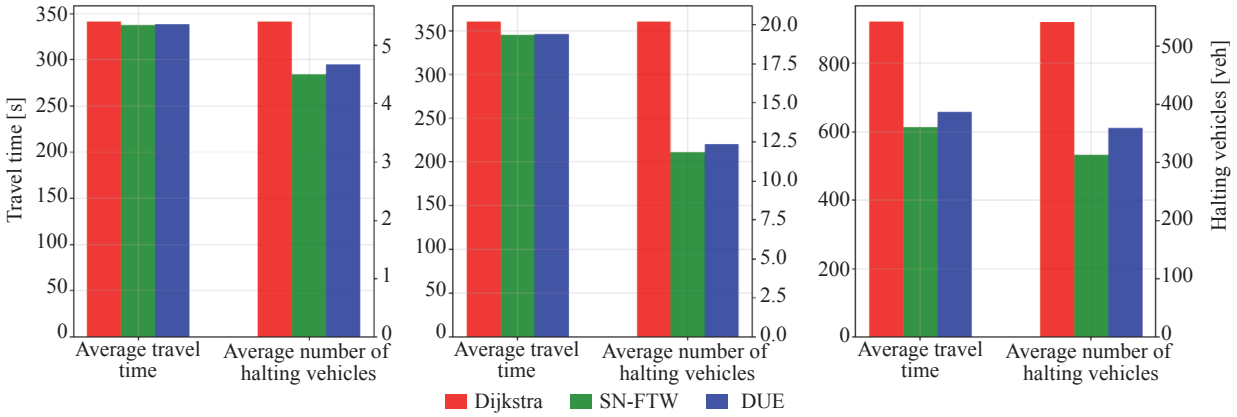


Figure 3 – Comparison of average travel time



a) λ=0.2 *b) λ=0.25* *c) λ=0.333*

Figure 4 – Comparison of average travel time and average number of halting vehicles

Detailed simulation results

Because we aim to achieve the system optimal, we may have to sacrifice some vehicles' benefit; therefore, we demonstrate the average travel times of different ODs.

From *Figure 5a*, we can observe that the performance of our method is almost the same as Dijkstra and DUE, which is consistent with our analysis. In *Figure 5b*, there is an obvious improvement between X0 and X5, compared to Dijkstra, although the average of all ODs does not differ substantially between the three methods, which shows that our method can arrange traffic better even though the traffic flow on the road network is minimal. *Figure 5c* shows a large difference between the three methods. We can observe that many OD pairs exhibit a large improvement when using our method.

We also notice that for some OD pairs, the performance of our method is worse and the difference of some ODs is greater than 100 s. As we previously mentioned, with a large number of vehicles, it is difficult to find a way for all vehicles to be assigned a route that can make their travel times reach a very low level. For the vehicles that enter late, because we have assigned many vehicles that entered early, the routes that we assign to them are the best in this situation, although it appears that they have long travel times. This phenomenon can also be observed in the DUE, such as X0 to X5 and X1 to X5.

We have already determined that our method can reduce many halting vehicles, but we need to check if it performs well during the whole simulation. Thus, *Figure 6* is shown to verify it.

From *Figures 6a–6c*, we can observe that the increase in the number of vehicles slows or stops at approximately 1000 steps. In *Figure 6c*, we observe that the maximum load of this network under this OD demand, which can rarely cause congestion, is approximately 500 vehicles (black circle); subsequently, if we continue to let vehicles enter the network at this flow rate (λ=0.333), the network will become increasingly congested. In *Figure 6c*, compared to Dijkstra, the average number of vehicles at every time step of our method declined by 25.2%. In *Figures 6a and 6b*, when the load is lower than 500, it can maintain a stable state. From *Figure 6f*, we can observe that the peak value decreases by nearly 600, compared to Dijkstra. As shown in *Figures 6d and 6e*, it is easy to determine that there are few halting vehicles on the road network during the whole simulation. In *Figure 6e*, we can also observe some peaks (black ellipses) when using Dijkstra, but not with our method or DUE. This finding reflects that if we let all vehicles select the shortest route according to the current network state, congestion easily occurs. Before the peak, the number of halting vehicles increases gradually, which means that traffic congestion does not emerge suddenly, and the seeds of problems have been planted when the wrong routes are assigned.

To explore more details of the running state of the road network, we chose some roads to demonstrate the number of vehicles on them during the simulation.

In *Figure 7*, we can observe that our method can distribute vehicles better. *Figure 7f* shows that congestion emerges on many roads because A3A2, B3B2, C3C2, D3D2, and E3E2 are feasible for entrances X0, X1, and X2, and due to the priority of

