Transportation Research Part C 137 (2022) 103560

Contents lists available at ScienceDirect

Transportation Research Part C

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

Multi-agent reinforcement learning for Markov routing games: A new modeling paradigm for dynamic traffic assignment

Zhenyu Shou[a], Xu Chen[a], Yongjie Fu[a], Xuan Di[a,b,*]

[a] *Department of Civil Engineering and Engineering Mechanics, Columbia University, United States of America*
[b] *Data Science Institute, Columbia University, United States of America*

A R T I C L E I N F O

Keywords:
Markov game
Multi-agent reinforcement learning (MARL)
Atomic agents
Dynamic user equilibrium (DUE)

A B S T R A C T

This paper aims to develop a paradigm that models the learning behavior of intelligent agents (including but not limited to autonomous vehicles, connected and automated vehicles, or human-driven vehicles with intelligent navigation systems where human drivers follow the navigation instructions completely) with a utility-optimizing goal and the system's equilibrating processes in a routing game among atomic selfish agents. Such a paradigm can assist policy-makers in devising optimal operational and planning countermeasures under both normal and abnormal circumstances. To this end, we develop a Markov routing game (MRG) in which each agent learns and updates her own en-route path choice policy while interacting with others in transportation networks. To efficiently solve MRG, we formulate it as multi-agent reinforcement learning (MARL) and devise a mean field multi-agent deep Q learning (MF-MA-DQL) approach that captures the competition among agents. The linkage between the classical DUE paradigm and our proposed Markov routing game (MRG) is discussed. We show that the routing behavior of intelligent agents is shown to converge to the classical notion of predictive dynamic user equilibrium (DUE) when traffic environments are simulated using dynamic loading models (DNL). In other words, the MRG depicts DUEs assuming perfect information and deterministic environments propagated by DNL models. Four examples are solved to illustrate the algorithm efficiency and consistency between DUE and the MRG equilibrium, on a simple network without and with spillback, the Ortuzar Willumsen (OW) Network, and a real-world network near Columbia University's campus in Manhattan of New York City.

1. Introduction

When one chooses a path dynamically while navigating a road network, others may also do so to compete for limited road resources. Therefore it is crucial to model the dynamic routing choices of many drivers simultaneously. This paper aims to develop a game-theoretic paradigm that models one's learning behavior and the system's equilibrating processes in a dynamic routing game among atomic selfish agents, while accounting for competition among drivers.

While traveling from an origin to a destination, one needs to select a sequence of road segments. Thus, en-route path choice is inherently a sequential decision-making process, in which at each junction one decides what the next link to take. Markov decision processes (MDP) (Puterman, 1994) and reinforcement learning (RL) (Sutton and Barto, 1998) have become popular tools to model such a sequential process, in which travelers are rational agents to optimize a prescribed objective function (or reward) (Seongmoon Kim et al., 2005; Kim et al., 2016). In literature, recently there is a growing body of research using RL based

[*] Corresponding author at: Department of Civil Engineering and Engineering Mechanics, Columbia University, United States of America.
E-mail address: sharon.di@columbia.edu (X. Di).

https://doi.org/10.1016/j.trc.2022.103560
Received 7 November 2020; Received in revised form 21 December 2021; Accepted 7 January 2022
Available online 10 February 2022

approaches on route choice behavior. We will first review the literature on route choice of single-agent and then move to that of multi-agent.

1.1. Literature on reinforcement learning based route choices

The existing literature on dynamic routing games using RL-based approaches is listed in Table 1. Based on how to define the action of an agent, we broadly categorize RL-based route choice models into two groups. In the first group, the action of an agent is a route (Zhou et al., 2020; Ramos et al., 2018; Stefanello et al., 2016). For example, in Ramos et al. (2018) and Stefanello et al. (2016), for an agent aiming to go to her destination node from her origin node, the action space for her is the k shortest paths from her origin to her destination. These studies assume that every driver does follow her chosen route until she reaches her destination. In reality, however, drivers could deviate from their chosen route when they realize that traffic condition on some alternative routes might be better, especially when they get stuck in traffic on the current route. To capture this en-route adaptive behavior of drivers, the second group of studies focus on en-route traffic assignment, or route choice from the perspective of a driver. In these studies, the action space of an agent currently at a node is the outbound links from the node (Grunitzki et al., 2014; Bazzan and Grunitzki, 2016; Mao and Shen, 2018). In other words, every agent needs to decide which outbound link to choose whenever the agent arrives at a node, until the agent reaches some terminal node. This en-route decision-making process captures the adaptive behavior of real drivers.

To study the en-route decision-making process of drivers, both single-agent RL and multi-agent RL are used in the literature. A single-agent Q-learning approach is developed in Mao and Shen (2018) with the use of some global information such as congestion status in the definition of state. The authors bootstrapped two traffic profiles from the PeMS dataset and derived an optimal policy for the agent. Unfortunately, single-agent RL fails to capture the competition among adaptive agents. Multi-agent RL is recently used to tackle the multi-driver route choice task (Grunitzki et al., 2014; Bazzan and Grunitzki, 2016). Despite of modeling multi-driver interactions, these studies use independent multi-agent tabular Q-learning where every agent is treated as an independent learner who has no information of other agents. The assumption of independent learners suffers from two issues. First, it is impractical to assume that agents select routes independently without accounting for interactions with other agents and congestion effects on transportation networks. Second, theoretical convergence guarantee for Q-learning does not hold, because the environment is no longer Markovian and stationary (Matignon et al., 2012; Nguyen et al., 2020).

To fill the aforementioned research gaps, in this paper, we will model en-routing decision-making processes in a multi-agent system as a Markov game in which one's payoff function depends on all others' routing strategies.

1.2. MARL and multi-agent Markov game

As a game-theoretic framework for MARL, Markov game (Littman, 1994; Solan and Vieille, 2015) is a generalization of MDP-like environments to multiple interacting agents with competing goals, in which the environment makes transitions probabilistically in response to the agents' actions. Markov game is typically defined as a non-cooperative game where self-interested agents aiming to maximize their own payoffs. In a Markov game, the solution concept of Nash equilibrium is generally adopted (Hu and Wellman, 1998). At a Nash equilibrium, one agent's strategy (i.e., policy) is the best response to other agents' strategy. MARL is an efficient and versatile tool to solve for the optimal policy of each agent. Among the pioneering studies, Littman (1994) proposes a minimax Q-learning algorithm to solve for the optimal policies of two agents with opposed goals in a zero-sum game, and Hu and Wellman (1998) extends it to a general-sum game and provided a multi-agent Q-learning algorithm.

Recent literature has witnessed various applications of MARL to high-dimensional and complicated tasks such as playing the game of Go (Silver et al., 2016, 2017), Poker (Brown and Sandholm, 2018, 2019), Dota 2 (OpenAI, 2018), and StarCraft II (Vinyals et al., 2019). Outside the computer game domain, MARL has also attracted significant attention and been used in energy sharing (Prasad and Dusparic, 2019), federated control (Kumar et al., 2017), and sequential social dilemma (Leibo et al., 2017), to name a few. Interested readers are referred to Nguyen et al. (2020) for a comprehensive review of applications and challenges of MARL. In the transportation domain, we have also seen a growing trend of applying MARL to vehicle routing (Bazzan and Klügl, 2008; Bazzan and Grunitzki, 2016), traffic signal control (Bakker et al., 2010; Chen et al., 2020), fleet management (Lin et al., 2018; Shou et al., 2020; Shou and Di, 2020; Di et al., 2018; Di and Ban, 2019; Chen and Di, 2021), order dispatching (Li et al., 2019), and autonomous driving (Palanisamy, 2019; Bhalla et al., 2020). However, its application to dynamic routing game is still at its nascent stage.

1.3. Contributions of this paper

Here we will highlight the main contributions of this study as follows.

1. Multi-agent dynamic routing is modeled as a Markov routing game (MRG) that accounts for the interactions among adaptive agents and traffic congestion. It is capable of modeling individual agents' routing behavior with a partial or unknown information structure and a model-free environment, in which agents need to learn optimal policies through repeated interactions with other agents and the traffic environment.
2. An efficient multi-agent reinforcement learning algorithm is developed via a mean action, which is defined as the traffic flow on the chosen link right after an agent enters the link. The use of the mean action not only captures the competition among agents, but also enables value sharing and policy sharing, which is computationally favorable, especially in a large-sized problem with a large number of agents.

Table 1
Existing research on dynamic routing using RL based approaches.

Research type	Reference	Setting	State	Action set	Reward	Algorithm	Bilevel optimization	Gap
Route-based	Stefanello et al. (2016)	Multi-agent	–	k shortest routes from origin to destination	Negative travel time	Independent tabular Q-learning	No	–
	Ramos et al. (2018)	Multi-agent	–	k shortest routes from origin to destination	App-based regret (anticipated disutility)	Independent tabular Q-learning	No	–
	Zhou et al. (2020)	Multi-agent	–	Feasible routes from origin to destination	Negative travel time	Bush-Mosteller RL scheme	No	–
En-route	Grunitzki et al. (2014)	Multi-agent	(node)	Outbound links	Negative travel time (Individual and systematic)	Independent tabular Q-learning	No	Independent learning suffers from non-stationary and non-Markovian issues; tabular Q-learning could not handle continuous or large state/action space
	Bazzan and Grunitzki (2016)	Multi-agent	(node)	Outbound links	Negative travel time	Independent tabular Q-learning	No	
	Mao and Shen (2018)	Single-agent	(time, node, congestion status)	Successor nodes	Negative travel time	Tabular Q-learning	No	Using global information, i.e., congestion status in both training and execution
	This study	Multi-agent	(time, node)	Outbound links	Negative travel cost	Mean-field deep Q-learning	Yes	

3. A variety of dynamic network loading (DNL) models are used as the simulation environment of the proposed Markov routing game. We then demonstrate the consistency of our computed Markov equilibrium and its associated dynamic user equilibrium (DUE). The linkage between the classical dynamic traffic assignment (DTA) paradigm and our proposed game is also demonstrated. We illustrate that the developed Markov game for dynamic routing depicts analytical DTA assuming complete information of environment transition dynamics and travel or delay time functional forms.
4. Computational efficiency of the developed Markov routing game is demonstrated on mid- and large-sized networks.

The remainder of the paper is organized as follows. Section 2 introduces the developed Markov game for dynamic routing and devises a mean field multi-agent deep Q-learning algorithm. Section 3 demonstrates the linkage between the classical DUE paradigm and our proposed MARL paradigm. Section 4 presents four examples to demonstrate the consistency between DUE and the equilibrium learned from our model as well as algorithm efficiency, on a simple network without and with spillback, the OW network, and a real-world network near Columbia University's campus in Manhattan of New York City. Section 5 concludes this study.

2. From single- to multi-agent routing models

In this section, we first state formally the problem of dynamic en-route routing in the multi-agent setting. Then we explain how one's sequential decision-making process can be formulated as a single-agent RL. Building on this, we generalize it to multi-agent RL and then propose an efficient algorithm.

Definition 2.1 (*MRG Problem Statement*)**.** There are N intelligent agents (autonomous vehicles (AV) for example) indexed by $i \in \{1, 2, \ldots, N\}$ moving from a set of origins, denoted as $\mathcal{O}$, to a set of destinations $\mathcal{D}$. Each agent aims to select its optimal next-go-to edge at every intermediate node by minimizing a travel cost functional over a predefined planning horizon $[0, T]$. The traffic system evolves according to all agents' actions and some transition probability at each time step. With the system state updated, those agents at intermediate nodes select the next-go-to edge. The process continues till either all agents reach the destination or the planning time horizon is terminated.

When one solves its own optimal control problem while others are doing so simultaneously, a congestion (routing) game forms. We assume that the system evolution dynamic remains unknown to these agents. Thus they have to learn the game and select their individual routing strategy online via their interaction with other agents and the traffic environment. The outcome is a Nash equilibrium that represents a scalable decentralized online routing strategy for each controllable agent.

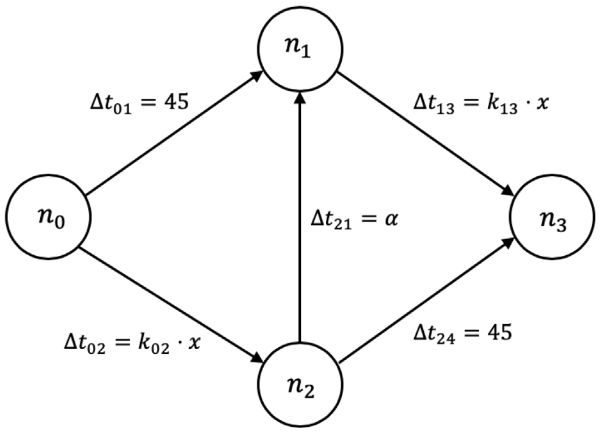


Fig. 1. Braess network.

Remark.

