Network-aware Multi-agent Reinforcement Learning for the Vehicle Navigation Problem

Fazel Arasteh*
arastehf@yorku.ca
EECS Department, Lassonde School of Engineering,
York University
Toronto, Ontario, Canada

Soroush SheikhGarGar
soroushs@eecs.yorku.ca
EECS Department, Lassonde School of Engineering,
York University
Toronto, Ontario, Canada

Manos Papagelis
papaggel@eecs.yorku.ca
EECS Department, Lassonde School of Engineering,
York University
Toronto, Ontario, Canada

ABSTRACT

Traffic congestion is characterized by longer trip times, and increased air pollution. In a static road network, the travel time to a destination is constant and can be computed using the shortest path first algorithm (SPF). However, road network conditions are dynamic, rendering the SPF to perform sub-optimally at times. In addition, in a realistic multiple-vehicle scenario, the SPF routing algorithm can cause congestion by routing all vehicles through the same shortest path. In this paper, we propose a network-aware multi-agent reinforcement learning model for addressing this problem. Our key idea is to assign an RL agent to intersections. Each RL agent operates as a router agent and is responsible for providing routing instructions to approaching vehicles. When a vehicle reaches an intersection, it submits a routing query to the RL agent consisting of its final destination. The RL agent generates a routing response based on (i) the destination, (ii) the current state of the road network, and (iii) routing policies learned by cooperating with other neighboring RL agents. Our experimental evaluation shows that the proposed MARL model outperforms the SPF algorithm by (up to) 20.2% in average travel time.

CCS CONCEPTS

• **Computing methodologies** → **Multi-agent reinforcement learning**; • **Applied computing** → *Transportation.*

KEYWORDS

multi-agent reinforcement learning, adaptive vehicle navigation, intelligent transportation systems, Graph Attention Network (GAT)

ACM Reference Format:
Fazel Arasteh, Soroush SheikhGarGar, and Manos Papagelis. 2022. Network-aware Multi-agent Reinforcement Learning for the Vehicle Navigation Problem. In *The 30th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '22), November 1–4, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/https://doi.org/10.1145/3557915.3561005

*The code for this research is publicly available at https://github.com/FazelYU/Adaptive-Navigation

SIGSPATIAL '22, November 1–4, 2022, Seattle, WA, USA

ACM ISBN 978-1-4503-9529-8/22/11.
https://doi.org/https://doi.org/10.1145/3557915.3561005

1 INTRODUCTION

Traffic congestion in urban road networks is a condition characterized by longer trip times, and increased air pollution. The emergence of new technologies like widely available internet connection, and GPS data has enabled algorithmic traffic navigation [6]. Currently, services like Google Maps[1] and Waze[2] help people with route planning mainly relying on a variant of the popular *Shortest Path First* (SPF) algorithm [2]. In a static network and for a single vehicle, the SPF algorithm is optimal. However, road network conditions are not always static. In a dynamic road network, the SPF path between an origin and a destination is harder to compute due to variable traffic conditions. The main approach to address this issue is to recursively breaking down the problem and estimating the travel time for smaller road segments, where the traffic conditions remain unchanged. This is usually referred to as the *traffic prediction problem*. Several methods have been proposed to address the *traffic prediction problem* of a road segment, such as moving average models, Support Vector Regression, and Random Forest. More recently, deep learning methods have been proposed to address the *traffic prediction problem* [9, 13]. Still, the estimated travel times, specifically long-term predictions, may be inaccurate, rendering the global SPF algorithm to be sub-optimal at times.

Another drawback of the SPF algorithm is that in multiple-vehicles scenarios, it will route every single vehicle through the currently available shortest path. As a result, due to the limited capacity of roads, the current shortest path gets quickly congested. Other methods, such as probabilistic dynamic programming [11] and ant colony optimization [8] have been proposed to directly route the vehicles in the dynamic network. More recently, deep reinforcement learning has also been proposed for end-to-end routing without individual road segment travel time prediction [3, 4, 7]. Moreover, graph convolution networks have been proposed to embed the structure of the road network and exploit together with reinforcement learning for routing in large dynamic networks [12]. To adequately capture the semantics and conditions of the problem that relates to its dynamic nature and optimization, we propose the *vehicle navigation problem*; given a dynamic road network and a fleet of vehicles, the objective is to minimize the overall travel time of all vehicles. The *vehicle navigation problem* is novel and it is different to the Shortest Path First problem; however it can be reduced to it for the simple case of a static graph and a single vehicle. The main idea of the *vehicle navigation problem* stems from the *packet routing problem* in the IP network [1]. It is easy to see the