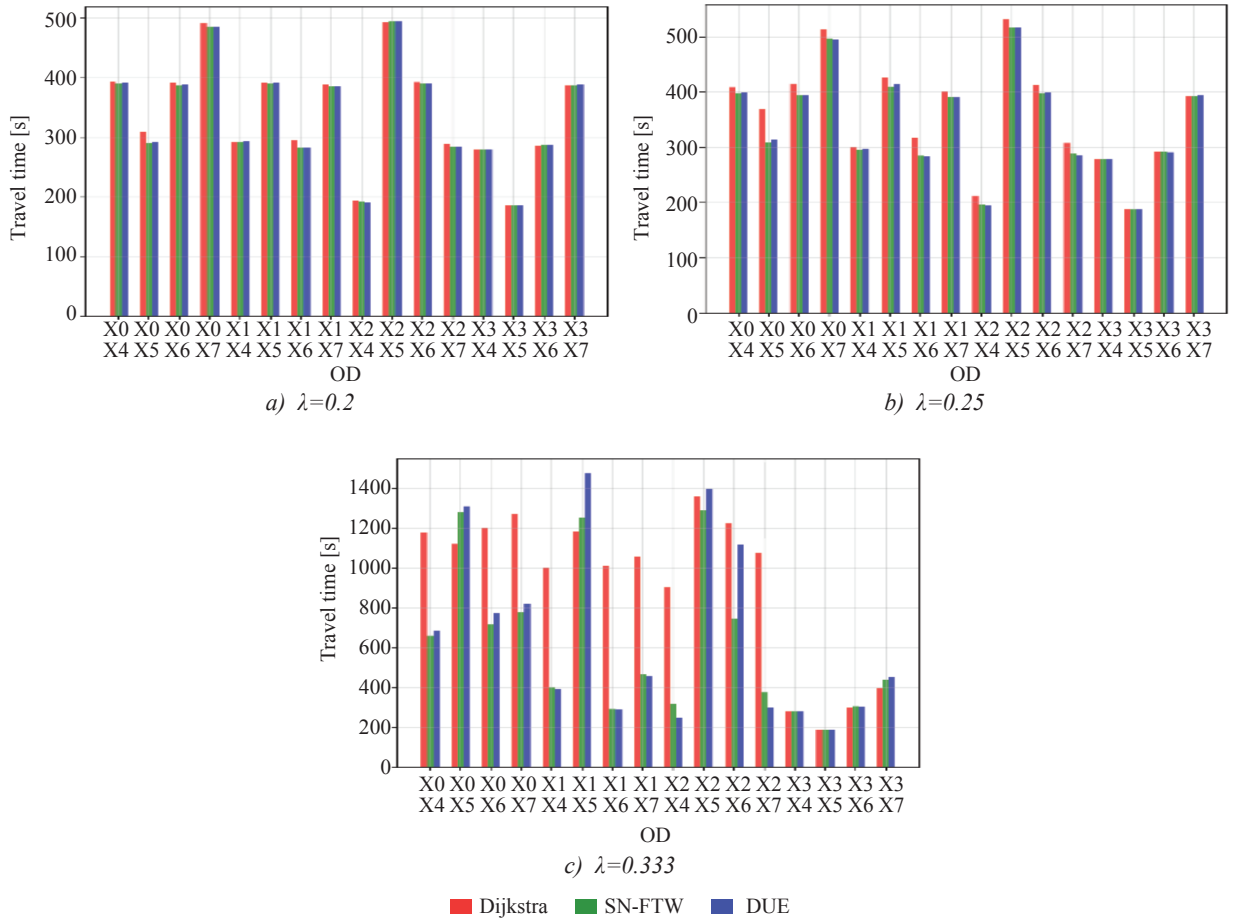


Figure 5 – Average travel time of different ODs using different lambda values

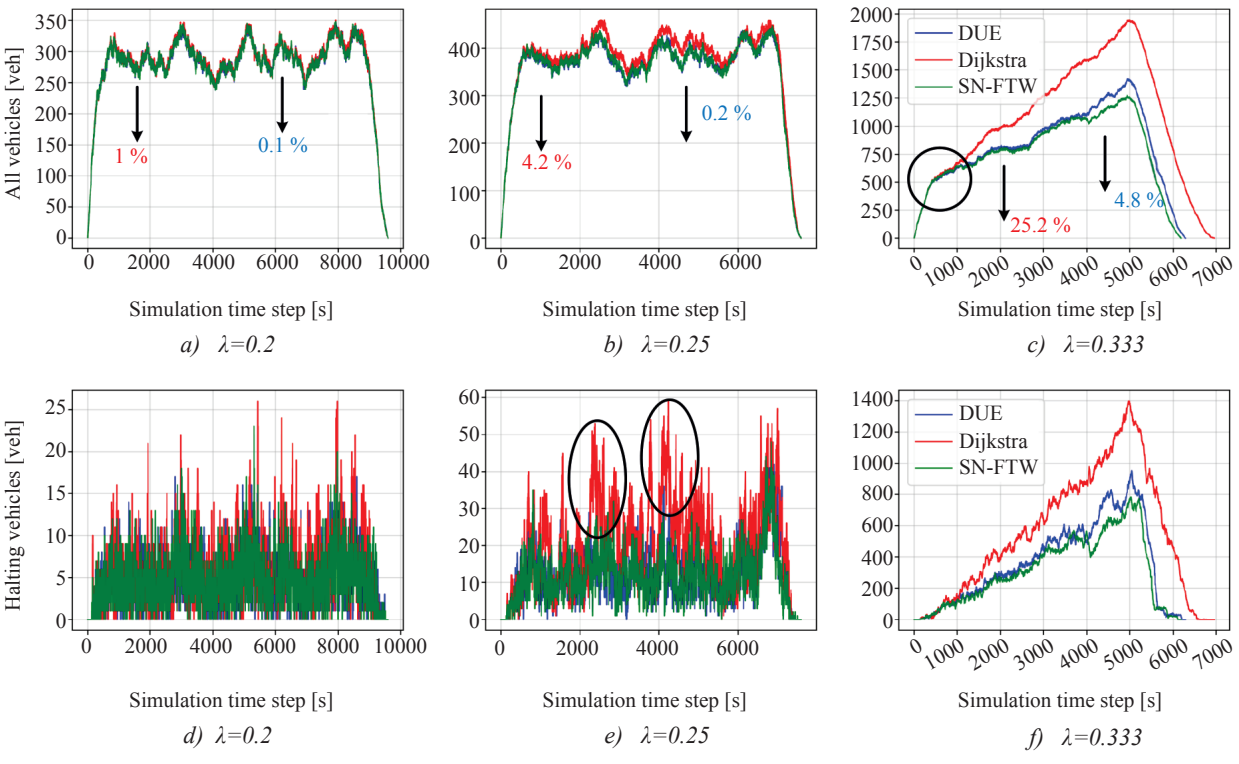


Figure 6 – Number of vehicles during the simulation (all vehicles and halting vehicles)