1. In this paper, we assume travelers are perfectly rational agents in the context of selfish routing. In other words, every traveler is a self-interest agent who aims to maximize individual accumulative rewards or minimize individual cost or maximize payoffs, including but not limited to autonomous vehicles, connected and automated vehicles, or human-driven vehicles with intelligent navigation systems and human drivers follow the navigation instructions completely. The device or mechanism that executes routing algorithms can be smart computers residing in autonomous vehicles or connected and automated vehicles, intelligent navigation aids or phone apps, and even human drivers who are perfectly rational in dynamic routing.
2. We use agents, players, travelers, and controllable vehicles interchangeably, which all represent those intelligent vehicles that select routes based on utility-optimizing behavior.

2.1. Single-agent reinforcement learning

As a stepping stone, we first introduce a single-agent RL approach. Within the single-agent scope, there is only one agent interacting with the stochastic environment. The goal of the deep Q-learning approach is to derive an optimal policy so that the agent could get the maximum expected cumulative reward by following the policy in the environment.

To be specific to the route choice task, we introduce the single-agent deep Q learning approach on a Braess network, as presented in Fig. 1. In the Braess network, there are four nodes, namely $\{n_0, n_1, n_2, n_3\}$. n_0 is the origin node and n_3 the destination/terminal node. There are five directed links connecting these nodes. We denote the link connecting n_i and n_j as l_{ij}. Link travel time on l_{ij} is denoted by Δt_{ij}. In this study, $\Delta t_{01} = 45$, $\Delta_{23} = 45$, $\Delta t_{02} = k_{02} \cdot x$, $\Delta t_{13} = k_{13} \cdot x$, and $\Delta t_{21} = \alpha$, where x denotes the travel flow (i.e., number of vehicles) on the link, k_{02} and k_{13} are two multipliers, and α is a control parameter.

First and foremost, the interaction between the agent and the environment is typically characterized by a Markov decision process or an MDP for short (Puterman, 1994). An MDP is specified by a tuple (S, A, R, P, γ), where S denotes the state space, A the action space, R the reward function, P the state transition probability matrix, and $\gamma \in [0, 1]$ the discount factor. An MDP goes as follows. At any state $s \in S$ but some terminal state, the agent chooses action $a \in A$ and executes the action in the environment. It observes a state transition from s to a new state $s' \in S$ with probability $P(s'|s, a)$ and receives a reward $r(s, a, s') \in R$. In the context of route choice,

- $s \in S$. State s consists of two components, namely node and time, i.e., $s = (n, t)$.
- $a \in A$. Action a for the agent currently in state $s = (n, t)$ is simply one of the outbound links from node n. For example, $a = l_{n,n'}$, where $l_{n,n'}$ is the link connecting node n and node n'. In this study, we assume action is deterministic, meaning that the agent will enter the chosen link. Note that the route choice for each agent is a discrete action space and the number of outbound links from a node varies from node to node in the network, indicating that the number of allowable actions is dependent on the node where the agent is currently located.
- P. After taking action a in state s, the agent arrives at a new state $s' = (n', t')$ with probability $P(s'|s, a)$, where n' is the end node of the chosen link (i.e., $l_{n,n'}$), and t' is the time right after the state transition. P is typically unknown to the agent. Therefore, the agent needs to repeatedly interact with the environment to gain state transition experiences, i.e., (s, a, s').
- $r \in R$. In addition to the observed state transition $s \to s'$, the agent receives a reward $r_{t'}(s, a, s')$ after executing action a, where the subscript t' explicitly denotes that the reward is received by the agent at time t'. In the context of route choice, the reward function $r_{t'}(s, a, s')$ is typically chosen as some negative travel cost related to the state transition $s \to s'$. For example, $r_{t'}(s, a, s') = -(t' - t)$, i.e., negative travel time, and $r_{t'}(s, a, s') = -\text{dist}(n, n')$, where $\text{dist}(n, n')$ measures the distance between node n and n', i.e., negative travel distance.

- γ. The discount factor γ is used to discount the future reward. When $\gamma = 1$, the agent does not differentiate future rewards from immediate rewards. As γ get smaller, the agent cares less about rewards received in the distant future, therefore her decision-making gets more myopic. In this study, we take $\gamma = 1$ because usually drivers aim to minimize their cumulative travel cost for a trip. In other words, they value a future travel cost and an immediate cost similarly.
- $\rho = \sum_{t=0}^{T} \gamma^t r_t$. ρ is the discounted cumulative reward. The agent aims to maximize ρ by deriving an optimal policy.

A *policy* μ is a mapping from state s to action a, i.e., $\mu(s) = a$. Following policy μ, value function $V^{\mu}(s)$ is defined as the expected discounted cumulative reward from state s. Mathematically, it is given in a recursive form, namely $V^{\mu}(s) = \mathbb{E}_{a\sim\mu(s),s'\sim P(\cdot|s,a)}[r(s,a,s')+\gamma V^{\mu}(s')]$.

An *optimal policy*, denoted as μ^*, is the policy that maximizes the value function, i.e., $\mu^* = argmax_{\mu} V^{\mu}(s),\ \forall s \in S$. The optimal value function, denoted as $V(s)$, is given as (Sutton and Barto, 1998)

$$V(s) = max_a \mathbb{E}_{s'\sim P(\cdot|s,a)}[r(s,a,s') + \gamma V(s')]. \tag{1}$$

While the value function $V(s)$ captures the optimal expected cumulative reward that an agent can earn from state s, the state–action value $Q(s,a)$ forces the agent to take action a and therefore captures the optimal expected cumulative reward from state s and action a. Mathematically, $V(s) = max_a Q(s,a)$. Substituting the relation between V and Q into Eq. (1) yields

$$Q(s,a) = \mathbb{E}_{s'\sim P(\cdot|s,a)}[r(s,a,s') + \gamma max_{a'} Q(s',a')]. \tag{2}$$

With the optimal action-value function Q, optimal policy can be explicitly derived as $\mu^*(s) = argmax_a Q(s,a)$.

With the collected experience tuple (s,a,s',r), the agent can update the Q value by the widely used tabular Q-learning algorithm

$$Q(s,a) \leftarrow Q(s,a) + \eta[r + \gamma max_{a'} Q(s',a') - Q(s,a)], \tag{3}$$

where $\eta \in (0,1]$ is the learning rate. Convergence is guaranteed for the Q-learning algorithm when the learning rate is decayed properly over time (Sutton and Barto, 1998). Thanks to the emergence of deep Q networks (DQNs) (Mnih et al., 2015), we could resort to neural networks as a functional approximator of the Q function. Denoting the functional approximator parameterized by θ as Q_θ, DQN updates its parameter θ by minimizing the following loss

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,s'}\left[\left(r(s,a,s') + \gamma max_{a'} Q_{\theta^-}(s',a') - Q_\theta(s,a)\right)^2\right], \tag{4}$$

where Q_{θ^-} is called a target network copied from Q_θ every τ episodes to ensure training stability, where τ is a hyperparameter.

Example 2.1 (*Single-agent Route Choice*)**.** To be more concrete, we demonstrate the aforementioned notations in the Braess network. Supposing an agent is initially placed at n_0 in the Braess network, the initial state of the agent is $s = (n_0, 0)$. There are two allowable actions for the agent, namely l_{01} and l_{02}. Assuming the agent chooses action l_{02} in the initial state, the agent arrives at $s' = (n_2, 1)$ with probability $P(s' = (n_2,1)|s = (n_0,0), a = l_{02}) = 100\%$. The time component in s' is 1 because the link travel time $\Delta t_{02} = x = 1$ (i.e., there is one agent on this link). The state transition probability is 100% because both the action and the state transition are deterministic in this case. Assuming the reward function is the negative travel time, the agent receives a reward of $-\Delta t_{02} = -1$ along with the state transition $s \rightarrow s'$.

We now apply the single-agent Q-learning to the Braess network with one agent initially at n_0. For the purpose of demonstration, we assume $\alpha = 1$ and $k_{02} = k_{13} = 40$ in this example. Due to the simplicity of this case, it could be solved analytically from the destination and moving backwards. Noticing the static nature of this example, we neglect the time component in state. n_3 is the terminal node, meaning that $V(n_3) = 0$. There are two nodes connecting n_3, namely n_1 and n_2. Note that the agent could arrive at n_1 from n_2, meaning that $V(n_2)$ is dependent on $V(n_1)$ and $V(n_1)$ is independent on $V(n_2)$, we thus discuss n_1 first. At node n_1, the only allowable action is l_{13} leading the agent to terminal node n_3, yielding $Q(n_1, l_{13}) = -\Delta t_{13} + \gamma V(n_3) = -40$ and $V(n_1) = -40$. At node n_2, there are two actions, namely l_{21} and l_{24}. Action l_{21} leads the agent to n_1, yielding $Q(n_2, l_{21}) = -\Delta t_{21} + \gamma V(n_1) = -(\alpha + 40) = -41$. Action l_{24} leads the agent to the terminal node n_3, yielding $Q(n_2, l_{24}) = -\Delta t_{24} + \gamma V(n_3) = -45$. Therefore, $V(n_2) = max(Q(n_2, l_{21}), Q(n_2, l_{24})) = -41$ and $\mu^*(n_2) = l_{21}$. At node n_0, there are two actions, namely l_{01} and l_{02}. Action l_{01} leads the agent to n_1, yielding $Q(n_0, l_{01}) = -\Delta t_{01} + \gamma V(n_1) = -85$. Similarly, $Q(n_0, l_{02}) = -\Delta t_{02} + \gamma V(n_2) = -81$. Therefore, $V(n_0) = max(Q(n_0, l_{01}), Q(n_0, l_{02})) = -81$ and $\mu^*(n_0) = l_{02}$. Following optimal policy μ^*, the agent takes route $n_0 \rightarrow n_2 \rightarrow n_1 \rightarrow n_3$, which is the shortest path in this example.

2.2. Multi-agent reinforcement learning and Markov games

Although the single-agent Q learning efficiently finds the shortest path, it fails to capture the competition among agents in a multi-agent system (MAS) (Di and Shi, 2021). In a single-agent RL problem, only one agent is placed in the environment to learn the optimal policy, which maximizes the expected cumulative reward of the agent. Unfortunately, when multiple agents follow this optimal policy, their expected cumulative reward might be low. For example, if there are 50 agents at node n_0 initially in the Braess network, their expected cumulative reward is -110 by following the optimal policy (i.e., the shortest path $n_0 \rightarrow n_2 \rightarrow n_1 \rightarrow n_3$). This reward (i.e., -110) is lower than that of following another route (e.g., $n_0 \rightarrow n_1 \rightarrow n_3$ yields an expected cumulative reward of -95). Therefore, to tackle a multi-driver route choice task, we develop a multi-agent deep Q-learning approach in this section to capture the competition among agents.

2.2.1. Problem formulation

Similar to the previous single-agent case, we introduce how to formulate the multi-agent routing problem on the Braess network shown in Fig. 1.

Due to the existence of multiple agents, the multi-agent routing problem is formulated as a Markov game (Littman, 1994), which is a generalization of Markov decision processes to multiple interacting agents with competing goals, in which the environment makes transitions probabilistically in response to the agents' actions. We denote the Markov game by a partially observable Markov decision process (Littman, 1994), defined by a tuple $(S, O_1, O_2, \dots, O_N, A_1, A_2, \dots, A_N, P, R_1, R_2, \dots, R_N, N, \gamma)$, where N is the number of agents and S is the environment state space. Environment state $\mathbf{s} \in S$ is not fully observable. Instead, agent i draws a private observation $o_i \in O_i$ which is correlated with $\mathbf{s}$. O_i is the observation space of agent i, yielding a joint observation space $O = O_1 \times O_2, \times \cdots \times O_N$, A_i is the action space of agent $i \in \{1, 2, \dots, N\}$, yielding a joint action space $A = A_1 \times A_2 \times \cdots \times A_N$, $P : S \times A \times S \to [0, 1]$ is the state transition probability, $R_i : S \times A \times S \to \mathbb{R}$ is the reward function for agent i, and γ is the discount factor. Agent $i \in \{1, 2, \dots, N\}$ uses a policy $\pi_i : O_i \times A_i \to [0, 1]$ to choose actions after drawing observation o_i. After all agents taking actions, the joint action $\mathbf{a}$ triggers a state transition $\mathbf{s} \to \mathbf{s}'$ based on the state transition probability $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$. Agent i draws a private observation o_i' corresponding to $\mathbf{s}'$ and receives a reward $r_i(\mathbf{s}, \mathbf{a}, \mathbf{s}')$. Agent i aims to maximize its discounted expected cumulative reward by deriving an optimal policy μ_i^*. This process repeats until agents reach their own terminal state.

Due to the existence of other agents, the Q-value function for agent i, i.e., Q_i, is now dependent on the environment state $\mathbf{s} \in S$ and the joint action $\mathbf{a} \in A$ of all agents, i.e,

$$Q_i = Q_i(\mathbf{s}, \mathbf{a}). \tag{5}$$

Similarly, the value function of agent i, i.e., $V_i = V_i(\mathbf{s})$, is dependent on the environment state $\mathbf{s}$.

