Game-Theoretic Cooperative Lane Changing Using Data-Driven Models

Guohui Ding, Sina Aghli, Christoffer Heckman, and Lijun Chen*

***Abstract*— Self-driving cars are being regularly deployed in the wild, and with them has come the concern of how such autonomous vehicles will cooperate. This is one of the most important next hurdles for autonomous driving: how will such vehicles optimally interact with one another and with their surroundings? It is feasible that a well-developed strategy for the interaction and collaboration of vehicles on the road will bring more efficient performance of a system of cars as a whole. In this work, we start with the lane change present to all vehicles and develop the agent-environment and agent-agent relationships through a combination of deep reinforcement learning (DRL) and game theory. We introduce a passive-proactive lane change framework and compare the independent, asynchronous lane change scheme as in a single-agent RL setting with a synchronous strategic scheme consisting of a Markov game which is solved optimally in the framework of DRL. In the synchronous scheme, the passive agent and proactive vehicles are trained cooperatively based on the hypothesized Nash equilibrium of the Markov game to finish the lane changing maneuver. To test our approach, we apply a physics engine over a high fidelity model that has previously been shown to well-describe experimental four-wheeled vehicles. The framework developed here demonstrates the potential of using reinforcement learning to solve cooperative autonomous vehicle tasks as they are posed as Markov games.**

I. Introduction

The development of autonomous vehicles is in full swing, concentrating largely on their ability to localize within an environment, make safe decisions in reaching their destination, and many human-centered factors that are complex and fundamentally difficult to enumerate. Some recent work has focused on the fascinating topic of improving their traffic performance and road space usage; this work aims to allow autonomous vehicles to cooperate with traditional vehicles while also optimizing road usage based on autonomous vehicles' potential capabilities to meaningfully interact with one another. However, these challenges can be quite complex due to numerous scenario-dependent factors. For example, a vehicle might be more aggressive in overtaking others on city streets compared to on the highway. As one of the basic operations of a road-bound vehicle, the lane change task has been studied for a long time; there is an incredible diversity of strategies in this task, and many have analyzed it [1], [2], [3], [4].

In traditional driving, the driver's intent and predelictions play a crucial role in lane change even in the presence of driver assists such as lane-drift detection or adaptive cruise control. The subtle evaluation-action-correction interactions between drivers are particularly ridden with complexities. As a result of this, autonomous vehicles need well-designed strategies and controllers to deal with the uncertainty and human-executed optimization that occurs in the process of merging lanes.

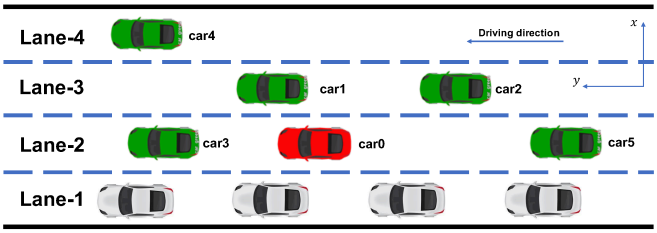


Fig. 1: A diagram of the lane change problem. The highlighted $car0$ is attempting to merge between $car1$ and $car2$ while avoiding a collision with $car3$ and minimizing the amount of slowing on the part of $car5$.

In this paper, we mainly consider two problems. The first is developing a cooperative framework to characterize the lane change process. This framework should consider the interactions among vehicles actually involved in lane change tasks and the vehicle-environment (e.g., other vehicles or freeway/urban) connections. The second is determining the proper moment to take merging actions in a multi-vehicle scenario. In this case, a merging vehicle must determine the proper actions to take based on its state so that it may adjust its behavior to find the optimal moment to change lanes.

Since reliable and scalable models of human driving patterns can be difficult to obtain, we pursue a model-free approach to solve for actions given the observed states. We apply deep reinforcement learning (DRL) [5] to solve the multi-agent Markov game of three vehicles – a proactive-passive pair, and a reference vehicle that provides the forward boundary for the merge – that models the lane change problem.

In this paper, we aim to enable vehicles to finish the lane change process strategically by considering the following problems: 1) how can we characterize the lane change process, 2) what kind of schemes do vehicles select in order to work strategically with respect to others' schemes, and 3) which moment is the proper one to finish adjustment and start merging. In addressing these issues, the main contributions of the paper are as follows:

- We build a passive-proactive multi-agent RL lane change framework so that the roles of different vehicles in the lane change process are clear.

This work was supported by NSF award No. 1646556.