the roads, which is higher, (middle horizontal line in *Figure 2*) from X3 and X4, vehicles on roads such as A3A2 have to wait when conflicts occur. Some vehicles choose a road, such as A3A2, when they arrive at this road; then, the vehicles that just enter the network will choose another road, such as B3B2, and the same occurs for other roads. Therefore, the phenomenon shown in *Figure 7f* (two straight lines) easily emerges; when using Dijkstra, congestion will emerge on different roads at staggered times, which can also be observed in *Figure 7e*. We refer to this phenomenon as a congestion wave that transmits from link to link. However, in *Figure 7i*, our method almost chooses A3A2 for vehicles of which A3A2 is feasible, while the interesting thing is that we can observe a similar pattern in *Figure 7c*. Because A3A2 is a road along the shortest route from X0 to X5, compared to a detour such as via B3B2 and being congested on other roads, our method considers waiting on A3A2 a better choice.

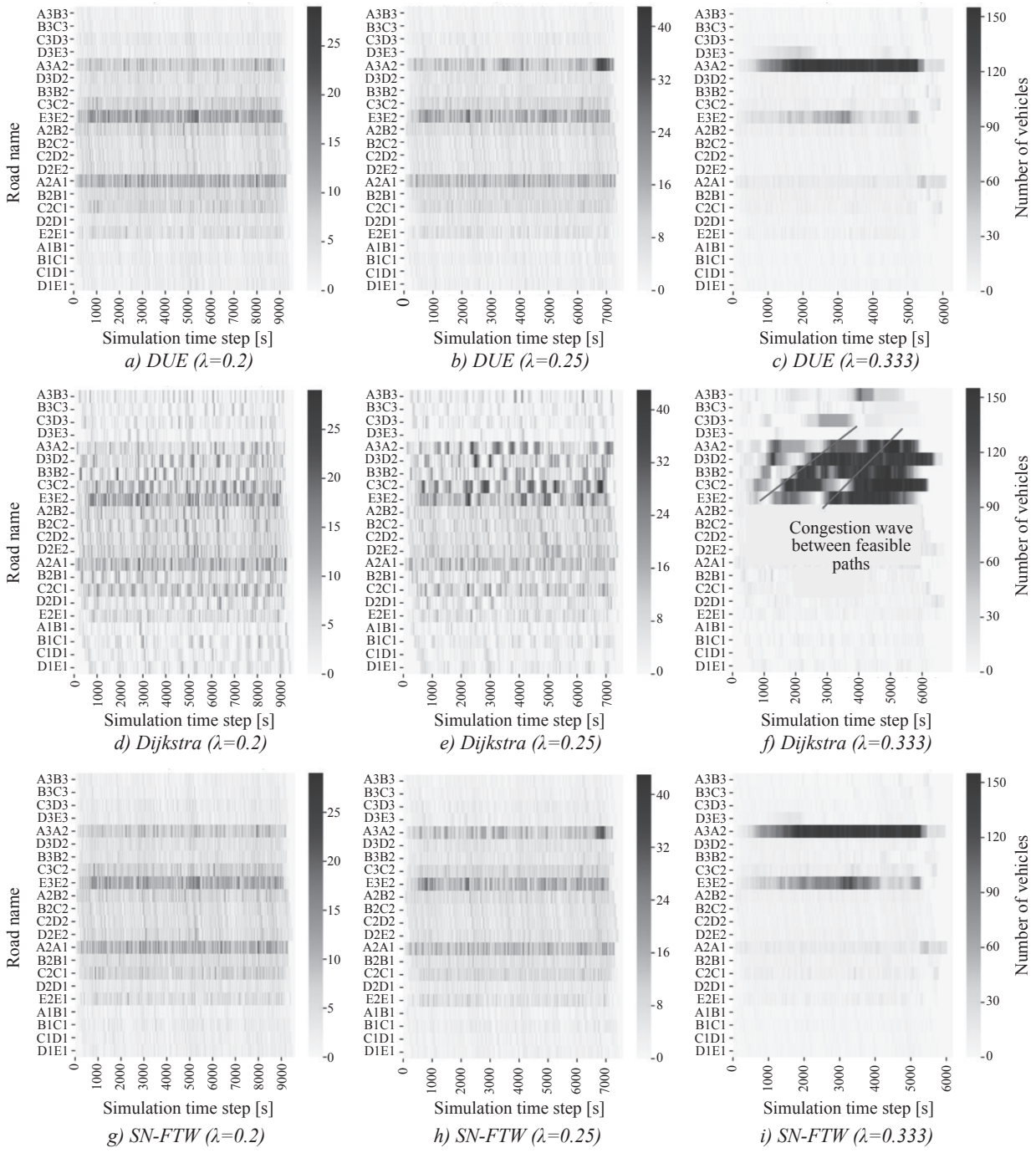


Figure 7 – Spatiotemporal pattern of the number of vehicles on some roads

In *Figure 8*, we demonstrate the driving distance of vehicles during the simulation using different routes to observe the severity of congestion. We choose the shortest routes from X0 to X5, X1 to X6, X2 to X7, and X3 to X4.