In the multi-agent system, *optimal policy* of agent i, denoted as μ_i^*, is defined as the ith agent's best response to others' policies when others hold their policies constant. Below we introduce the formal definition of optimal policy in a multi-agent system using the concept of Nash equilibrium (Littman, 2001).

Definition 2.2.

1. A set of policies $\mu_i, \dots, \mu_N$ is a Nash equilibrium if each is a best response to the others. In other words, nobody can improve her value function by unilaterally switching her policy, while all others hold their policies fixed. That is, $\forall i \in \{1, \dots, N\}$, the value attained by agent i from any state s,

$$V_i(s) = max_{a_i} \mathbb{E}_{\mathbf{a}^{-i} \sim A} \mathbb{E}_{s' \sim P(\cdot|s, \mathbf{a})} [r_i(s, \mathbf{a}, s') + \gamma V_i(s')]. \tag{6}$$

   At a Nash equilibrium, each agent maximizes its value function given that all other agents remain fixed. Every Markov game has a Nash equilibrium in stationary policies (Filar and Vrieze, 1997).
2. *Markov strategy* is the policy that depends only on the current state. If all players other than i play Markov strategies, then player i has a best response that is a Markov strategy (Solan and Vieille, 2015). In this paper, we mainly focus on Markov strategy that is a mapping from the current state s to action $\mathbf{a}$.

Multi-agent RL is a computational tool for Markov games. We further specify each component using the language of MARL below.

- N. There are N controllable adaptive agents, denoted by $\{1, 2, \dots, N\}$. Note that in a traffic network, there are also other traffic participants who are not controlled by the learning algorithm developed in this study. We regard those uncontrollable traffic participants as background traffic. The background traffic affects the behavior of controllable agents by its impact on link travel cost. For example, if the background traffic has caused a traffic jam on a link, it is very likely that controllable agents will avoid this jammed link.
- $\mathbf{s} \in S$. Environmental state $\mathbf{s}$ consists of some global information such as distribution of agents and traffic condition on each link. $\mathbf{s}$ is not fully accessible to agents.
- $\mathbf{o} \in O$. Although the environmental state $\mathbf{s}$ is not observable by agent i, the agent is able to draw a private observation $o_i \in O_i$, which is correlated with $\mathbf{s}$. The Cartesian product of private observation spaces of all agents forms the joint observation space, i.e., $O = O_1 \times O_2 \times \cdots \times O_N$. In this paper, o_i consists of two components, namely node and time, i.e., $o_i = (n, t)$. Joint observation $\mathbf{o} = (o_1, o_2, \dots, o_N)$.
- $\mathbf{a} \in A$. Based on $o_i = (n, t)$, the allowable action set for agent $i \in \{1, 2, \dots, N\}$ consists of all outbound links from node n. For example, $a_i = l_{n,n'}$, where $l_{n,n'}$ is the link connecting n and n'. Joint action $\mathbf{a} = (a_1, a_2, \dots, a_N)$.
- P. Joint action $\mathbf{a}$ triggers a state transition $\mathbf{s} \to \mathbf{s}'$ with probability $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$. Agent $i \in \{1, 2, \dots, N\}$ draws a new private observations, namely $o_i' = (n', t')$, where n' is the end node of the chosen link $l_{n,n'}$ and t' is the time when agent i arrives at n'. The private observation o_i' is correlated with $\mathbf{s}'$.
- $r \in R$. In addition to o_i', agent $i \in \{1, 2, \dots, N\}$ receives a reward $r_{i,t'}(\mathbf{s}, \mathbf{a}, \mathbf{s}')$, where the subscript i, t' explicitly denotes that the reward is received by agent i at time t'. Similar to the single-agent RL, reward is typically some negative travel cost such as travel time and travel distance in the context of route choice.
- γ. Similar to the single-agent RL, $\gamma = 1$.
- $\rho_i = \sum_{t=0}^{T} \gamma^t r_{i,t}$. ρ_i is the discounted cumulative reward received by agent i. If the goal of agent i is to maximize ρ_i, agents are non-cooperative and selfish; if the goal of agent i is to maximize the average discounted cumulative rewards of all agents, i.e., $\frac{1}{N} \sum_{i=1}^{N} \rho_i$, agents are cooperative.

Due to the coexistence of other agents, Q function of agent i, i.e., Q_i is now a function of environmental state $\mathbf{s}$ and joint action $\mathbf{a}$, namely

$$Q_i = Q_i(\mathbf{s}, \mathbf{a}). \tag{7}$$

Similarly, value function of agent i, i.e., V_i is a function of environmental state $\mathbf{s}$, namely $V_i = V_i(\mathbf{s})$. Note that each agent i has her own Q function, i.e., Q_i, which may be different from Q functions of other agents, and therefore has an optimal policy μ_i^* different from policies of other agents. This distinguishes multi-agent RL from its single-agent counterpart, because agents may behave differently even in the same state with multi-agent RL while they choose the same action in the same state with single-agent RL.

Considering that environmental state $\mathbf{s}$ is not fully observable by agents, Q function shown in Eq. (7) is not tractable from the perspective of agent i. Actually, it is private observation o_i based on which agent i chooses next action. We thus rewrite Q function for agent i as

$$Q_i = Q_i(o_i, o_{-i}, a_i, a_{-i}), \tag{8}$$

where o_{-i} and a_{-i} denote the joint observation and joint action of all agents except agent i, respectively. In a non-coordinated environment (i.e., agents are not sharing their private observations or actions), o_i and a_i are but o_{-i} and a_{-i} are not accessible to agent i. Unfortunately, removing the dependency of Q_i on a_{-i} or o_{-i} may introduce large instability in Q-learning. In addition, convergence guarantee in single-agent Q-learning is no longer valid due to the adaptive behavior of other agents if o_{-i} and a_{-i} are not included (Matignon et al., 2012).

For a real-world multi-agent problem, there are commonly more than tens of thousands of agents. Maintaining one Q function, i.e., Q_i for each agent is thus computationally infeasible. Leveraging some homogeneity among agents (e.g., some agents share the same utility function), we broadly categorize agents into C groups and enable Q function sharing within each group. We denote the shared Q function for agents in group $c \in \{1, 2, \dots, C\}$ as

$$Q^c(o_i, o_{-i}, a_i, a_{-i}). \tag{9}$$

The dimension of the input to the shared Q function easily becomes prohibitively large when the number of agents in the environment increases. For example, the joint action space for 1000 agents is $1000 \times d$ dimensional if each action is a d dimensional vector. We take d as a generic representation of the dimension of the action space of each agent. For example, d could be 1 action of agents denoted as a scalar (e.g., 1,2, …, i.e., label encoding) or 3 when the agent has three actions to take and these actions are represented by one hot encodings. To tackle this issue, we first notice that joint observation o_{-i} and joint action a_{-i} record full information of other agents, which may contain redundant information from the perspective of agent i. Actually, in the task of route choice, observations and actions of agents who are currently far away from agent i have a very limited influence on agent i. It is nearby agents who exert an impact on the traffic condition around agent i. For example, after agent i chooses a link at a node, an immediate influential quantity is the traffic flow on the chosen link. If the flow is high, agent i may experience a high travel cost on the link; if the flow is low, agent i may have a smooth transition to the end of the link. Therefore, high-dimensional o_{-i} and a_{-i} in Eq. (9) could be approximated by some low-dimensional aggregate information of nearby agents. Similar to the mean field approximation in Yang et al. (2018), we call this aggregate information as the mean action of nearby agents and denote it by $\bar{a}_i$. Q function in Eq. (9) thus becomes

$$Q^c(o_i, o_{-i}, a_i, a_{-i}) \approx Q^c(o_i, a_i, \bar{a}_i). \tag{10}$$

We now formally define the mean action in the context of route choice.

Definition 2.3 (*Mean Action*). Mean action $\bar{a}_i$ is defined as the traffic flow on the link that is chosen by agent i. Note that the traffic flow is calculated right after agent i enters the link.

2.2.2. Mean field multi-agent deep Q-learning

Different from single-agent RL, agent i in group $c \in \{1, 2, \dots, C\}$ in an MAS interacts with not only the environment but also other agents and collects experience tuples in the form of $(o_i, a_i, o_i', r_i, \bar{a}_i)$. Considering that mean action $\bar{a}_i$ is typically in a large discrete or even continuous space, a DQN parameterized by θ_c, denoted by $Q^c(o_i, a_i, \bar{a}_i|\theta_c)$, is used to approximate $Q^c(o_i, a_i, \bar{a}_i)$. Similar to Eq. (4), DQN updates its parameter θ_c by minimizing the following loss

$$\mathcal{L}(\theta_c) = \mathbb{E}_{o_i,a_i,\bar{a}_i,o_i'} \left[(r_i + \gamma max_{a_i'} \mathbb{E}_{\bar{a}_i' \sim \mu_{-i}^*} [Q^c(o_i', a_i', \bar{a}_i'|\theta_c^-)] - Q^c(o_i, a_i, \bar{a}_i|\theta_c))^2 \right] \tag{11}$$

where $Q^c(\cdot|\theta_c^-)$ is a target network copied from $Q^c(\cdot|\theta_c)$ every τ episodes to stabilize training. After updating Q function, optimal policy

$$\mu_i^*(o_i) = argmax_{a_i} \mathbb{E}_{\bar{a}_i \sim \mu_{-i}^*} [Q^c(o_i, a_i, \bar{a}_i|\theta_c)], \ \forall o_i \in O_i. \tag{12}$$

Algorithm 1 Mean field multi-agent deep Q-learning (MF-MA-DQL)

```
1: Input: exploration parameter $\epsilon = \epsilon_0$, learning rate $\eta = \eta_0$, target network update period $\tau$
2: Initialize one DQN $Q^c(o, a, \bar{a}|\theta)$, parameterized by $\theta_c$, and one target network $Q^c(o, a, \bar{a}|\theta_c^-)$ for each group $c \in \{1, 2, \cdots, C\}$
3: Initialize $N$ dictionaries to store the optimal policy for agents, i.e., $\mu_i^*, \forall i \in \{1, 2, \cdots, N\}$
4: Initialize one experience replay buffer $B_c$ for each group $c \in \{1, 2, \cdots, C\}$
5: Set $episode = 0$
6: repeat
7:     From the initial environmental state $s_0$, each agent $i$ draws observation $o_i$
8:     Set $t = 0$
9:     repeat
10:        For each agent $i$, select action $a_i$ according to the $\epsilon$-greedy method, i.e., randomly select an allowable action with probability $\epsilon$ and greedily according to policy $\mu_i^*$ with probability $1 - \epsilon$
11:        Execute joint action $\mathbf{a} = (a_1, a_2, \cdots, a_N)$ in the environment to trigger state transition $\mathbf{s} \to \mathbf{s}'$
12:        Each agent $i$ draws new observation $o_i'$, receives reward $r_i$, and observes mean action $\bar{a}_i$
13:        Store experience tuple $(o_i, a_i, o_i', r_i, \bar{a}_i)$ into replay buffer $B_c$ if agent $i$ belongs to group $c$
14:        $t \leftarrow t + 1$
15:    until $t = T$
16:    for $c = 1$ to $C$ do
17:        for $j = 1$ to $J_c$ do
18:            Sample a batch of size $K_c$ from replay buffer $B_c$
19:            Update parameter $\theta_c$ of $Q^c$ by minimizing the loss defined in Eq. (11)
20:            Update optimal policy $\mu_i^*$ of agent $i$ by Eq. (12)
21:        end for
22:    end for
23:    $episode = episode + 1$
24:    Decrease the exploration parameter $\epsilon$
25:    Decrease the learning rate $\eta$
26:    if $episode$ mod $\tau = 0$ then
27:        for $c = 1$ to $C$ do
28:            Update parameter $\theta_c^-$ of the target network, i.e., $\theta_c^- \leftarrow \theta_c$
29:        end for
30:    end if
31: until the algorithm converges
32: Return optimal policies $\mu_i^*, \forall i \in \{1, 2, \cdots, N\}$
```

The mean field multi-agent deep Q-learning (MF-MA-DQL) algorithm is summarized in Algorithm 1. We first initialize a centralized deep Q network, parameterized by θ_c, and a target Q network, parameterized by θ_c^- for each group $c \in \{1, 2, \dots, C\}$. Although Q function is shared among agents in the same group, each agent i in group c has her own optimal policy μ_i^*, which is a dictionary mapping from observation o_i to action a_i. From initial environmental state s_0, each agent $i \in \{1, 2, \dots, N\}$ keeps drawing private observation o_i and taking action a_i according to the widely used ϵ-greedy method (i.e., agent i chooses action randomly with probability ϵ and greedily from optimal policy μ_i^* with probability $1 - \epsilon$), until reaching some terminal state. Joint action $\mathbf{a} = (a_1, a_2, \dots, a_N)$ is executed in the environment to trigger the state transition $\mathbf{s} \to \mathbf{s}'$. Each agent i draws new private observation o_i', receives reward r_i, and observes mean action $\bar{a}_i$. An experience replay buffer B_c for each group c is used to store experience tuple $(o_i, a_i, r_i, \bar{a}_i, o_i')$ generated by agent i in group c. Batch training is then used to update the centralized Q function by sampling a batch of size K_c from buffer B_c and minimizing the loss shown in Eq. (11) over this batch. With the updated centralized Q function, each agent updates her optimal policy by Eq. (12).