[1]https://maps.google.com/
[2]https://www.waze.com/

analogy — vehicles are continuously routed from one intersection to another until they reach their destination, similarly to how IP packets are routed from one router to another. In our context, to address the *vehicle navigation problem*, we propose a *network-aware multi-agent reinforcement learning (MARL)* model. Specifically, in our MARL model, an RL agent is assigned to every intersection. When a vehicle approaches to an intersection, it submits a *routing query* to the agent, including its final destination. The agent generates a *routing response* based on the *vehicle's final destination* and the current *state of the road network traffic*. The vehicle follows the instructions of the routing response, including the next intersection. The process continues until the vehicle reaches its final destination. Note that the vehicle doesn't need to know its exact route to its destination.

2 PROBLEM DEFINITION

Consider a road network represented as a directed graph $W = \{I, R\}$, where $I = \{i_1, ..., i_N\}$ is a set of vertices that represent the intersections, and $R = \{r_1, ..., r_M\}$ is a set of edges that represent the roads. A road $r \in R$ is a directed edge from $r\text{-}head \in I$ to $r\text{-}tail \in I$. This assumption means that every road connects two intersections. Also, consider a set of L vehicles $VCs = \{vc_1, ..., vc_L\}$, and N router agents $U = \{u_1, ..., u_N\}$ each corresponding to an intersection. When a vehicle approaches an intersection it generates a routing query. We first define a routing query:

Definition 1. *q: **routing query** A query for routing at time t generated by vehicle vc that is currently driving in the road r_c, to the router u, the agent assigned to the intersection $r_c\text{-}tail$ [3], with the destination intersection i_d. t_{max} is the deadline for arriving.*

The router $q\text{-}u$ at the end of the current road of the vehicle generates a routing response to the routing query q. To define a routing response, we first define the next-hop roads set:

Definition 2. *$NH(r)$: **next-hop road set of road** r the set of all the outgoing roads from intersection $r\text{-}tail$ that are connected to r. We say that r_k is connected to r if the road network structure at intersection $r\text{-}tail$ allows the flow of traffic from r to r_k.*

Definition 3. *$resp(q)$: **routing response to query** q*

$$resp(q) = \begin{cases} < success >, & if\ q\text{-}r_c\text{-}tail == q\text{-}i_d \\ < fail >, & if\ q\text{-}t_{max} < q\text{-}t \\ < r \in NH(q\text{-}r_c) >, & otherwise \end{cases}$$

The routing response $resp(q)$ can be *success*, $fail$, or the next road to take. We next define the trip and the path for a vehicle:

Definition 4. *$trip$: **trip of vehicle** vc the trip of vehicle vc starting at time t from road r with a destination intersection i.*

We denote the set of all the trips as $Trips = \{trip | trip\text{-}vc \in VCs\}$.

Definition 5. *$path(trip)$: **path of** $trip$*

$path(trip) = (resp(q_1), ..., resp(q_z) =< success > / < fail >)$ is a sequence of routing responses for vehicle $trip\text{-}vc$.

[3]We use a short hyphen (-) to refer to an attribute of an object

We denote the length of path as $|path| = z$, and the last element of the path as $path_{|path|} = resp(q_z)$. We denote the set of all paths as $Paths = \{path(trip) | trip \in Trips\}$.

Definition 6. *$tt(p)$: **travel time of path** p is the difference between time of the last and fist queries of the path p.*

We next define the routing success of an episode to know about the number of vc that reached their destination. Also, we define the average travel time for all the vcs that reach their destination:

Definition 7. *RS: **Routing Success** is the the set of paths that end up in a $<$ success $>$ routing response.*

Definition 8. *$AVTT$: **average travel time** The average travel time of all the paths that end up in <success>.*

We now formally define the adaptive navigation problem:

Problem 1. ***Adaptive Navigation** Consider a locally accessible road network W, and a set of routing queries Q, our problem is to generate a response $resp(q)$ for each $q \in Q$ to (1) maximize $|RS|$ while (2) minimizing $AVTT$.*

3 METHODOLOGY

3.1 MARL Formulation

In this section we present our MARL methodology [14].

Router agent at intersection i, u_i. We assign a unique agent u_i to intersection $i \in I$. Agent u_i only responds to the queries $q \in Q$ with $q\text{-}r_c\text{-}tail == i$.

State of query q, s_q. The state of query q, is the unique representation of the destination intersection $[q\text{-}i_d]$.

State of intersection i, s_i^t. State of intersection i at time t consists of the traffic congestion condition in its outgoing roads. A road is considered congested ($C(r) == True$) if its current speed is smaller than a fixed portion of the free-flow speed of the road.