All authors are with the Department of Computer Science, University of Colorado, Boulder, CO 80309, USA

*Corresponding author; e-mail: lijun.chen at colorado.edu

- We propose a synchronous scheme for passive-proactive lane changing.
- According to the requirements of the lane change problem, we build a game theoretic deep Q-learning algorithm to help autonomous vehicles successfully complete lane change tasks.

II. Related work

Traditionally, reinforcement learning (RL) has exclusively considered the optimization of single-agent policies in static environments. Unfortunately, the lane change task on roadways, along with many other tasks in this environment, is a multi-agent participation process. One challenge that arises as a result of this multi-agent RL (MARL) problem is that the environment will become non-stationary, rendering algorithms seeking stationary solutions vulnerable to thrashing. This is due to the fact that one agent's policy update will be considered as the non-stationary disturbance of the environment by other agents. To remedy this, one intuitive solution is to enforce that the current agent only consider the other agents as part of the environment and use the single-agent RL method directly. For example, Tan et al. [6] analyzed the performance of collaboration vs. independent action in a mixed-agent scenario. Matignon et al. [7] showed that in the uncoordinated case, independent learning results in non-stationary and shadowed equilibria. However, these approaches generally result in the current agent not leveraging all prior information when in cooperative or competitive scenarios.

A significant amount of effort has been devoted to determining how best to optimize the value function approximation and how to overcome local minima or sluggish convergence on the dynamic programming side of the problem. To address this, Littman et al. [8], [9] proposed that Markov games could be leveraged to address the MARL problem. This minimax-Q algorithm focuses on the zero-sum game of the agents such that agents policies will converge to a fixed,"safe" strategy.

Separately from MARL, recent advances in DRL have shown incredible utility on a variety of challenging RL problems. DRL is an extension to classical reinforcement learning where the policy, value function and/or underlying model are replaced with a deep neural network. This method has been applied to many different problems, including optimizing dialogue management [10], governing a quadrupedal trot gait [11], and designing a controller for helicopter with inverted flight [12]. At the core of these methods is deep learning (DL) [13], i.e. the use of deep neural networks as data-driven high dimensional function approximations, which have provided a computational method to represent functions with a controllable complexity. Introducing DL into reinforcement learning has generally provided greater flexibility for the decision-making process and improved computational efficiency through the leveraging of graphical processing units (GPUs). For example, Mnih et al.[14] show that deep Q-learning can be applied play Atari 2600 games based only on the raw pixel data as input with amazing performance.

Especially recent work has shown incredible promise in the proliferation of these methods to achieve control problems. For example, [15] mapped raw RGB-D observations directly to torques in order to assist the learning of control inputs for manipulation. Others have also extended the utility of DRL by both improvements on the optimization process [16] and designing neural network architectures [17]. As part of the former effort, Bahdanau et al. [18] proposed an actor-critic algorithm to generate sequences in natural language processing (NLP) problems. So-called Deep Q-Networks (DQNs), named for their replacement of the Q function in RL with a deep neural network, have even reached human-level performance on some video games [19]. Recently this has been extended to other problems via the AlphaGo [20] and AlphaGo Zero [21] frameworks.

Very recently, Tampuu et al. [22] considered the two-player pong game using DQN for learning collaborative playing strategies. Building on these successes, Jakob et al. built a general decentralized multi-agent actor-critic algorithm called *counterfactual multi-agent* (COMA), which considered a centralized critic function [23] to approximate the Q-function and subsequently optimize agent policies. COMA demonstrated impressive improvement of average performance when tested in *StarCraft unit micromanagement*. Meanwhile, Lowe et al. [24] proposed a general-purpose multi-agent learning algorithm by considering the one-to-one critic functions for each agent under the mixed cooperative-competitive environment. Their work only used local information to handle cooperative, competitive, or mixed interaction behaviors of agents without assuming a differentiable model. Other recent works [25], [26], [27], [28] also considered these cooperative-agent problems in the context of RL.

III. Methodology

In this section, we briefly review some background knowledge on the method that we will use.

A. Single-agent RL

In the standard single-agent RL setting, one agent interacts with the environment over discrete times. At time t, the agent observes the state $s_t \in S$ and executes the action $a_t \in A$ based on certain policy $\pi : S \to PD(A)$ where $PD(\cdot)$ represents the probability distribution over a certain space. The environment then transits to the next state $s_{t+1} \sim P(\cdot|s_t, a_t)$ through transition function $P : S \times A \to PD(S)$, and the agent attains a reward $r_t : S \times A \to \mathbb{R}$. Given an initial state s at time t, we aim to find a policy π to maximize the expected total rewards $\mathbf{E}[\sum_{k=0}^{T-t} \gamma^k r_{t+k}|s_t = s]$ with a discount factor $\gamma \in (0, 1)$. There are two general approaches to solving a RL problem: policy optimization and dynamic programming.