Remark. As a value based approach, the developed MF-MA-DQL algorithm does not suffer the problem of a variable action set. A variable action set means that the number of allowable actions might vary from state to state. For example, there are two outbound links from node n_0 and one outbound link from node n_1 in the Braess network shown in Fig. 1, causing the problem of variable action set. A policy based approach, e.g., the widely used actor–critic method, does not fit naturally well into a RL problem with variable action set. The main reason is that the policy neural network typically takes as input a state and outputs a probability distribution on all allowable actions (i.e., the action set). Consequently, a variable action set might result in some challenges to maintain a good structure of the policy network. Fortunately, value based approaches automatically handle the problem of variable action set, because a value based approach calculates the expected Q value of all possible actions at a state and selects the action with the highest Q value as the optimal policy.

Example 2.2 (*Multi-agent Route Choice*). Now we apply the MF-MA-DQL algorithm to a multi-agent route choice problem. In the Braess network shown in Fig. 1, now two agents are initially placed at node n_0. The goal of both agents is to travel to the destination node n_3 as soon as possible. Consequently, these two agents are considered to be homogeneous, because they share the same reward

Table 2
Optimal values of interest.

Observation o	Action a	Mean action $\bar{a}$	$Q(o, a, \bar{a})$	$V(o)$
n_1	l_{13}	1	-40	$V(n_1) = \mathbb{E}_{\bar{a}}[Q(n_1, l_{13}, \bar{a})]$ is within the range of $[-80, -40]$.
n_1	l_{13}	2	-80	
n_2	l_{21}	1 or 2	≤ -50	Action l_{23} strictly dominates l_{21}. $V(n_2) = Q(n_2, l_{23}, \bar{a}) = -45$.
n_2	l_{23}	1 or 2	-45	
n_0	l_{01}	1	-85	$V(n_0) = -85$ if agents choose actions differently at n_0 and $V(n_0) = -125$ if agents choose the same action.
n_0	l_{01}	2	-125	
n_0	l_{02}	1	-85	
n_0	l_{02}	2	-125	

function (i.e., the negative travel time) and go to the same destination. We first demonstrate the notations of Markov game in this example.

There are $N = 2$ agents in the Braess network. Environmental state $\mathbf{s}$ is not observable, Correlated with $\mathbf{s}$, the joint observation of agents $\mathbf{o} = ((n_0, 0), (n_0, 0))$. There are two allowable actions for both agents, namely l_{01} and l_{02}. The joint action $\mathbf{a} = (a_1, a_2)$. $\mathbf{a}$ triggers state transition $\mathbf{s} \to \mathbf{s}'$ with probability $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$. Each agent $i \in \{1, 2\}$ then draws new private observation o_i', receives reward r_i, and observes mean action $\bar{a}_i$. To be precise, we illustrate o_i', r_i, and $\bar{a}_i$ under two joint actions.

- With $\mathbf{a} = (l_{01}, l_{02})$, $o_1' = (n_1, \Delta t_{01} = 45)$ and $o_2' = (n_2, \Delta t_{02} = k_{02})$. $r_1 = -\Delta t_{01} = -45$ and $r_2 = -\Delta t_{02} = -k_{02}$. $\bar{a}_1 = 1$ and $\bar{a}_2 = 1$.
- With $\mathbf{a} = (l_{02}, l_{02})$, $o_1' = (n_2, \Delta t_{02} = 2 \cdot k_{02})$ and $o_2' = (n_2, \Delta t_{02} = 2 \cdot k_{02})$. $r_1 = -\Delta t_{02} = -2 \cdot k_{02}$ and $r_2 = -\Delta t_{02} = -2 \cdot k_{02}$. $\bar{a}_1 = \bar{a}_2 = 2$.

Note that although agent 2 takes action l_{02} in both cases, the agent arrives at different observations, i.e., (n_2, k_{02}) and $(n_2, 2 \cdot k_{02})$ due to the changing behavior of agent 1 (i.e., taking action l_{01} in the first case and l_{02} in the second).

Due to the simplicity of this case, we now analyze optimal value at each node in the network. For the purpose of demonstration, we assume $\alpha = 10$ and $k_{02} = k_{13} = 40$ in this example. Similar to Example 2.1, we neglect the time component in private observation due to the static nature of this problem.

1. Noticing that node n_3 is the terminal node, $V(n_3) = 0$.
2. At node n_1, the only action that agents can take is l_{13}. After one or two agents, depending on whether both of them reach node n_1, taking action l_{13}, there are two scenarios: (1) there is only one agent (i.e., either agent 1 or agent 2) on link l_{13}, meaning that traffic flow on l_{13} is 1 and it takes the agent $\Delta t_{13} = k_{13} \cdot 1 = 40$ to reach n_3. Therefore, $Q(o = n_1, a = l_{13}, \bar{a} = 1) = -40$. Note that here we drop the subscript index i for the agent for notation simplicity, because agents are homogeneous in this case and $Q(o_1 = n_1, a_1 = l_{13}, \bar{a}_1 = 1) = Q(o_2 = n_1, a_2 = l_{13}, \bar{a}_2 = 1) = -40$. In other words, no matter who the agent is, it takes the agent 40 time steps to reach n_3 starting from n_1, taking action l_{13}, and observing $\bar{a} = 1$; (2) there are two agents on link l_{13}, meaning that traffic flow on l_{13} is 2 and it takes the agent $\Delta t_{13} = k_{13} \cdot 2 = 80$ to reach n_3. Therefore, $Q(o = n_1, a = l_{13}, \bar{a} = 2) = -80$. Note that in scenario (2), it could be either both agents choose action l_{13} at node n_1 simultaneously or one agent chooses action l_{13} first because she arrives at n_1 first and the other agent chooses action l_{13} later due to her later arrival at node n_1 (e.g., one agent chooses route $n_0 \to n_1$ and arrives at n_1 at $t = 45$, and the other agent chooses route $n_0 \to n_2 \to n_1$ and arrives at n_1 at $t = k_{02} + \alpha = 50$). In the former case, both agents observe $\bar{a} = 2$ and spend time 80 on their way to n_3; in the latter case, the agent who arrives at n_1 first observes $\bar{a} = 1$ and spends time 40 on her way to n_3 and the other agent observes $\bar{a} = 2$ and spends time 80. With $Q(n_1, l_{13}, 1) = -40$ and $Q(n_1, l_{13}, 2) = -80$, optimal value at n_1, i.e., $V(n_1)$, is within the range of $[-40, -80]$ and depends on the distribution of $\bar{a}$.
3. At node n_2, there are two outbound links, namely l_{21} and l_{23}. If one or two agents at n_2 choose action l_{21}, $Q(n_2, l_{21}, \bar{a}) = -\alpha + \gamma \times V(n_1) = -10 + V(n_1)$, which is lower than -50, regardless of $\bar{a}$. If one or two agents at n_2 choose action l_{23}, $Q(n_2, l_{23}, \bar{a}) = -\Delta t_{23} = -45$. Therefore, action l_{21} is strictly dominated by action l_{23}, indicating that optimal policy at n_2 for both agents is to choose action l_{23} and $V(n_2) = -45$.
4. Finally, at node n_0, there are two possible actions, namely l_{01} and l_{02}, and both agents choose action simultaneously. There are 3 possible scenarios: (1) when both agents choose action l_{01}, $Q(n_0, l_{01}, \bar{a} = 2) = -\Delta t_{01} + \gamma V(n_1) = -45 + (-80) = -125$. Note that $V(n_1) = -80$ because both agents arrive at node n_1 and choose action l_13 at the same time. (2) when both agents choose action l_{02}, $Q(n_0, l_{02}, \bar{a} = 2) = -\Delta t_{02} + \gamma V(n_2) = -80 + (-45) = -125$. (3) when one agent chooses action l_{01} and the other chooses action l_{02}, $Q(n_0, l_{01}, \bar{a} = 1) = -\Delta t_{01} + \gamma V(n_1) = -45 + (-40) = -85$ and $Q(n_0, l_{02}, \bar{a} = 1) = -\Delta t_{02} + \gamma V(n_2) = -40 + (-45) = -85$. Note that $V(n_1) = -40$ because only one agent uses link l_{13}. Therefore, the optimal policy of one agent is to choose action l_{01} and that of the other agent is to choose action l_{02} at n_0.

In summary, optimal values of interest are listed in Table 2. Link l_{21} is not used by agents because it is strictly dominated by l_{23}. From n_0, the payoff for agents are $(-85, -85)$ if they choose actions differently and $(-125, -125)$ if they choose the same action (i.e., either l_{01} or l_{02}). Therefore, optimal policy of one agent is to choose action l_{01} and that of the other is to choose l_{02}, and this is a Nash equilibrium because no agent can get better payoff by a unilateral deviation.

Markov games to model en-route choice are independent of data-generating mechanism. Here data refers to background traffic in which vehicles move and by which travel choices are influenced, including but not limited to travel time or delay and/or travel

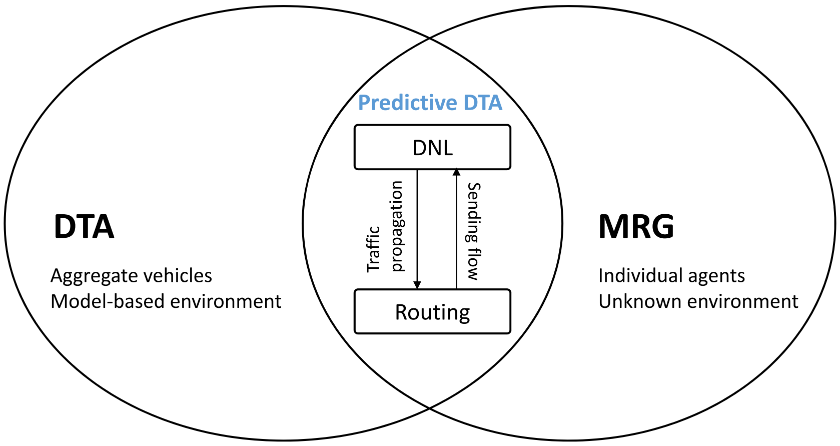


Fig. 2. Connection between DTA and Markov routing game (MRG).

speed on a link, queuing delay at a node, as well as transition flows between links. Routing flows can arise from models like DNL, or model-free simulation. In the next two sections, we will demonstrate the proposed routing game using data first generated from existing DNL models and then from a traffic simulator where its data generating mechanism remains unknown.

3. Linkage between dynamic traffic assignment (DTA) and Markov routing games (MRG)

DTA is the descriptive modeling of dynamic traffic flows on networks that is consistent with traffic flow theory, travel behaviors, and established travel demands. It has been widely used to model route choice behavior of travelers. A general DTA problem typically consists of two parts, namely a dynamic network loading (DNL) module and a route choice model.

DNL. Given traffic demand and route choices, the DNL module propagates traffic flow dynamics and/or traffic congestion through the network. There are two components in the DNL module, namely a link model and a junction model. A link model is used to propagate inflow traffic demand to exit and is usually described by either a delay/latency function or an exit-flow function (Nie and Zhang, 2005). As for the former, both linear delay functions and nonlinear functions (e.g., the widely used BPR function) are used in the literature. As for the latter, there are a large body of literature proposing or using various exit-flow functions such as the M-N model (Merchant and Nemhauser, 1978a,b), the point-queue model (Kuwahara and Akamatsu, 1997; Ban et al., 2012), and the cell transmission model (Daganzo, 1994, 1995). A junction model is used to determine the flow distribution at an intersection.

Route choice. Route choice models guide travelers to properly choose next go-to links right before they arrive at a node/intersection. According to the objective of travelers, there are typically two types of research problems, namely dynamic system optimum (DSO) and dynamic user equilibrium (DUE). While DSO aims to minimize total or average travel cost of all travelers (Ziliaskopoulos, 2000), DUE describes the equilibrium state in which no individual traveler could decrease her travel cost by unilaterally deviating from her current route choices (Friesz and Han, 2019; Friesz et al., 2013). (Huang et al., 2021) applies mean-field game to model both velocity and route choices of a large number of intelligent agents on a road network.

We now discuss the difference and connection between DTA and the MRG developed in this paper. Deriving a Nash equilibrium in general requires a perfect knowledge of the system evolution dynamics and the rewards of the game (Pérolat et al., 2017). However, the proposed MRG models individual agents' routing behavior with a partial or unknown information structure and a stochastic model-free environment. In other words, in MRG, agents lack the knowledge of the transition probability of system dynamics or travel time functions of road segments. Thus, agents need to learn through repeated interactions with other agents and the traffic environment. This setting mimics the reality that we usually have no knowledge of how traffic evolves nor a mathematical relation between travel time and congestion. This is the key difference between DTA models and MRG presented in this paper. DTA models stipulate the system evolution dynamics and the travel cost of using each link or route, while MRG does not contain such information and thus requires agents to learn and adapt their strategy according to historical episodes generated from interactions.