From *Figures 8a and 8e*, we observe that severe traffic congestion (triangles) occurred and lasted a long time. This occurrence corresponds to our analysis whereby we let many vehicles wait on A3A2. In *Figure 8i*, although only a few vehicles used A3A2, they still have to wait; and some vehicles waited a long time. However, as shown in *Figures 8f and 8g*, our method is much better; it can make vehicles run more efficiently. We also know that even though we assign fewer vehicles to a road, it may not make the traffic better, which means Dijkstra is not an efficient way to arrange traffic for navigation. We should let vehicles on the right road at the right time. *Figures 8d, 8h and 8l* show the influence of the highest priority that all vehicles from X3 to X4 are assigned the shortest route because vehicles from other roads have to wait for them to pass.

We also explored the performance of our method when some vehicles do not obey our navigation, which is shown in *Figure 9*.

From *Figure 9*, we notice that as more vehicles obey, the average travel time gradually decreases. In *Figures 9a and 9b*, the line is almost straight because all roads were not crowded during the simulation. When more vehicles follow the navigation instructions, the total travel time changes slightly. In *Figure 9c*, we can observe that the curve drops quickly at first, and when the ratio exceeds 40%, it declines slowly. If some vehicles can be assigned effectively (approximately

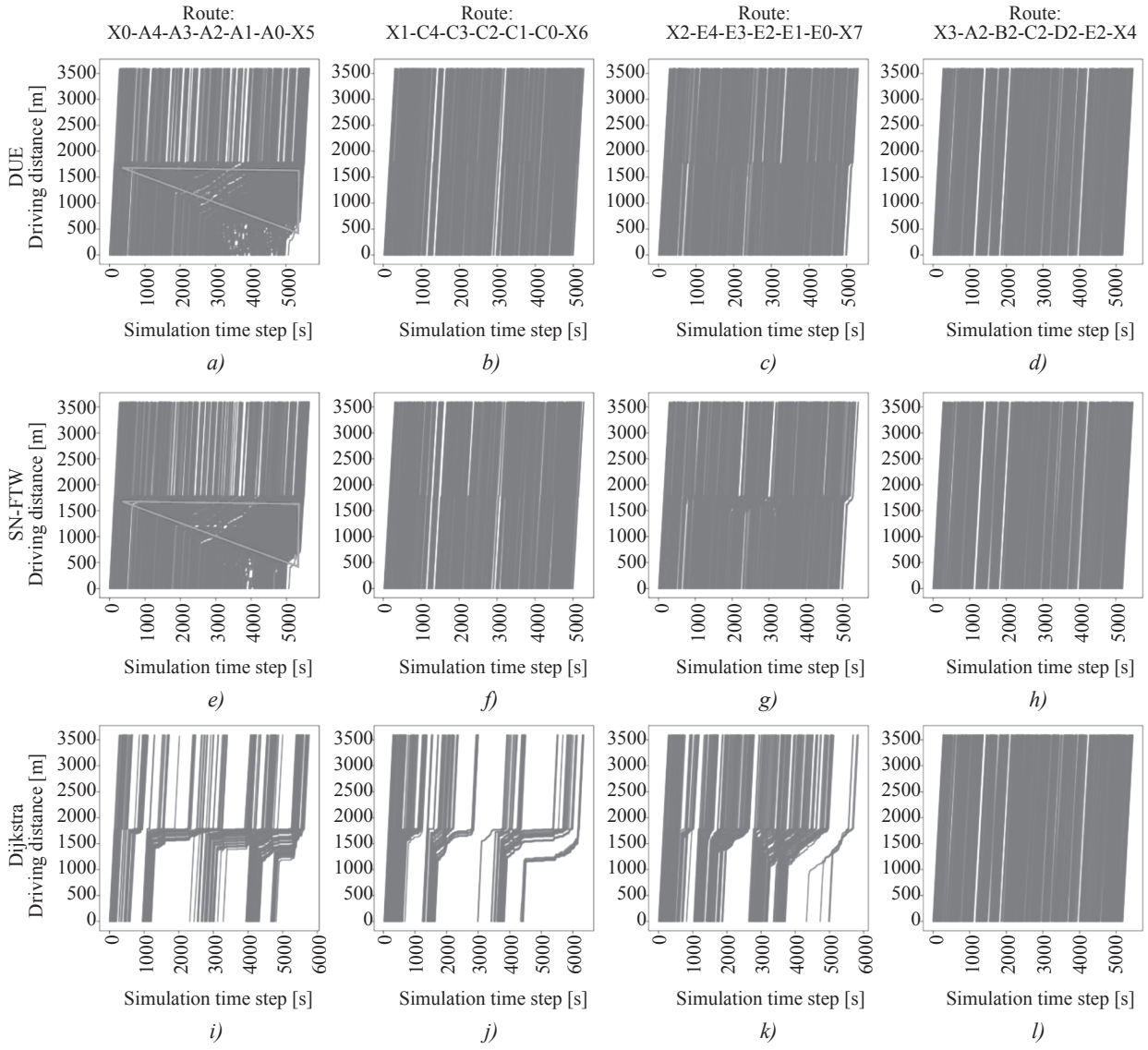
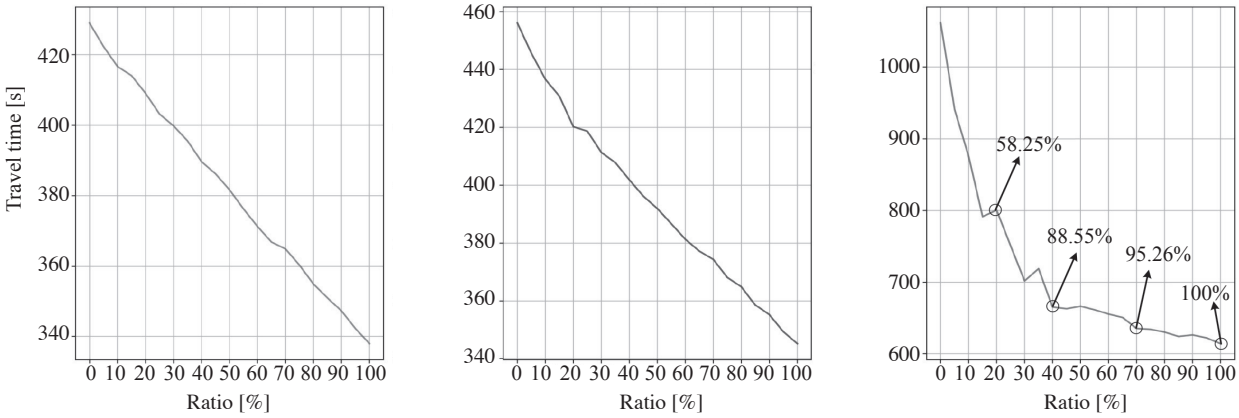