State of road network W at time t, s_W^t. The state of the road network W at time t is the concatenation of the states of all intersections.

State of query q at step τ, s_q^τ. Assume that τ is the index of q in the path of vehicle $q\text{-}vc$. State of query q at step τ is a tuple consisting of the state of the query and the state of the network at that time.

Action of agent u_i for s_q^τ, $a(s_q^\tau)$. Selecting one of the outgoing road-segments of the intersection $i = q\text{-}r_c\text{-}tail$.

Next state of s_q^τ, $s_q^{\tau+1}$. Assume that q' is the $\tau + 1$ query in path of vehicle $q\text{-}vc$.

Reward, $r(a(s_q^\tau))$. Assume that q' is the $\tau + 1$ query in path of vehicle $q\text{-}vc$:

$$\Delta T = (q'\text{-}t) - (q\text{-}t)$$
$$r(a(s_q^\tau)) = -\Delta T \quad (1)$$

3.2 Model Architecture

Figure 1 shows the architecture of the Adaptive Navigation algorithm. The destination of the routing query q, is forwarded to the agent $u_i = q\text{-}u$, and goes through its linear layer and produces embeddings of the destination IDs:

$$[s_q] = ReLU(Linear_i(s_q))$$

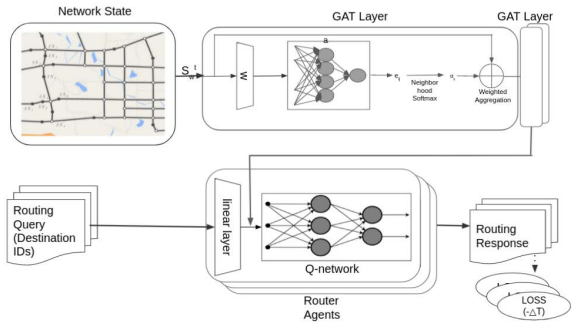


Figure 1: Adaptive Navigation Model Architecture

The network state s_W^t is passed to the GAT model as a graph signal and produces intersection state embeddings:

$$s_i' = (GAT(s_W^t))_i$$

The concatenation of $[s_q]$ and s_i' is passed to the Q-network of agent u_i to get the Q-values of the actions. The routing response is the action with the highest Q-value:

$$resp(q) = \underset{a}{\text{argmax}}\ Q_i(s_q^\tau, a)$$

We follow the conventional MSE loss (2) in the Q-learning algorithm per agent, where γ is the discount factor, and Q_{i+1} is the Q-network of the next agent. Note that the training of the agents is intertwined.

$$L(s_q^\tau, a, r : \theta) = \mathbb{E}[(r + \gamma \max_{a'} Q_{i+1}(s_q^{\tau+1}, a') - Q_i(s_q^\tau, a)^2] \quad (2)$$

Back-propagating the loss leads to end-to-end optimization of all the network parameters θ. Note that all agents contribute to the training of the GAT while only optimizing their own Q-network.

4 EXPERIMENTAL EVALUATION

4.1 Experimental Setup

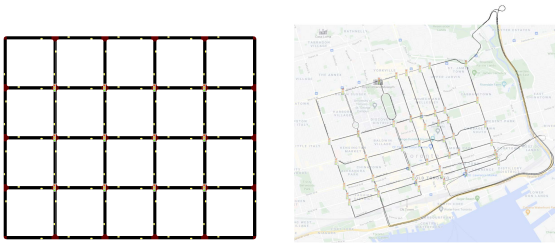
For the traffic simulation purposes, we used Simulation of Urban Mobility (SUMO) [5]. We used the SUMO-provided python API, TraCI, for interacting with the simulation. For our experimental evaluation, we used both *synthetic* and *realistic* networks with *synthetic traffic data*.

Synthetic Network. We considered a synthetic network based on the regular lattice network model that resembles a Manhattan-like urban road network. The lattice network is "regular" because each inner node has exactly the same number of edges (i.e., four in our case). Figure 2a shows the topology of our synthetic network, a 5x6 lattice/grid network, where 26 routing agents are assigned, one at each intersection (except for the corner ones).

Realistic Network. We considered the realistic road network of an urban area (downtown Toronto). We extracted the road network structure from Open Street Maps[4], and abstracted it by removing the less important roads (e.g., one-lane roads). At the end, we are left with 52 intersections that are each assigned a routing agent. Figure 2b shows the abstracted network of downtown Toronto.