Other than the difference between DTA and MRG, we would like to draw their linkage. Markov game is the generalization of MDP-like environment to a multi-agent system. The environment of an agent is comprised of other agents and unknown objects. In other words, stochasticity of the environment arises from the actions of other agents and unknown sources. To generate such an MDP-like environment, we not only need to model all the agents' Markov strategies, but also their interactions with one another. Various DNL models of DTA can provide such an environment. Specifically, there are two components in a DTA, namely a DNL module and a route choice model, and two components in a Markov game, namely an environment and agents. The connection between DTA and Markov game is componentwise, as schematically presented in Fig. 2. The DNL in DTA can be used as the environment engine in Markov game, while the route choices in DTA are made by agents in Markov game. Further, the interaction between the DNL and route choice in DTA is similar to that between the environment and agents in Markov game. In DTA, a route choice model outputs route choices of travelers whenever they arrive at a node or an intersection, and the DNL module propagates

Table 3
DTA versus MRG.

Model	Game	Traffic dynamics				Travel Cost		Equilibrium
	Atomic or non-atomic	Knowledge of dynamics	State transition	Functional form	Temporal resolution	Knowledge of reward	Form	
DTA	Non-atomic	Perfect	Stochastic or deterministic	Model-based	Discrete or continuous	Known	Actual or instantaneous	Predictive or reactive
MRG	Atomic	Imperfect	Stochastic	Model-free	Discrete	Unknown	Actual	Predictive

Atomic–Discrete vehicles; Non-atomic–Aggregate traffic flow.

traffic flow dynamics. In MRG, agents (i.e., travelers) take actions (i.e., route choices) whenever they arrive at a node, and the environment experiences a state transition determined by the DNL module in DTA. To be precise, the actions taken by agents in Markov game are route choices in DTA, and the state transition and rewards in Markov game are determined by the DNL module in DTA. Moreover, DUE and MRG have non-overlapped regimes that delineate the differences of these two model types: DTA models encompass those with reactive DUE (Li et al., 2000; Lam and Huang, 1995) and are primarily solved using aggregate traffic flow variables; while MRG is focused on en-route dynamic choice of individuals within traffic environments of which transition dynamics and travel time functions remain unknown to agents.

In summary, the MRG coupled with DNL models depicts one type of DTA models, of which the decision-making processes of discrete vehicles (in other words, atomic agents (Roughgarden, 2007)) instead of aggregate traffic flows are modeled, traffic dynamic evolution is modeled as a stochastic transition probability without any need of model specification and in discrete time steps, and the equilibrium outcome is predictive DUE. The detailed comparisons of DTA and MRG are summarized in Table 3.

We further specify two components, namely, dynamic network loading and route choice, in DTA and MRG, respectively, using mathematical representations.

DTA: find optimal path flow $h_k^{t*}, k \in \mathcal{K}, \forall t \in \{0, T\}$.

- Dynamic network Loading: $\begin{cases} N_l^{\uparrow}(t+1) = N_l^{\uparrow}(t) + \sum_{l' \in \Gamma^{-1}(l)} y_{l'l}(t), \\ N_l^{\downarrow}(t+1) = N_l^{\downarrow}(t) + \sum_{l'' \in \Gamma(l)} y_{ll''}(t), \end{cases} \quad l \in \mathcal{L}^D, 0 \le t \le N_t - 1,$
  where $N_l^{\uparrow}$ and $N_l^{\downarrow}$ are the cumulative counts of vehicles at the upstream and downstream of link l, respectively. $\sum_{l' \in \Gamma^{-1}(l)} y_{l'l}$ and $\sum_{l'' \in \Gamma(l)} y_{ll''}$ are the inflow and outflow of link l, respectively.
- Route choice for aggregate vehicle flows:
  $\forall k \in \mathcal{K}, \begin{cases} h_k^t > 0 \rightarrow \pi_k^t = min_k \mathcal{T}(H), \\ h_k^t = 0 \rightarrow \pi_k^t > min_k \mathcal{T}(H), \end{cases}$
  where, h_k^t is the traffic flow on path $k \in \mathcal{K}$, H is the path flow matrix, $\mathcal{T}(H)$ is the travel time matrix regarding each path and π_k^t is the minimum travel time.

Remark.

1. $y_{l'l}$ is the transit flow from link l' to link l, depending on sending and receiving flow in DNL models, which will be introduced in Section 4.
2. Each element τ_k^t in $\mathcal{T}(H)$ is the travel time on path k and $\tau_k^t = T_k^{\downarrow}(N_k^{\uparrow}) - T_k^{\uparrow}(N_k^{\uparrow})$. $T_k^{\downarrow}$ and $T_k^{\uparrow}$ represent the time when vehicles leave and enter path k, respectively.

MRG: find optimal policies $\mu_i^*, i = 1, \cdots, N, \forall t \in \{0, T\}$.

- Environment transition: $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$, where $\mathbf{s}$ represents the environment state, i.e., the agent distribution on the network, which determines the cumulative counts of vehicles $N_l^{\uparrow}, N_l^{\downarrow}, l \in \mathcal{L}^D$.
- Route choice for each agent:
  $\forall i \in N, \begin{cases} a_{i,l}^t > 0 \rightarrow V_i(s) = max_{a_i} \mathbb{E}_{\mathbf{a}^{-i} \sim A} \mathbb{E}_{s' \sim P(\cdot|s,\mathbf{a})}[r_i(s, \mathbf{a}, s') + \gamma V_i(s')], \\ a_{i,l}^t = 0 \rightarrow V_i(s) < max_{a_i} \mathbb{E}_{\mathbf{a}^{-i} \sim A} \mathbb{E}_{s' \sim P(\cdot|s,\mathbf{a})}[r_i(s, \mathbf{a}, s') + \gamma V_i(s')]. \end{cases}$
  where, for agent i, $a_{i,l}^t$ is her route choice, indicating whether agent i chooses link l or not. The reward $r_i(s, \mathbf{a}, s')$ is the negative realized travel time after agent i choosing action a_i.

4. Numerical examples

In this section, we demonstrate the effectiveness of our MF-MA-DQL algorithm to solve optimal routing policies. In Example 4.1, we apply the developed algorithm to a simple network based on a variety of DNL models and compare it with a traditional iterative

Table 4
Numerical examples.

	Example	Network	Dynamic loading environment
Model-based	4.1	Simple	PQ, SQ, CTM, LTM
	4.2	Simple (spillback)	LTM
	4.3	OW	LTM
Model-free	4.4	Real-world network	SUMO

method. In Example 4.2, we specify one of the DNL models - LTM to explore the spillback phenomenon. In Example 4.3, we apply the developed algorithm to the OW Network (Ortuzar and Willumsen, 2011) with LTM as its data-generation environment. In Example 4.4, our algorithm is applied to a real-world network with traffic simulated in SUMO (i.e., model-free environment). To reiterate, the first three examples contain model-based traffic environments using DNL models, while the last one has a model-free environment in which travel time function and traffic state update are unknown and thus requires agents to learn from their experiences (see Table 4).

Example 4.1. In this example, we apply the developed MF-MA-DQL algorithm to a simple network. We first briefly introduce several DNL models used as the environment of MARL.

1. **PQ:** In the point queue (PQ) model, vehicles have no physical length (no spillback) and move at free flow speed. The link traversal time consists of free flow travel time and queuing delay (Zhang et al., 2013).
2. **M-N:** In the M-N model, exit flow is a generalized function of road density (Merchant and Nemhauser, 1978a). When the function satisfies flow capacity constraints, the M-N model becomes a special case of PQ (Nie, 2011). For simplicity, we use the PQ model to demonstrate our algorithm.
3. **SQ:** The different between spatial queue (SQ) and PQ models is that the exit flow in the SQ model depends on the flow capacity and the remaining capacity of downstream links (Zhang et al., 2013).
4. **CTM:** The cell transmission model (CTM) approximately solves the LWR system by discretizing space and time (Daganzo, 1994, 1995). CTM considers not only flow capacity and jam density, but also backward speed into the fundamental diagram.
5. **LTM:** The link transmission (LTM) model exactly solves the LWR system using the Newell–Daganzo method (Yperman, 2007). Compared with CTM, LTM reduces computational complexity.
6. **DQ:** The deterministic double queue (DQ) model coincides with LTM (Osorio et al., 2011). For simplicity, we use LTM in this paper.

We then show how to construct the environment of MARL based on DNL models. Note that a link can be discretized into cells such that the length of each cell is exactly the distance traveled by a vehicle within a time interval Δt at free flow speed. We assume that the time interval Δt is also the time step in the environment of our MARL. To better understand this, we use LTM to illustrate the environment. Environments based on other DNL models are in Appendix A.1.

LTM Environment: LTM is applied to only to the upstream and downstream ends of a link (Yperman, 2007; Osorio et al., 2011). It is a discrete model, calculating the upstream and downstream flow at each time step. At time t, the cumulative counts of vehicles at the upstream and downstream ends of the link are denoted by $N^{\uparrow}(t)$ and $N^{\downarrow}(t)$, respectively. The sending flow $S(t)$ at the downstream end of the link is calculated based on the free flow speed v, representing how many vehicles can leave during the time interval $(t,\ t+\Delta t)$, which is calculated as:

$$S(t) = min\{N^{\uparrow}(t - \frac{L}{v} + \Delta t) - N^{\downarrow}(t),\ q_{max}\Delta t\},$$

where L represents the link length and $N^{\uparrow}(t-\frac{L}{v}+\Delta t)$ represents the upstream cumulative count of vehicles at time $t-\frac{L}{v}+\Delta t$. q_{max} represents the flow rate capacity and $q_{max}\Delta t$ is the maximum number of vehicles that can leave at each time step. The receiving flow $R(t)$ at the upstream end of the link is calculated based on the congested wave speed (backward speed) w, representing how many vehicles enter the link during the time interval $(t,\ t+\Delta t)$, which is calculated as:

$$R(t) = min\{N^{\downarrow}(t - \frac{L}{w} + \Delta t) + k_j L - N^{\uparrow}(t),\ q_{max}\Delta t\},$$

where $N^{\downarrow}(t-\frac{L}{w}+\Delta t)$ is the downstream cumulative count of vehicles at time $t-\frac{L}{w}+\Delta t$, k_j is the jam density and k_jL is the link capacity, representing the maximum number of vehicles that can be accommodated into the link.

Note that the receiving and sending flows only capture the upstream and downstream ends of a link. To determine the flow propagation in the environment, we also need node models to calculate how many vehicles can move from an upstream link to a downstream link. There are some simple node models capturing merging and diverging cases (Daganzo, 1994). However, general intersection with more than one inflow link and one outflow link is not standardized as merge and diverge models. It can be signal-controlled or priority-controlled. In this paper, we use discharging priorities (Ma et al., 2017) based on agents' route choice at a general intersection node where the inflow links of the node are ranked in some order, representing different priorities. Vehicles on each inflow link may leave based on the link priority and enter outflow links based on their route choice (actions).

With the aforementioned dynamic loading environment, now we apply the developed MF-MA-DQL algorithm to solve DUE where travel cost on all used routes for any given origin–destination pair is supposed to be the same according to Wardrop's first principle.

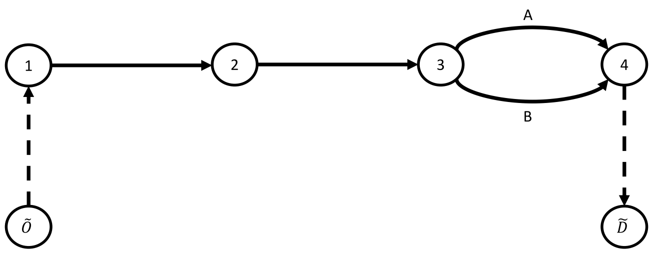


Fig. 3. Simple network.

Table 5
Parameter in Example 4.1.