Figure 8 – Time-space diagram of vehicle trajectories of different routes (λ=0.333)



a) λ=0.2 *b) λ=0.25* *c) λ=0.333*

Figure 9 – Average travel time with different ratios of obedience

20%), the road network can run more efficiently. If 40% of vehicles can be assigned by our method, congestion will be greatly reduced.

4.3 Experiment 2: Road network with traffic lights

Introduction of road network

The road network (*Figure 10*) is a real-world network around Zhujiang New Town in Guangzhou, Guangdong, China, in which X1 and X3 are entrances; Y2 and Y5 are exits; then the others are both entrances and exits. All the intersections in this road network are signalised and we use Webster's method for traffic signal design [39].

The total number of vehicles that will enter this road network is 8000 (approximately 1000 vehicles from each entrance), and the probability of a vehicle driving from an entrance to any exit is also equal.

In this experiment, we set γ=3 for all ODs.

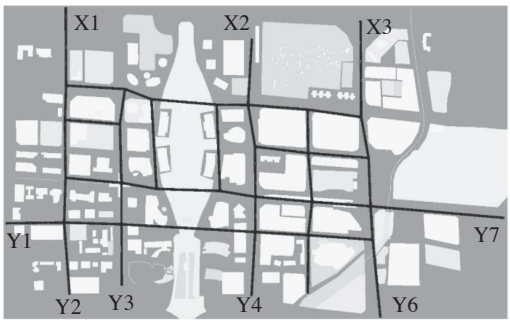
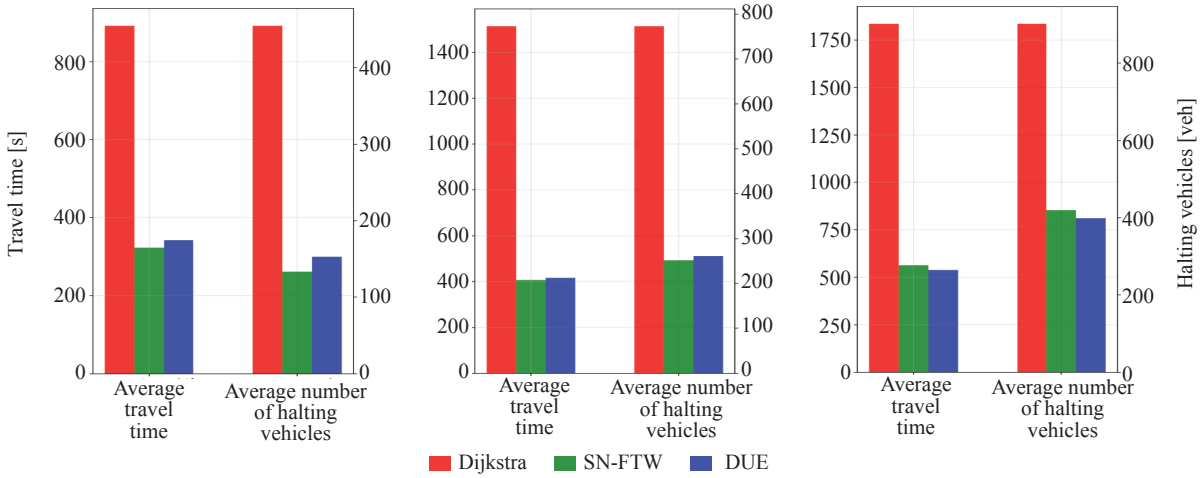


Figure 10 – The road network around Zhujiang New Town

Simulation results

First, we demonstrate the average travel time and the average number of halting vehicles among all the ODs in *Figure 11*. In this experiment, we compared the performance of the three methods when λ=0.13, 0.16, and 0.18. Because there are eight entrances, if λ is bigger, such as 0.3, the road network is easily overburdened. In *Figures 11a and 11b*, the average travel time of our method is the low-



a) λ=1.3 *b) λ=0.16* *c) λ=0.18*

Figure 11 – The comparison of average travel time and average number of halting vehicles in the signalised road network