1) Policy Optimization: This approach tries to find the optimal policy through direct searching or solving the corresponding optimization problem. For

instance, policy gradient method considers a class of polices $\pi_\theta = \pi(a|s;\theta)$ that are parameterized by θ. Denoted the total discounted reward under the policy with parameter θ by $J(\theta) = \mathbf{E}_{\tau\sim p(\tau;\theta)}[\sum_{t\geq 0}\gamma^t r_t]$, where $p(\tau;\theta)$ is the distribution over the trajectory $\tau \doteq (s_0, a_0, r_0, s_1, a_1, r_1, \cdots)$. We can update and find the optimal θ (i.e., the optimal policy) by the gradient method; see, e.g., [29], [30], [15]. In particular, the REINFORCE algorithm[29] uses the sample of trajectory directly to update θ. To address the high variance brought by the direct use of the gradient $\nabla_\theta J(\theta)$, Vanilla REINFORCE introduces unbiased baseline $b(s_t)$ such that $\nabla_\theta J(\theta) = \mathbf{E}_\tau\left[\sum_{t=0}^T\left(\sum_{t'=t}^T \gamma^{t'-t} r_{t'} - b(s_t)\right)\nabla_\theta \log \pi_\theta(a_t|s_t)\right]$ has low variance [30].

2) Dynamic Programming: We will use one of the dynamic programming approaches – Q-learning method (Watkins, 1989) and its variants in this paper. To maximize the state-value function $V_\pi(s) = \mathbf{E}[\sum_{k=0}^{T-t}\gamma^k r_{t+k}|s_t = s, \pi]$ with the initial state s at time t, we define action-state value function $Q_\pi(s,a) = \mathbf{E}[\sum_{k=0}^{T-t}\gamma^k r_{t+k}|s_t = s, a_t = a, \pi]$. Then for any $(s,a) \in S \times A$, a policy π^* maximizing $Q_\pi(s,a)$ also maximizes V_π. Furthermore, the optimal $Q_{\pi^*}(s_t, a_t)$ satisfies the following *Bellman equation*:

$$Q_{\pi^*}(s_t, a_t) = r_t + \gamma \mathbf{E}_{a_{t+1}\sim\pi^*, s_{t+1}} Q_{\pi^*}(s_{t+1}, a_{t+1}), \quad (1)$$

which can be solved iteratively with certain deterministic function $\phi : S \to A$ according to

$$\begin{aligned} Q_{t+1}(s_t, a_t) \quad = \quad & (1-\alpha_t)Q_t(s_t, a_t) \\ & +\alpha_t[r_t + \gamma Q_t(s_{t+1}, \phi(s_{t+1}))] \end{aligned} \quad (2)$$

under certain condition [5].

Mnih *et al.* recently developed deep Q-learning [14], [19] that integrates neural networks with Q-learning method. They introduced the approximation $Q_{\pi^*}(s,a) \approx Q(s,a;\theta)$ including additional replay buffer and target neural network. At training iteration i with current state s_t, the objective is minimizing the loss function $L_i(\theta_i) = \mathbf{E}_{s_t}[(y_i - Q(s_t, a_t; \theta_i))^2]$ with $y_i = \mathbf{E}_{s_{t+1}}[r_t + \gamma \max_a Q(s_{t+1}, a; \theta_{i-1})]$, and the stochastic gradient descent is used to update neural network weights θ_i where the gradient is given by

$$\begin{aligned} \nabla_{\theta_i} L_i(\theta_i) = \mathbf{E}[&\Big(r_t + \gamma \max_a Q(s_{t+1}, a; \theta_{i-1}) - Q(s_t, a_t; \theta_i)\Big) \\ &\nabla_{\theta_i} Q(s_t, a_t; \theta_i)]. \end{aligned} \quad (3)$$

B. Markov Game

A Markov game [8], also called stochastic game, is defined as $\{\mathcal{N}, S, (A^k)_{k\in\mathcal{N}}, P, (r^k)_{k\in\mathcal{N}}, (\pi^k)_{k\in\mathcal{N}}\}$ where $\mathcal{N} = \{1, \cdots, n\}$ is the set of agents, S the set of states, A^k the set of actions available for agent k, $P : S \times A^1 \times \cdots \times A^n \to PD(S)$ the state transition function, $r^k : S \times A^1 \times \cdots \times A^n \to \mathbb{R}$ the agent k's reward function, and $\pi^k : S \to PD(A^k)$ the agent k's policy. Each agent k aims to maximize its expected discounted total rewards with the starting state s at time t,

$$V_{\pi^k}(s) = \mathbf{E}\left[\sum_{i=0}^{T-t}\gamma^i r^k_{t+i} | s_t = s, (\pi^k)_{k\in\mathcal{N}}\right]. \quad (4)$$

In this paper, we will use a two-agent Markov game to model the lane change problem, solve the resulting game based on the deep Q-learning method.