Link	PQ		SQ	LTM/CTM
	L	q_{max}	k_j	w
$\tilde{O} \to 1$	∞	∞	–	–
$1 \to 2$	0.4	20	300	0.1
$2 \to 3$	0.4	20	300	0.1
$3 \to A \to 4$	0.8	2	60	0.1
$3 \to B \to 4$	0.4	2	30	0.1
$4 \to \tilde{D}$	∞	∞	–	–

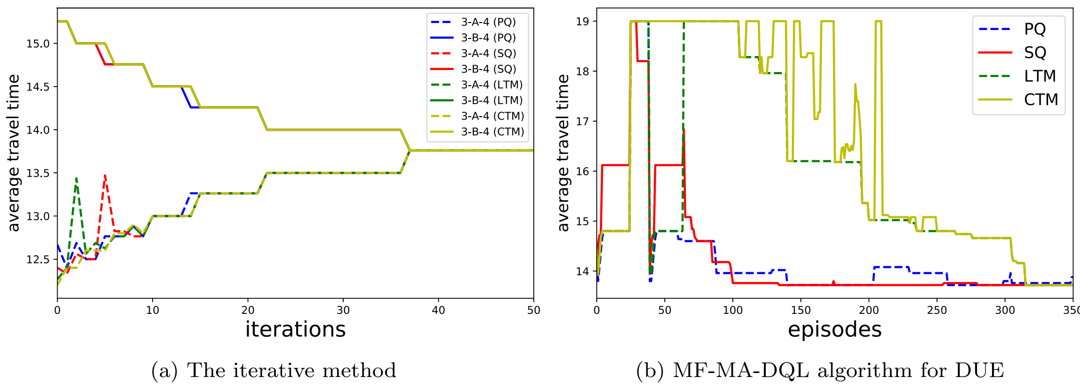


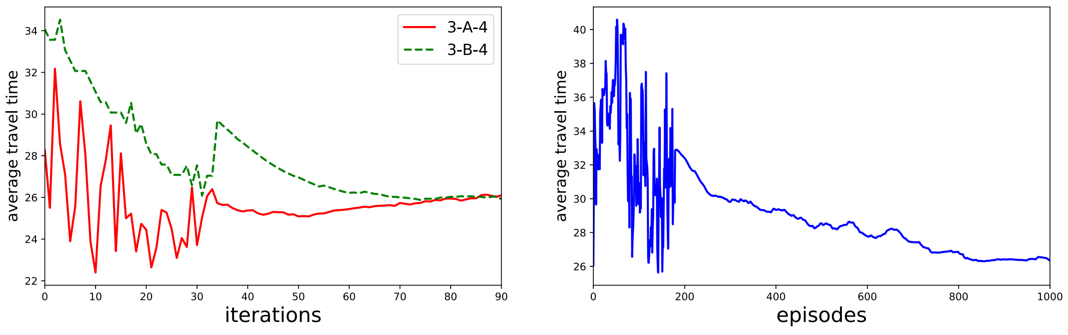
Fig. 4. Convergence plots for DUE.

At equilibrium, no individual selfish vehicle could decrease her travel cost by unilaterally deviating from her current route choice strategy. Therefore, the reward for each is simply the negative of her own travel time. In other words, each selfish agent aims to minimize her own travel time.

The simple network used to test the proposed algorithm is shown in Fig. 3. The origin and destination are node 1 and 4, respectively. When vehicles reach node 3, they have to make a route choice between link $3 \to A \to 4$ and $3 \to B \to 4$. $\tilde{O}$ is the dummy node of origin. We assume that at the beginning, vehicles are on the dummy link $\tilde{O} \to 1$, representing actual travel demand at the origin node (Ma et al., 2014). $\tilde{D}$ is the dummy node of destination. All vehicles are on the dummy link $4 \to \tilde{D}$ after reaching destination. The travel demand at the origin node $d(t)$ is $d(0) = 50$, which means there are 50 vehicles on the dummy link $\tilde{O} \to 1$ at time 0. The free flow speed is $v = 0.2$. Other parameters of this network is shown in Table 5 where link length, flow rate capacity, jam density and backward speed are denoted as L, q_{max}, k_j, and w, respectively. Note that SQ, LTM and CTM share the same link length and flow capacity with PQ. LTM and CTM share the same link capacity with SQ.

We make a comparison of the developed algorithm and a traditional iterative method (See Appendix A.2). We plot the convergent performance based on environments of PQ, SQ, LTM and CTM. Convergence plots of both methods for the DUE scenario are presented in Fig. 4. The x-axis represents the number of iterations/episodes and the y-axis is the average travel time. In Fig. 4a, the iterative method reaches convergence when the average travel times on both links become the same. After 40 iterations, the average travel times on both links $3 - A - 4$ and $3 - B - 4$ converge. The converged average travel time for $d(0) = 50$ is 13.7. At the equilibrium, there are 20 and 30 vehicles choosing link $3 - A - 4$ and $3 - B - 4$, respectively.

For the MF-MA-DQL algorithm, after bouncing back and forth in Fig. 4(b), average travel time gradually reaches its converged value 13.7, which is consistent with the iterative method. In this case, no spillback happens and travel times in all DNL environments converge to the same value.



(a) The iterative method (b) MF-MA-DQL algorithm for DUE

Fig. 5. Convergence plots for DUE.

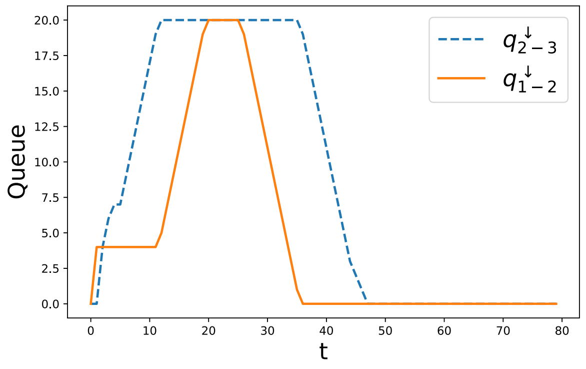


Fig. 6. Queue spillback on the network for DUE.

Example 4.2. In this example, we investigate a special case on the simple network where queue spillback may happen. The dynamic loading environment we use is LTM. The travel demand at the origin node $d(t)$ is $d(0) = 90$, which means there are 90 vehicles on the dummy link $\tilde{O} \to 1$ at time 0. The free flow speed is $v = 0.2$ and the backward speed is $w = 0.1$. The length of link $1 \to 2$, $2 \to 3$, $3 \to A \to 4$, $3 \to B \to 4$ are 0.2, 0.2, 0.4 and 0.4, respectively. The flow rate capacity of link $1 \to 2$ and $2 \to 3$ is 8. The flow rate capacity of dummy link $\tilde{O} \to 1$ is 4, which means at each time step, four vehicles can leave the origin node. The flow rate capacity of link $3 \to A \to 4$ and $3 \to B \to 4$ is:

$$q_{max} = \begin{cases} 2, & t \leqslant 5 \\ 1, & t > 5 \end{cases}$$

Convergence plots of both methods for the DUE scenario are presented in Fig. 5. In Fig. 5a, the iterative method reaches convergence when the average travel times on both links become the same. After 80 iterations, the average travel times on both links $3 - A - 4$ and $3 - B - 4$ converge. The converged average travel time for $d(0) = 90$ is 26.

For the MF-MA-DQL algorithm, after bouncing back and forth during the first 200 episodes in Fig. 4(b), average travel time gradually reaches its converged value 26, which is consistent with the iterative method. At the equilibrium, there are 45 and 45 vehicles choosing link $3 - A - 4$ and $3 - B - 4$, respectively.

Fig. 6 plots the queue spillback on the network for DUE. The x-axis represents the time and the y-axis represents the queue length. The queue length $q^{\downarrow}(t)$ at each time step is the number of vehicles in the downstream queue of each link (Ma et al., 2014; Osorio et al., 2011; Yperman, 2007) in the LTM environment, which is calculated as:

$$q^{\downarrow}(t) = N^{\uparrow}(t - \frac{L}{v} + \Delta t) - N^{\downarrow}(t).$$

It is shown that the queue forms on the link $2 - 3$ when $t = 12$ due to the limited link capacity of its downstream links $3 - A - 4$ and $3 - B - 4$. After that, the queue moves backward and forms on the link $1 - 2$ when $t = 20$ due to the limited link capacity of its downstream link $2 - 3$.

Example 4.3. In this example, we apply the developed algorithm to the OW network (13 nodes and 24 links) based on the LTM environment. The free flow speed is $v = 0.2$ and the backward speed is $w = 0.1$. Other parameters of the OW network is shown in Fig. 7. The origin is 1 and the destination is 13. The travel demand is $d(0) = 20$.

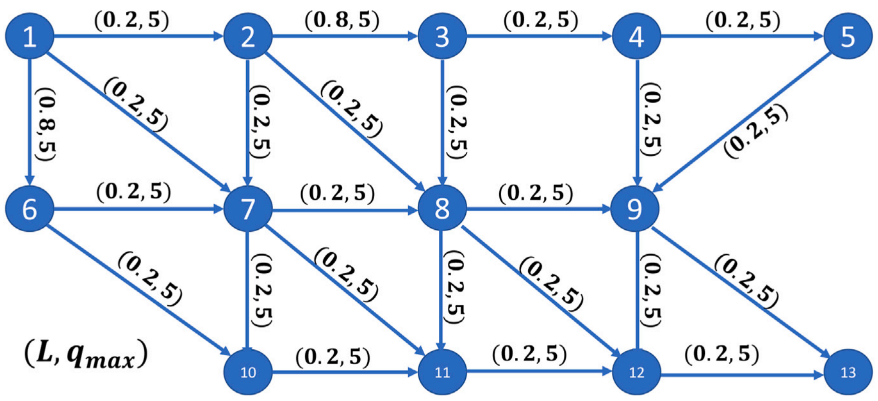


Fig. 7. OW network.

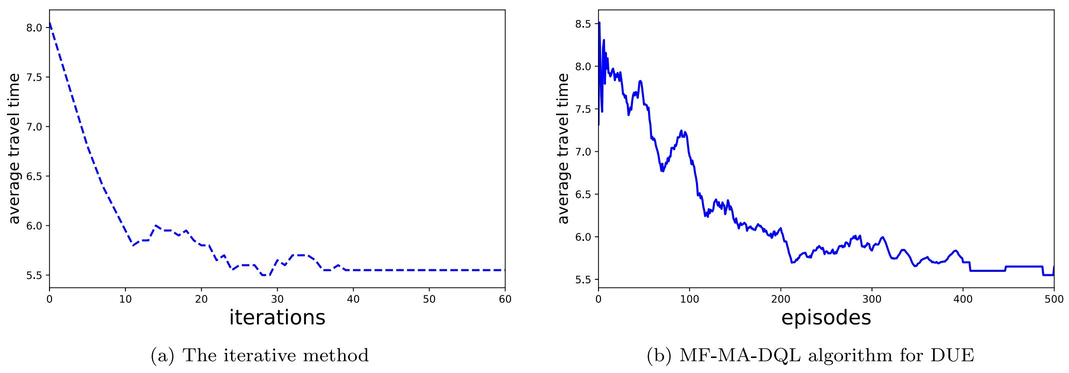


Fig. 8. Convergence plots for DUE.

Convergence plots of our algorithm and the traditional iterative method are presented in Fig. 8. In Fig. 8a, the average travel time converges to 5.5 after 40 episodes. At the equilibrium, there are 9, 1, 4 and 6 vehicles choosing paths $1-2-8-9-13$, $1-7-8-9-13$, $1-7-8-12-13$ and $1-7-11-12-13$, respectively. For the MF-MA-DQL algorithm, after bouncing back and forth during the first 300 episodes in Fig. 8(b), average travel time gradually reaches its converged value around 5.5, which is consistent with the iterative method.

Example 4.4. In this example, we apply the developed algorithm to a real-world road network with 69 nodes and 166 links, as presented in Fig. 9. This road network covers the area from 133th Street (south) to 146th Street (north) and from Riverside Drive (west) to Convent Avenue (east) in upper Manhattan. In addition to Riverside Drive and Convent Avenue, there are Broadway and Amsterdam Avenue in the north–south direction. The road network is imported into SUMO.

We first briefly introduce the environment. There are two types of vehicles in the road network, namely controllable agents and background traffic. The former actively adapts her route choices while the latter constantly follows some prescribed travel pattern. There are four groups of controllable agents in this case study, namely the first group traveling from node 14 to node 60, the second group from node 15 to node 60, third group from node 48 to node 1, and the fourth group from node 69 to node 1. In total there are 120 controllable agents who enter the road network according to their starting time. On average, there are 12 controllable agents entering the road network every 50 s. With respect to the background traffic, there are in total around 1600 vehicles in the south–north direction and 500 vehicles in the east–west direction within the simulation time period (i.e., 1000 s).

The goal of each controllable agent is to minimize her travel cost from origin to destination. Noticing that the first two groups of controllable agents share the same destination, i.e., node 60, these agents thus share the same Q function. We denote this Q function as Q^{60}, where the superscript 60 indicates that this Q function is used by agents whose destination is node 60. Obviously, Q value at destination node is 0, i.e., $Q^{60}(o_i = (60, t), a_i, \bar{a}_i) = 0$, regardless of time t, action a_i, and mean action $\bar{a}_i$. Similarly, the remaining controllable agents whose destination is node 1 share the same Q function, which is denoted by Q^1. $Q^1(o_j = (1, t), a_j, \bar{a}_j) = 0$.