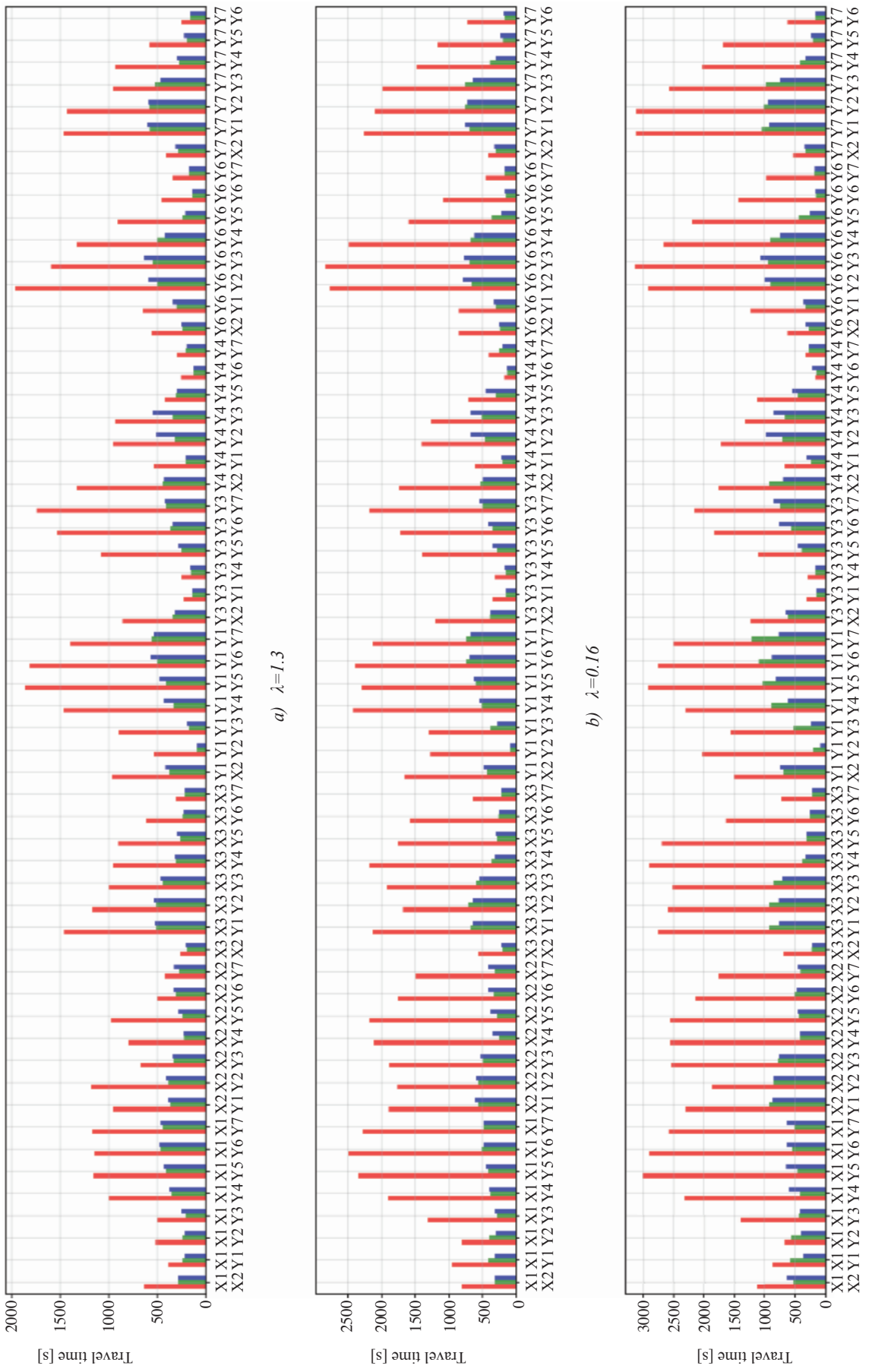


Figure 12 – The comparison of average travel time between different ODs in the signallised road nework

est, although the DUE is also satisfying, compared to Dijkstra. We can also observe that when λ=0.13, the average travel time of Dijkstra is about 900 s, but when λ=0.16, it is up to 1400 s. The reason is not only the number of vehicles becoming larger, but this method does not know the change of traffic lights, so it chooses wrong routes. In *Figure 11c*, the DUE is a little better than our method and we reckon it is because when there are more vehicles, our method has a little deviation of the estimation of traffic lights and the dispersing time of vehicle queues.

Next, the average travel time of different ODs is shown in *Figure 12*. In *Figures 12a and 12b*, the performance of our method among all ODs is almost the best, which reflects that our method can effectively arrange the traffic, even in a signalised road network. From *Figure 12c*, we can observe that in case of Y1 to Y2, Y3, Y4, Y5, Y6, Y7, our method does not perform well. In *Figure 10*, we can tell that the intersections near these exits are more easily congested than others and it may not be efficient to detour too far for vehicles entering from Y1, which verifies that there may be a slight deviation of our estimation that we mentioned above. We can also observe that even from Y1 to Y2, the performance of Dijkstra is very bad. Through the simulation, we found that due too many vehicles in the road network, Dijkstra could not assign traffic well, so the road network was paralysed and the vehicle queue reached the last intersection, which caused some vehicles to wait too long.

5. CONCLUSION

In this paper, we proposed an in-car navigation method to alleviate congestion and save the total travel time, not requiring full compliance. The experiments indicate that compared to Dijkstra, the method may take slightly more time for some of the OD pairs, but for most of them, it can save a considerable amount of time for drivers. Even for a 40% compliance rate in an unsignalised road network, the traffic congestion in the road network is still greatly alleviated. Compared to the DUE, our method can also obtain a better performance and in some experiments, we can observe that they have similar patterns.