Traffic Data. Since we don't have access to the detailed real-world traffic demands, we turn to use synthetic traffic demands consisting of a uniform demand, a biased demand, and stochastic congestion. **Uniform Demand:** the Uniform Demand is a set of origin-destination tuples distributed uniformly in the road network. **Biased Demand:** the Biased Demand is a set of user-defined origin-destination tuples. **Congestion:** we simulate congestion in a road by reducing its allowed speed by a fixed ratio. With a fixed period, we apply a network state change, in which every road is prone to congestion with a predefined probability.

[4]https://www.openstreetmap.org/copyright



(a) Synthetic network (5x6 grid) **(b) Downtown Toronto**

Figure 2: Road Network Datasets

	D.T Toronto	Grid
AN(h=2)	479.3	145.4
AN(h=1)	<u>476.4</u>	**138.4**
AN(h=0)	477.6	<u>143.7</u>
Q-routing	∞	159.6
SPF	551.7	173.4
SPFWR	**475.6**	205.1

Table 1: Testing Results: Average Travel Time (AVTT)

4.1.1 Baselines. We used three baselines for our evaluations:

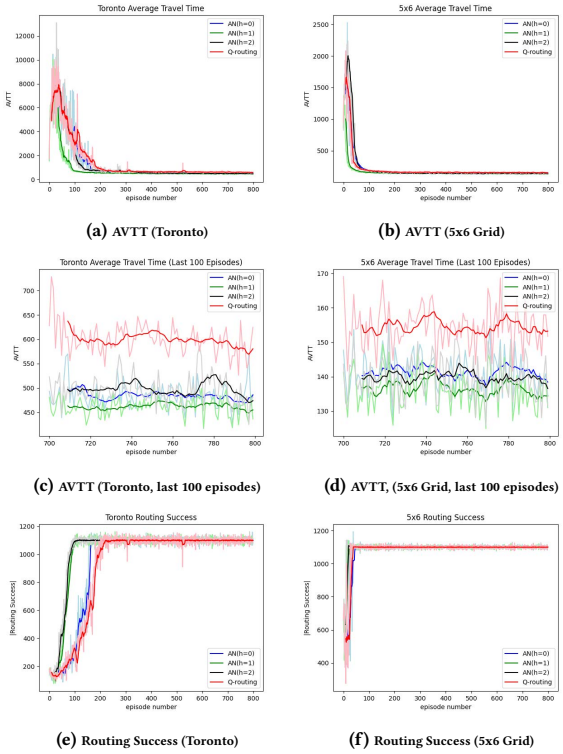
Travel Time Shortest Path First (SPF): This method finds the path with the shortest travel time in the current situation of the network using the SPF algorithm.

Travel Time Shortest Path First with Rerouting (SPFWR): This method is similar to the previous method. However, every time the vehicle reaches an intersection, we recompute the shortest path based on the new traffic situation.

Q-routing (QR) [1]: As another baseline, we implemented a deep reinforcement learning version of the Boyan, and Litman Q-routing for packet routing. This baseline is not aware of the network state.

These baselines are compared against three variants of our Adaptive Navigation method. **AN(h=0):** In this version, only the immediate intersection state is used as input to the router agents. **AN(h=1):** In this version, we apply one layer of GAT to the network state and then pass layer-1 embedding to the router agent. **AN(h=2):** In this version, we apply two layers of GAT to the network state and pass the layer-2 embedding to the router agent.

4.1.2 Evaluation Metric. Following the previous works [10] in traffic optimization, we use the average travel time defined in definition 8 as our primary evaluation metric.



(a) AVTT (Toronto) **(b)** AVTT (5x6 Grid)

(c) AVTT (Toronto, last 100 episodes) **(d)** AVTT, (5x6 Grid, last 100 episodes)

(e) Routing Success (Toronto) **(f)** Routing Success (5x6 Grid)

Figure 3: Average Travel Time (AVTT) and Routing Success during 800 training episodes for Toronto (a,b, and c) and 5x6 Grid (d,e,f)

4.2 Performance Evaluation

Training In the training phase, we define an episode as a fixed number of simulation time steps. We don't set a deadline for arrival. For every episode we report, the Routing Success (number of vehicles that reached their destination during this episode), and Average Travel Time (for vehicles that reached their destination). Figures 3a and 3b, show the training results for 800 episodes of training for Downtown Toronto and 5x6 Grid network respectively. Figures 3c, 3d show the average travel time in the final episodes. In figures 3a,3c,3b,3d the Y-axis shows the average travel time in seconds, and the X-axis shows the episode number. In figures 3e, and 3f the Y-axis shows the number of vehicles that have successfully arrived at their destination in that episode (|Routing Success|), and the X-axis shows the episode number.