To demonstrate the effectiveness of the proposed MF-MA-DQL algorithm, we now apply it to the aforementioned setup (i.e., 120 controllable agents and thousands of vehicles as background) with an accident blocking the link connecting node 26 and node 27,

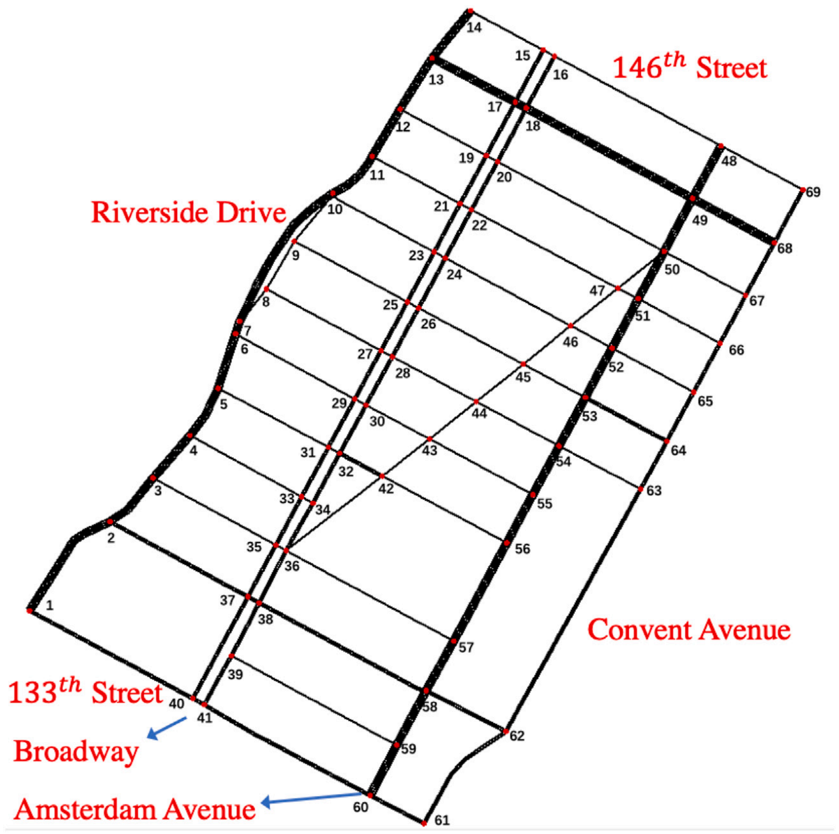


Fig. 9. Road network in SUMO.

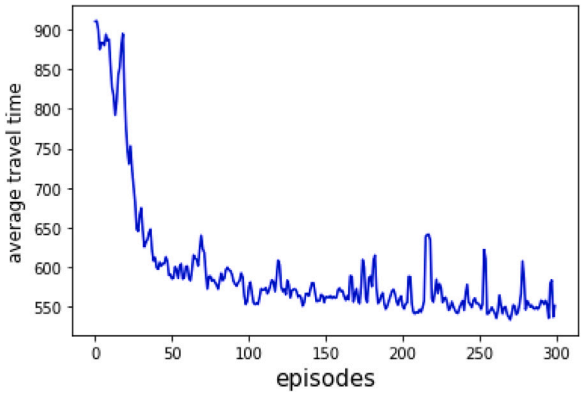


Fig. 10. MF-MA-DQL algorithm (accident occurs).

starting at $t = 180$ s and ending at $t = 490$ s. The convergence plot is presented in Fig. 10. The y-axis is the average travel time of all controllable agents and the x-axis is the index of episodes in training. Initially, agents spend a very long time (i.e., more than 900 s) on traveling to their destinations. Actually, some agents may not be able to reach their destinations and could record a total travel time of 1000 s (i.e., the maximum allowed time in one episode). During the first 50 episodes, agents explore the road network and learn towards a better policy pretty fast. Therefore, their average travel time is substantially decreased from above 900 s to around 600 s. Despite some bouncing back and forth between 50 episodes and 300 episodes, the average travel time of all controllable agents stabilizes around 570 s after 100 episodes. Given that there is an accident happening from $t = 180$ s and lasting for more than 300 s, one natural question arises, i.e., whether agents' route choice behavior is changed by this accident. Therefore,

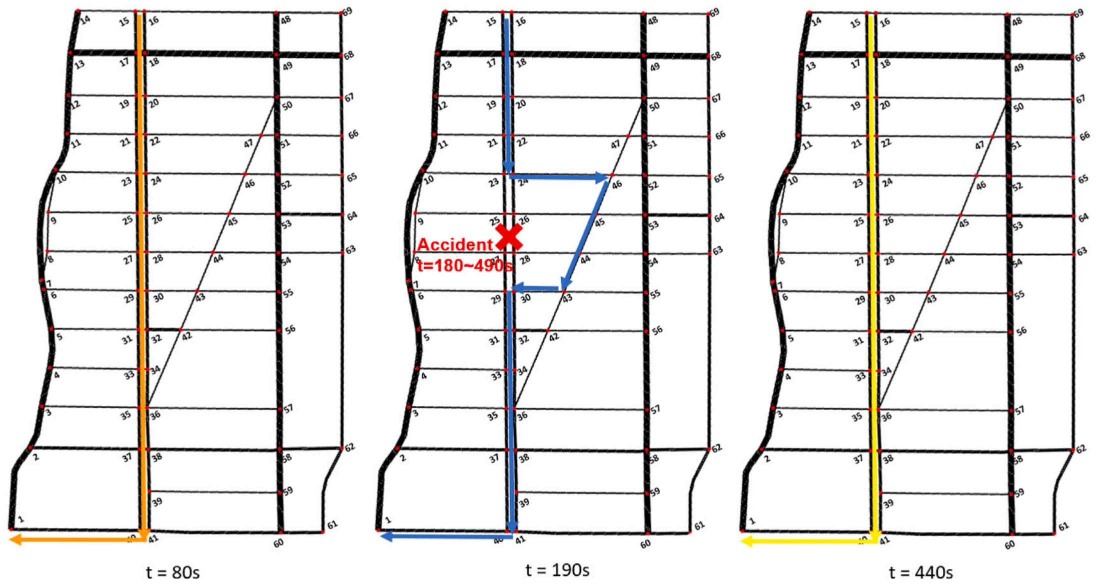


Fig. 11. Route choice of controllable agents at different time steps.

Table 6
Computation time.

	MARL	Fixed point
4.1 Simple network (50 agents, no spillback)	1 min 29 s	11 s
4.2 Simple network (90 agents, spillback)	6 min 37 s	57 s
4.3 OW network (20 agents)	2 min 7 s	19 s
4.4 SUMO simulation (120 agents, 2100 background vehicles)	12 h (Simulation: 10 h)	–

we visualize agents' route choices before, during, and around the end of the accident in Fig. 11. At $t = 80$ s, i.e., before the accident, controllable agents going to node 1 mainly choose Broadway instead of other north–south roads (i.e., Riverside drive, Amsterdam avenue, and Convent avenue) due to a comparatively light traffic condition on Broadway. At $t = 190$ s, an accident occurs on the link connecting node 25 and node 27, and thus no vehicle could reach node 27 from node 25. In other words, no vehicle could go south by entirely staying on Broadway. As one may see in the middle plot of Fig. 11, controllable agents have learned the accident and are able to avoid being stuck in traffic by using the adjacent link as a slight detour. The route choice of agents at $t = 440$ s (i.e., around the end of the accident) show that some controllable agents are now willing to wait in traffic until the traffic accident is resolved, indicating that from the perspective of these agents, detouring may be more costly than waiting in traffic when they have learned that the accident is about to end.

We list the computation time of all numerical examples in Table 6. In Examples 4.1–4.3, the computation time of the MARL approach is the running time of the training process until agents stabilize their behaviors. The computation time of the fixed point method is the running time of the iterative process until traffic flow reaches a stationary point. It is shown that the computation time of solving the Markov game is longer than that of the fixed point method. The explanation is in the Markov game, each individual agent needs to learn from the traffic environment and update their own policies. In Example 4.4, the computation time is the running time of the training process with traffic simulated in SUMO. Note that the computation time in this example is much longer than others. This is because the simulation is time-consuming. For each episode, the running time of the simulator is set as 20 min. The total time of simulation is $300 \cdot 20 = 10h$.

5. Conclusion

This paper develops a Markov game and multi-agent reinforcement learning that models intelligent agents' learning behavior and the system's equilibrating processes in a routing game among atomic selfish agents. A multi-agent mean field deep Q-learning algorithm is developed to tackle the route choice task of multiple self-interested agents. The proposed mean action, which is defined as the traffic flow on the chosen link, carries partial but not full information of nearby agents and is thus able to partially capture the competition among agents. In addition, the use of the mean action enables Q-value sharing and policy sharing, which is computationally efficient. The developed algorithm is demonstrated in four numerical examples, namely a simple network with various DNL models as its traffic propagation mechanism, the OW network with LTM, and a large-sized real-world road network with 69 nodes and 166 links implemented in SUMO.

The linkage between the classic DUE paradigm and our proposed MARL paradigm is demonstrated schematically. The difference is that, DTA models stipulate the system evolution dynamics and the travel cost of using each link or route, while the routing game does not contain such information and thus requires agents to learn and adapt their strategy according to historical episodes generated from interactions. Specifically, the developed Markov routing game depicts DUE assuming perfect information and deterministic environments propagated by DNL models. On the simple network without and with spillback, we further show that the numerical solution from the developed MARL algorithm agrees well with the DUE from the iterative method.

Nevertheless, there are several future directions we would like to explore. First, departure time choice and velocity control (Huang et al., 2021) are two travel choices of intelligent agents. The proposed model-free MARL paradigm will be extended to accommodate departure time choice and velocity control for discrete travelers. Second, in this paper, we assume drivers make decisions when they reach a node. In reality, drivers may make routing choices before they arrive at a node. We would like to incorporate such behavior in future work. Last but not the least, we will develop a bi-level network design problem that optimizes planning or policy countermeasures in the upper level and the adaptive routing behavior of discrete travelers in the lower level.

CRediT authorship contribution statement

Zhenyu Shou: Conceptualization, Formal analysis, Methodology, visualization, Writing. **Xu Chen:** Validation, Visualization, Writing. **Yongjie Fu:** Validation, Visualization. **Xuan Di:** conceptualization, Methodology, Supervision, Writing – review & editing.

Acknowledgments

This work is partially sponsored by the Region 2 University Transportation Research Center (UTRC) (subcontract RUTGER PO 966112/PID#824227) and the National Science Foundation under CAREER award number CMMI-1943998. The authors would also like to thank the three reviewers for their constructive comments.

Appendix

A.1.

PQ Environment: The sending flow in the PQ environment is calculated in the same way as LTM. Different from the LTM environment, PQ does not capture the link capacity. Therefore, the receiving flow is only determined by the flow rate capacity:

$$R(t) = q_{max}\Delta t.$$

SQ Environment: The SQ model takes the link capacity into consideration. Different from the LTM environment, it does not capture the backward speed. The receiving flow is calculated as:

$$R(t) = min\{N^{\downarrow}(t) + k_j L - N^{\uparrow}(t),\ q_{max}\Delta t\},$$

CTM Environment: In the CTM environment, each link is divided into cells. The length of each cell is supposed to be the distance traveled by a vehicle at free flow speed v. The sending flow $S(t)$ is the number of vehicles which can leave a cell at each time step. It is calculated as:

$$S(t) = min\{n(t), q_{max}\Delta t\},$$

where $n(t)$ is the number of vehicles on the cell at time t. It means the number of vehicles which can leave may not exceed the number of vehicles on the cell. The receiving flow $R(t)$ is the number of vehicles which can be accommodated on the cell at time t. It is calculated as:

$$R(t) = min\{\frac{w}{v}(N - n(t)), q_{max}\Delta t\},$$

where N is the cell capacity, representing the maximum number of vehicles on a cell.

A.2.

Iterative Method: We employ the iterative method proposed in Gawron (1998) to solve DUE. The basic idea of the iterative method is to let vehicles choose routes according to some policy and then calculate the network loading and travel cost by simulation so that vehicles gradually learn towards better route choices. By iterating this process, a stationary fixed point solution is supposed to be derived so that no vehicle could get better off by switching her route choices. To make this paper self-explanatory, we now briefly introduce the iterative method adapted to the two-node two-link network. Interested readers could refer to Gawron (1998) for more details.

Algorithm 2 An iterative method to solve DUE

```
1: Initialize a proportion $p_t(0)$ indicating the proportion of the demand at $t$ choosing link 0. Consequently, $p_t(1) = 1 - p_t(0)$
2: Initialize a learning step size $\eta \in (0, 1]$
3: Initialize function $g(x) = exp(\frac{ax}{1-x^2})$, where $a$ is a parameter, and function $f(x) = \frac{p_t(0)g(x)}{p_t(0)g(x) + p_t(1)}$
4: repeat
5:     Calculate the time-dependent travel cost of both links, denoted by $\tau_t(0)$ and $\tau_t(1)$, from simulation
6:     Calculate the relative cost difference $\delta_{01} = \frac{\tau_t(1) - \tau_t(0)}{\tau_t(1) + \tau_t(0)}$
7:     Update the proportion by $p_t(0) = (1-\eta)p_t(0) + \eta f(\delta_{01})$
8:     Calculate $p_t(1) = 1 - p_t(0)$
9:     Update function $f(x) = \frac{p_t(0)g(x)}{p_t(0)g(x) + p_t(1)}$
10: until the iterative method reaches a stationary fixed point
11: Return $p_t(0)$ and $p_t(1)$
```

The iterative method is listed in Algorithm 2. With demand $d(t)$ emerging from node O at time t, we randomly initialize $p_t(0)$ and $p_t(1)$ to denote the proportion of the demand choosing link 0 and link 1, respectively. We also initialize functions $g(x)$ and $f(x)$ which will be used to update $p_t(0)$ and $p_t(1)$. Note that $f : [-1, 1] \to [0, 1]$ is a monotonic increasing function. With $p_t(0)$ and $p_t(1)$, we run simulation to calculate the average travel cost on both links for the demand starting at time t and denote these time-dependent travel cost by $\tau_t(0)$ and $\tau_t(1)$. We then calculate the relative cost difference $\delta_{01} = \frac{\tau_t(1) - \tau_t(0)}{\tau_t(1) + \tau_t(0)}$ and use this cost difference to update the proportion $p_t(0)$ towards $f(\delta_{01})$ by step size η. The rationale of this updating rule is as follows. When δ_{01} is large (e.g., close to 1), meaning that the cost on link 0 is much less than that on link 1, we have $f(\delta_{01}) \approx 1$, indicating that $p_t(0)$ is updated towards a larger value; when δ_{01} is small (e.g., close to -1), meaning that the cost on link 0 is much larger than that on link 1, we have $f(\delta_{01}) \approx 0$, indicating that $p_t(0)$ is updated towards a smaller value. After updating $p_t(0)$ and $p_t(1)$, we repeat the iterative process until reaching a stationary fixed point.