This paper also points out the deficiencies of Dijkstra when it is used for navigation. Dijkstra is a satisfactory algorithm when a small number of vehicles operate on the road network. However, the algorithm's performance declines extremely fast when vehicles increase because it cannot consider a future situation in which a vehicle arrives at a road, and the traffic congestion before it may disappear. Because our method is iterative and needs less computational time, it can be deployed in practice in the real world.

The proposed method can also reduce the computational cost by obtaining a subset of all routes, which makes it easy to implement. We demonstrated the effect of future time windows, i.e., we can assign a route for a vehicle more reasonably. To accurately estimate future time windows, we use a viable approach to calculate how much time a vehicle needs at an intersection, whether signalised or unsignalised.

For future work, we will implement our method in a larger road network and consider dynamic signal timing. We also plan to use parallel computing to calculate the travel times of different routes in a route set for vehicles that simultaneously enter the road network when the road network becomes large.

ACKNOWLEDGEMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant 52072130 and U1811463, in part by the Science and Technology Program Project of Guangdong under Grant 2020A1515010349, and in part by the Fundamental Research Funds for the Central Universities of SCUT under Grant 2020ZYGXZR085.

林培群，周楚昊，程阳

一种基于未来时间窗的系统协同的车载导航方法

摘要

交通拥堵已经成为一个严重的问题，影响着司机的心理健康和经济。为了缓解交通拥堵，本文提出了一种利用未来时间窗的概念来估计路网未来状态的导航方法。通过该方法，我们不仅可以根据当前的交通状态，还可以根据车辆未来到达的状态来估计出行时间。为了验证我们的方法，我们基于*SUMO*软件进行了实验。实验结果表明，与基准算法*Dijkstra*相比，我们的方法能够显著降低所有车辆的总体行驶时间。我们还将我们的方法与*SUMO*提供的*DUE*算法进行了比较。结果表明，我们的方法的性能略优于*DUE*算法。在实际应用中，该方法计算时间短，且对低遵从率不敏感，遵从率低至*40%*，也能显著提高无信号路网的运行效率。我们也在一个信号化的道路网络中验证了我们的方法的有效性。结果表明，该方法依旧能够有效地分配流量。

关键词

导航；贪心算法；未来时间窗；动态交通分配；仿真

REFERENCES

[1] Melson CL, Levin MW, Hammit BE, Boyles SD. Dynamic traffic assignment of cooperative adaptive cruise control. *Transportation Research Part C: Emerging Technologies.* 2018;90: 114–133. doi: 10.1016/j.trc.2018.03.002.

[2] Pi X, Ma W, Qian ZS. A general formulation for multi-modal dynamic traffic assignment considering multi-class vehicles, public transit and parking. *Transportation Research Part C: Emerging Technologies.* 2019;104: 369–389. doi: 10.1016/j.trc.2019.05.011.

[3] Tajtehranifard H, et al. A path marginal cost approximation algorithm for system optimal quasi-dynamic traffic assignment. *Transportation Research Part C: Emerging Technologies.* 2018;88: 91–106. doi: 10.1016/j.trc.2018.01.002.

[4] Zhang P, Qian S. Path-based system optimal dynamic traffic assignment: A subgradient approach. *Transportation Research Part B: Methodological.* 2020;134: 41–63. doi: 10.1016/j.trb.2020.02.004.

[5] Wardrop JG. Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers.* 1952;1(3): 325–362. doi: 10.1680/ipeds.1952.11362.

[6] Han K, Szeto WY, Friesz TL. Formulation, existence, and computation of boundedly rational dynamic user equilibrium with fixed or endogenous user tolerance. *Transportation Research Part B: Methodological.* 2015;79: 16–49. doi: 10.1016/j.trb.2015.05.002.

[7] Hoang NH, Vu HL, Panda M, Lo HK. A linear framework for dynamic user equilibrium traffic assignment in a single origin-destination capacitated network. *Transportation Research Part B: Methodological.* 2019;126: 329–352. doi: 10.1016/j.trb.2017.11.013.

[8] Lu CC, et al. Eco-system optimal time-dependent flow assignment in a congested network. *Transportation Research Part B: Methodological.* 2016;94: 217–239. doi: 10.1016/j.trb.2016.09.015.

[9] Daganzo CF. The cell transmission model, part II: Network traffic. *Transportation Research Part B: Methodological.* 1995;29(2): 79–93. doi: 10.1016/0191-2615(94)00022-R.

[10] Daganzo CF. The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological.* 1994;28(4): 269–287. doi: 10.1016/0191-2615(94)90002-7.

[11] Yazici A, Kamga C, Ozbay K. Evaluation of incident management impacts using stochastic dynamic traffic assignment. *Transportation Research Procedia.* 2015;10: 186–196. doi: 10.1016/j.trpro.2015.09.068.

[12] Zhu F, Ukkusuri SV. A cell based dynamic system optimum model with non-holding back flows. *Transportation Research Part C: Emerging Technologies.* 2013;36: 367–380. doi: 10.1016/j.trc.2013.09.003.

[13] Carey M, Subrahmanian E. An approach to modelling time-varying flows on congested networks. *Transportation Research Part B: Methodological*. 2000;34: 157–183. doi: 10.1016/S0191-2615(99)00019-3.

[14] Long J, Szeto WY. Link-based system optimum dynamic traffic assignment problems in general networks. *Operations Research.* 2019;67(1): 167–182. doi: 10.1287/opre.2018.1775.

[15] Dijkstra EW. A note on two problems in connexion with graphs. *Numerische Mathematik.* 1959;1(1): 269–271. doi: 10.1016/0042-6989(66)90039-3.