Testing for testing purposes, to avoid the random noises, instead of having a flow, we predefined 2000 uniform trips. An episode in the testing phase lasts as long as all the vehicles reach their destination. Table 1 shows the average results of 5 episodes in the testing phase. In the Downtown Toronto network, SPFWR has the best performance, and AN(h=1) marginally stands in second place. **It is important to note that the goal is not to outperform the SPFWR**. SPFWR needs to compute all-pairs-shortest-path in every time-step rendering it to be computationally infeasible if the response time is important. In this experiment, Q-routing fails to route all the vehicles successfully since it creates infinite loops. In the 5x6 grid network, AN(h=1) outperforms the other algorithms. Counter-intuitively, SPFWR has the worst performance in this case. The network structure and its traffic capacity, e.g., number of lanes per road play an important role in the SPFWR performance. A road that has only one lane can easily get congested. SPFWR greedily sends all the vehicles to the current shortest path and congests it so that it is no longer the shortest path. Moreover, SPFWR does not consider the waiting times in the traffic lights queues. Current experiments show that adding an extra layer of GAT does not necessarily improve the results since the extra layer increases the model parameters leading to the performance decay.

5 CONCLUSIONS

The SPF algorithm is the predominant algorithm for routing in a static network. However, it proves sub-optimal in a dynamic network. In this paper, we introduced the adaptive navigation problem. Specifically, we assigned a Q-learning agent to every intersection, which is responsible for the routing of all incoming vehicles to that intersection. The router agent uses the destination ID of the approaching vehicle, and the traffic state of its neighborhood for determining the next intersection the vehicle should be forwarded to. Our empirical evaluation demonstrated a substantial improvement of **up to 20.2%** in average travel time, compared to the SPF-based baselines, for both synthetic and realistic networks.

REFERENCES

[1] Justin A Boyan and Michael L Littman. 1994. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in neural information processing systems*. 671–678.
[2] Edsger W Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische mathematik* 1, 1 (1959), 269–271.
[3] Yuanzhe Geng, Erwu Liu, Rui Wang, Yiming Liu, Weixiong Rao, Shaojun Feng, Zhao Dong, Zhiren Fu, and Yanfen Chen. 2021. Deep Reinforcement Learning Based Dynamic Route Planning for Minimizing Travel Time. In *2021 IEEE International Conference on Communications Workshops)*. IEEE, 1–6.
[4] Songsang Koh, Bo Zhou, Hui Fang, Po Yang, Zaili Yang, Qiang Yang, Lin Guan, and Zhigang Ji. 2020. Real-time deep reinforcement learning based vehicle navigation. *Applied Soft Computing Journal* 96 (11 2020), 106694.
[5] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. 2018. Microscopic Traffic Simulation using SUMO. In *The 21st International Conf. on Intelligent Transportation Systems*. IEEE.
[6] Arzoo Miglani and Neeraj Kumar. 2019. Deep learning models for traffic flow prediction in autonomous vehicles: A review, solutions, and challenges. *Vehicular Communications* 20 (2019), 100184.
[7] Aleksandr I Panov, Konstantin S Yakovlev, and Roman Suvorov. 2018. Grid path planning with deep reinforcement learning: Preliminary results. *Procedia computer science* 123 (2018), 347–353.
[8] Bogdan Tatomir, Leon JM Rothkrantz, and Adriana C Suson. 2009. Travel time prediction for dynamic routing using ant based control. In *Proceedings of the 2009 Winter Simulation Conference (WSC)*. IEEE, 1069–1078.
[9] David Alexander Tedjopurnomo, Zhifeng Bao, Baihua Zheng, Farhana Choudhury, and AK Qin. 2020. A survey on modern deep neural network for traffic prediction: Trends, methods and challenges. *IEEE Transactions on Knowledge and Data Engineering* (2020).
[10] Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. 2019. A survey on traffic signal control methods. *arXiv preprint arXiv:1904.08117* (2019).
[11] Lin Xiao and Hong K. Lo. 2014. Adaptive vehicle navigation with stochastic traffic information. *IEEE Transactions on Intelligent Transportation Systems* (2014).
[12] Jiaming Yin, Weixiong Rao, and Chenxi Zhang. 2021. Learning Shortest Paths on Large Dynamic Graphs. In *2021 22nd IEEE International Conference on Mobile Data Management (MDM)*. 201–208. https://doi.org/10.1109/MDM52706.2021.00040
[13] Xueyan Yin, Genze Wu, Jinze Wei, Yanming Shen, Heng Qi, and Baocai Yin. 2021. Deep Learning on Traffic Prediction: Methods, Analysis and Future Directions. *IEEE Transactions on Intelligent Transportation Systems* (2021).
[14] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2021. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control* (2021), 321–384.