References

Bakker, B., Whiteson, S., Kester, L., Groen, F.C.A., 2010. Traffic light control by multiagent reinforcement learning systems. In: Babuška, R., Groen, F.C.A. (Eds.), Interactive Collaborative Information Systems. In: Studies in Computational Intelligence, Springer, Berlin, Heidelberg, pp. 475–510. http://dx.doi.org/10.1007/978-3-642-11688-9_18.

Ban, X.J., Pang, J.-S., Liu, H.X., Ma, R., 2012. Continuous-time point-queue models in dynamic network loading. Transp. Res. B 46 (3), 360–380.

Bazzan, A.L.C., Grunitzki, R., 2016. A multiagent reinforcement learning approach to en-route trip building. In: 2016 International Joint Conference on Neural Networks (IJCNN). pp. 5288–5295, ISSN: 2161-4407.

Bazzan, A.L.C., Klügl, F., 2008. Re-routing agents in an abstract traffic scenario. In: Zaverucha, G., da Costa, A.L. (Eds.), Advances in Artificial Intelligence - SBIA 2008. In: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 63–72.

Bhalla, S., Ganapathi Subramanian, S., Crowley, M., 2020. Deep multi agent reinforcement learning for autonomous driving. In: Goutte, C., Zhu, X. (Eds.), Advances in Artificial Intelligence. In: Lecture Notes in Computer Science, Springer International Publishing, Cham, pp. 67–78.

Brown, N., Sandholm, T., 2018. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. Science 359 (6374), 418–424.

Brown, N., Sandholm, T., 2019. Superhuman AI for multiplayer poker. Science 365 (6456), 885–890.

Chen, X., Di, X., 2021. Ridesharing user equilibrium with nodal matching cost and its implications for congestion tolling and platform pricing. Transp. Res. C 129, 103233.

Chen, C., Wei, H., Xu, N., Zheng, G., Yang, M., Xiong, Y., Xu, K., Li, Z., 2020. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34. pp. 3414–3421. Number: 04.

Daganzo, C.F., 1994. The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. Transp. Res. B 28 (4), 269–287.

Daganzo, C.F., 1995. The cell transmission model, part II: Network traffic. Transp. Res. B 29 (2), 79–93.

Di, X., Ban, X.J., 2019. A unified equilibrium framework of new shared mobility systems. Transp. Res. B 129, 50–78.

Di, X., Ma, R., Liu, H.X., Ban, X.J., 2018. A link-node reformulation of ridesharing user equilibrium with network design. Transp. Res. B 112, 230–255.

Di, X., Shi, R., 2021. A survey on autonomous vehicle control in the era of mixed-autonomy: From physics-based to AI-guided driving policy learning. Transp. Res. C 125, 103008.

Filar, J., Vrieze, K., 1997. Applications and special classes of stochastic games. In: Competitive Markov Decision Processes. Springer, pp. 301–341.

Friesz, T.L., Han, K., 2019. The mathematical foundations of dynamic user equilibrium. Transp. Res. B 126, 309–328.

Friesz, T.L., Han, K., Neto, P.A., Meimand, A., Yao, T., 2013. Dynamic user equilibrium based on a hydrodynamic model. Transp. Res. B 47, 102–126.

Gawron, C., 1998. An iterative algorithm to determine the dynamic user equilibrium in a traffic simulation model. Internat. J. Modern Phys. C 09 (03), 393–407, Publisher: World Scientific Publishing Co..

Grunitzki, R., Ramos, G.d.O., Bazzan, A.L.C., 2014. Individual versus difference rewards on reinforcement learning for route choice. In: 2014 Brazilian Conference on Intelligent Systems. pp. 253–258.

Hu, J., Wellman, M.P., 1998. Multiagent reinforcement learning: Theoretical framework and an algorithm. In: Proceedings of the Fifteenth International Conference on Machine Learning. In: ICML '98, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 242–250.

Huang, K., Chen, X., Di, X., Du, Q., 2021. Dynamic driving and routing games for autonomous vehicles on networks: A mean field game approach. Transp. Res. C 128, 103189.

Kim, G., Ong, Y.S., Cheong, T., Tan, P.S., 2016. Solving the dynamic vehicle routing problem under traffic congestion. IEEE Trans. Intell. Transp. Syst. 17 (8), 2367–2380, Conference Name: IEEE Transactions on Intelligent Transportation Systems.
Kumar, S., Shah, P., Hakkani-Tur, D., Heck, L., 2017. Federated control with hierarchical multi-agent deep reinforcement learning. arXiv:1712.08266 [cs]. arXiv:1712.08266.
Kuwahara, M., Akamatsu, T., 1997. Decomposition of the reactive dynamic assignments with queues for a many-to-many origin-destination pattern. Transp. Res. B 31 (1), 1–10.
Lam, W.H., Huang, H.-J., 1995. Dynamic user optimal traffic assignment model for many to one travel demand. Transp. Res. B 29 (4), 243–259.
Leibo, J.Z., Zambaldi, V., Lanctot, M., Marecki, J., Graepel, T., 2017. Multi-agent reinforcement learning in sequential social dilemmas. arXiv:1702.03037 [cs]. arXiv:1702.03037.
Li, J., Fujiwara, O., Kawakami, S., 2000. A reactive dynamic user equilibrium model in network with queues. Transp. Res. B 34 (8), 605–624.
Li, M., Qin, Z., Jiao, Y., Yang, Y., Wang, J., Wang, C., Wu, G., Ye, J., 2019. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In: The World Wide Web Conference. In: WWW '19, ACM, New York, NY, USA, pp. 983–994, event-place: San Francisco, CA, USA.
Lin, K., Zhao, R., Xu, Z., Zhou, J., 2018. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. In: KDD '18, ACM, New York, NY, USA, pp. 1774–1783.
Littman, M.L., 1994. Markov games as a framework for multi-agent reinforcement learning. In: Proceedings of the Eleventh International Conference on International Conference on Machine Learning. In: ICML'94, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 157–163, event-place: New Brunswick, NJ, USA.
Littman, M.L., 2001. Value-function reinforcement learning in Markov games. Cogn. Syst. Res. 2 (1), 55–66.
Ma, R., Ban, X.J., Pang, J.-S., 2014. Continuous-time dynamic system optimum for single-destination traffic networks with queue spillbacks. Transp. Res. B 68, 98–122.
Ma, R., Ban, X.J., Pang, J.-S., 2017. A link-based differential complementarity system formulation for continuous-time dynamic user equilibria with queue spillbacks. Transp. Sci. 52.
Mao, C., Shen, Z., 2018. A reinforcement learning framework for the adaptive routing problem in stochastic time-dependent network. Transp. Res. C 93, 179–197.
Matignon, L., Laurent, G.J., Fort-Piat, N.L., 2012. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. Knowl. Eng. Rev. 27 (1), 1–31.
Merchant, D.K., Nemhauser, G.L., 1978a. A model and an algorithm for the dynamic traffic assignment problems. Transp. Sci. 12 (3), 183–199, Publisher: INFORMS.
Merchant, D.K., Nemhauser, G.L., 1978b. Optimality conditions for a dynamic traffic assignment model. Transp. Sci. 12 (3), 200–207, Publisher: INFORMS.
Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., 2015. Human-level control through deep reinforcement learning. Nature 518 (7540), 529–533.
Nguyen, T.T., Nguyen, N.D., Nahavandi, S., 2020. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. IEEE Trans. Cybern. 50 (9), 3826–3839. http://dx.doi.org/10.1109/TCYB.2020.2977374, Conference Name: IEEE Transactions on Cybernetics.
Nie, Y.M., 2011. A cell-based Merchant-Nemhauser model for the system optimum dynamic traffic assignment problem. Transp. Res. B 45 (2), 329–342.
Nie, X., Zhang, H.M., 2005. A comparative study of some macroscopic link models used in dynamic traffic assignment. Netw. Spat. Econ. 5 (1), 89–115.
OpenAI, 2018. OpenAI Five. https://blog.openai.com/openai-five/.
Ortuzar, J.d.D., Willumsen, L., 2011. Modelling Transport. http://dx.doi.org/10.1002/9781119993308.
Osorio, C., Flötteröd, G., Bierlaire, M., 2011. Dynamic network loading: a stochastic differentiable model that derives link state distributions. Procedia-Soc. Behav. Sci. 17, 364–381.
Palanisamy, P., 2019. Multi-agent connected autonomous driving using deep reinforcement learning. arXiv:1911.04175 [cs, stat]. arXiv:1911.04175.
Pérolat, J., Strub, F., Piot, B., Pietquin, O., 2017. Learning Nash equilibrium for general-sum Markov games from batch data. In: Artificial Intelligence and Statistics. PMLR, pp. 232–241.
Prasad, A., Dusparic, I., 2019. Multi-agent deep reinforcement learning for zero energy communities. In: 2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe). pp. 1–5.
Puterman, M.L., 1994. Markov Decision Processes: Discrete Stochastic Dynamic Programming, first ed. John Wiley & Sons, Inc., New York, NY, USA.
Ramos, G.d.O., Bazzan, A.L.C., da Silva, B.C., 2018. Analysing the impact of travel information for minimising the regret of route choice. Transp. Res. C 88, 257–271.
Roughgarden, T., 2007. Routing games. In: Algorithmic Game Theory, Vol. 18. Cambridge University Press, Cambridge, MA, pp. 459–484.
Seongmoon Kim, Lewis, M.E., White, C.C., 2005. Optimal vehicle routing with real-time traffic information. IEEE Trans. Intell. Transp. Syst. 6 (2), 178–188, Conference Name: IEEE Transactions on Intelligent Transportation Systems.
Shou, Z., Di, X., 2020. Reward design for driver repositioning using multi-agent reinforcement learning. Transp. Res. C 119, 102738.
Shou, Z., Di, X., Ye, J., Zhu, H., Zhang, H., Hampshire, R., 2020. Optimal passenger-seeking policies on E-hailing platforms using Markov decision process and imitation learning. Transp. Res. C 111, 91–113.
Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D., 2016. Mastering the game of Go with deep neural networks and tree search. Nature 529 (7587), 484–489.
Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., Hassabis, D., 2017. Mastering the game of Go without human knowledge. Nature 550 (7676), 354–359.
Solan, E., Vieille, N., 2015. Stochastic games. Proc. Natl. Acad. Sci. 112 (45), 13743–13746.
Stefanello, F., Silva, B.C.d., Bazzan, A.L.C., 2016. Using topological statistics to bias and accelerate route choice: Preliminary findings in synthetic and real-world road networks. In: ATT@IJCAI.
Sutton, R.S., Barto, A.G., 1998. Introduction to Reinforcement Learning, first ed. MIT Press, Cambridge, MA, USA.
Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J.P., Jaderberg, M., Vezhnevets, A.S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T.L., Gulcehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama, D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T., Kavukcuoglu, K., Hassabis, D., Apps, C., Silver, D., 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature 575 (7782), 350–354.
Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., Wang, J., 2018. Mean field multi-agent reinforcement learning. In: International Conference on Machine Learning. pp. 5571–5580.
Yperman, I., 2007. The link transmission model for dynamic network loading. URL: https://www.researchgate.net/publication/28360292_The_Link_Transmission_Model_for_dynamic_network_loading.
Zhang, H., Nie, Y., Qian, S., 2013. Modelling network flow with and without link interactions: The cases of point queue, spatial queue and cell transmission model. Transportmetr. B 1, 33–51.
Zhou, B., Song, Q., Zhao, Z., Liu, T., 2020. A reinforcement learning scheme for the equilibrium of the in-vehicle route choice problem based on congestion game. Appl. Math. Comput. 371, 124895.
Ziliaskopoulos, A.K., 2000. A linear programming model for the single destination system optimum dynamic traffic assignment problem. Transp. Sci. 34 (1), 37–49, Publisher: INFORMS.