[16] Hart PE, Nilsson NJ, Raphael B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics.* 1968;4(2): 100–107. doi: 10.1109/TSSC.1968.300136.

[17] Pan J, Sandu Popa I, Zeitouni K, Borcea C. Proactive vehicular traffic rerouting for lower travel time. *IEEE Transactions on Vehicular Technology*. 2013;62(8): 3551–3568. doi: 10.1109/TVT.2013.2260422.

[18] D'Andrea E, Marcelloni F. Incident detection by spatio-temporal analysis of GPS data. 2016 *IEEE International Conference on Smart Computing (SMARTCOMP)*. 2016. p. 1–5. doi: 10.1109/SMARTCOMP.2016.7501698.

[19] Xiao Q, He R, Ma C. The analysis of urban taxi carpooling impact from taxi GPS data. *Archives of Transport.* 2018;47(3): 109–120. doi: 10.5604/01.3001.0012.6514.

[20] Zhang K, Sun DJ, Shen S, Zhu Y. Analyzing spatiotemporal congestion pattern on urban roads based on taxi GPS data. *Journal of Transport and Land Use*. 2017;10(1): 675–694. doi: 10.5198/jtlu.2017.954.

[21] Kyriakou K, Lakakis K, Savvaidis P, Basbas S. Analysis of spatiotemporal data to predict traffic conditions aiming at a smart navigation system for sustainable urban mobility. *Archives of Transport.* 2019;52: 27–46. doi: 10.5604/01.3001.0014.0206.

[22] Mnih V, et al. Human-level control through deep reinforcement learning. *Nature.* 2015;518(7540): 529–533. doi: 10.1038/nature14236.

[23] Silver D, et al. Mastering the game of Go with deep neural networks and tree search. *Nature.* 2016;529(7587): 484–489. doi: 10.1038/nature16961.

[24] Silver D, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science.* 2018;362(6419): 1140–1144. doi: 10.1126/science.aar6404.

[25] Zhou B, Song Q, Zhao Z, Liu T. A reinforcement learning scheme for the equilibrium of the in-vehicle route choice problem based on congestion game. *Applied Mathematics and Computation.* 2020;371: 124895. doi: 10.1016/j.amc.2019.124895.

[26] Koh S, et al. Real-time deep reinforcement learning based vehicle navigation. *Applied Soft Computing.* 2020;96: 106694. doi: 10.1016/j.asoc.2020.106694.

[27] Lin KW, Hashimoto M, Li YL. Near-future traffic evaluation based navigation for automated driving vehicles considering traffic uncertainties. *2018 19th International Symposium on Quality Electronic Design (ISQED)*. 2018. p. 425–431. doi: 10.1109/ISQED.2018.8357324.

[28] Szeto WY, Lo HK. Dynamic traffic assignment: Properties and extensions. *Transportmetrica.* 2006;2(1): 31–52. doi: 10.1080/18128600608685654.

[29] Jayakrishnan R, Tsai WK, Chen A. A Dynamic traffic assignment model with traffic-flow relationships. *Transportation Research Part C: Emerging Technologies*. 1995;3(1): 51–72. doi: 10.1016/0968-090X(94)00015-W.

[30] Tong CO, Wong SC. A predictive dynamic traffic

assignment model in congested capacity-constrained road networks. *Transportation Research Part B: Methodological.* 2000;34(8): 625–644. doi: 10.1016/S0191-2615(99)00045-4.

[31] Carey M, Ge YE, McCartney M. A whole-link travel-time model with desirable properties. *Transportation Science.* 2003;37: 83–96. doi: 10.1287/trsc.37.1.83.12819.

[32] Drissi-Kaitouni O. A variational inequality formulation of the Dynamic Traffic Assignment Problem. *European Journal of Operational Research.* 1993;71(2): 188–204. doi: 10.1016/0377-2217(93)90048-R.

[33] Chen Y, Shafi SY, Chen Y. Simulation pipeline for traffic evacuation in urban areas and emergency traffic management policy improvements through case studies. *Transportation Research Interdisciplinary Perspectives.* 2020;7: 100210. doi: 10.1016/j.trip.2020.100210.

[34] Koh SS, et al. Reinforcement learning for vehicle route optimization in SUMO. *2018 IEEE 20th International Conference on High Performance Computing and Communications.* 2018. p. 1468-1473. doi: 10.1109/HPCC/SmartCity/DSS.2018.00242.

[35] Yang F, Yan X, Xu K. Evacuation flow assignment based on improved MCMF algorithm. *2008 First International Conference on Intelligent Networks and Intelligent Systems.* 2008. p. 637–640. doi: 10.1109/ICINIS.2008.70.

[36] Ahrens JH, Dieter U. Computer generation of poisson deviates from modified normal distributions. *ACM Transactions on Mathematical Software.* 1982;8: 163–179. doi: 10.1145/355993.355997.

[37] Ahrens JH, Dieter U. Computer methods for sampling from gamma, beta, poisson and bionomial distributions. *Computing.* 1974;12: 223–246. doi: 10.1007/BF02293108.

[38] DUE algorithm from SUMO. https://sumo.dlr.de/docs/Demand/Dynamic_User_Assignment.html#oneshot-assignment [Accessed 5th July 2021].

[39] Webster FV. *Traffic Signal Setting, Road Research Technical Paper No. 39*. London, England: Department of Scientific and Industrial Research; 1958.

[40] Pseudocode of our algorithm. https://github.com/DeusZero/navigation/tree/main [Accessed 5th July 2021].