IV. Lane Change Problem

The lane change problem has received lots of research; see, e.g., [31] that investigates how to adjust the actions in an aggressive lane-change situation to improve the traffic flow, [32] that introduces an integrated model to jointly consider the mandatory and discretionary lane-change behaviors, and [33] that proposes a data-driven model to simulate the lane change process. In this paper, we will study the collaborative lane change in a system of autonomous vehicles. We assume that each vehicle can observe the states of other vehicles with bounded errors, but no vehicle is at a position to (globally) coordinate the lane change process. Instead, the vehicles will cooperatively carry out the lane change through playing an equivalent Markov game.

As shown in Fig. 1, we consider a simplified while realistic lane change setting, where *car0* sends the signal to change its lane into lane-3, *car2* responds to this signal to create the merging space, and other neighboring vehicles are assumed to be not responsive (NR) to the signal. The lane change process includes an adjustment stage and a merging stage. During the adjustment stage, *car0* will keep adjusting its position and speed waiting for the merging moment, while *car2* is adjusting its position and speed to create the merging space. When the adjustment is finished, *car0* will take the merging action to complete the lane change process.

The above lane change setting results in a proactive-passive system with one proactive agent (*car0*), one passive agent (*car2*), and one reference agent (*car1*); see Table I. We empower the vehicles with RL to guide their actions, for instance, when the proactive agent switches from the adjustment stage to the merging stage.

Characters	Purpose	Examples
Reference agent	NR	*car1*
Proactive agent	Learn to take actions to merge	*car0*
Passive agent	Learn to create merging space	*car2*
Others	NR	*car3, car4, car5*

TABLE I: Vehicle roles

Specifically, the set of agents $\mathcal{N} = \{1, 2\}$ corresponding to the proactive and passive agents/vehicles. The state at time t is denoted by $s_t = (s_t^k)_{k\in\mathcal{N}}$, with $s_t^k = (x_t^k, y_t^k, v_t^k)$ where (x_t^k, y_t^k) denotes the 2D position/coordinate and v_t^k the linear velocity of agent k. At time t, vehicle k's action is denoted by $a_t^k \doteq (s, u)$, where $s \in \{1\ (acceleration), 0\ (keep), -1\ (deceleration)\}$ and $u \in \{1\ (veer-left), 0\ (keep), -1\ (veer-right)\}$. The basic actions include acceleration $(1, 0)$, deceleration $(-1, 0)$, keep $(0, 0)$, veer-left $(0, 1)$, and veer-right $(0, -1)$. We assume that merging action is either veer-left or veer-right. Since we focus on the proper stage-switching moment, the vehicle will keep on its lane before that and the basic operations include speed up and slow down. When the merging configuration is reached, the proactive vehicle takes the merging action.

As for the state dynamics of the vehicle, we use the *double integrator* model:

$$\begin{bmatrix} y_{t+1}^k \\ v_{t+1}^k \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_t^k \\ v_t^k \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u^k(a_t^k)\Delta t + \varepsilon_t, \quad (5)$$

where $u^k(a_t^k)$ specifies the input magnitude upon action a_t^k and ε_t is the disturbance. The transition function is $s_{t+1} \sim P(\cdot|s_t,(a^1,a^2))$ follows from the above dynamics.

The reward function is built by considering the possible risks and state changes, as follows:

$$r_t^k(s_t,(a_t^1,a_t^2)) = \begin{cases} dist(y_t^k), & \text{otherwise,} \\ 2, & \text{if} |dist(y_t^k)| < \eta, \eta > 0, \\ 20, & \text{if success,} \\ -10, & \text{if collision,} \end{cases}$$

where, based on the lane change setting in Fig. 1, $dist(y_t^1) = -c_1 \min\{y_t^1 - \underline{y}_t^{car2}, 0\} - c_2 \max\{y_t^1 - \overline{y}_t^{car1}, 0\} - c_3 \max\{y_t^1 - \underline{y}_t^{car3}, 0\}$ and $dist(y_t^2) = -c_4 \max\{y_t^2 - \overline{y}_t^{car1}, 0\} - c_5 \max\{y_t^2 - \underline{y}_t^{car5}, 0\}$, with $\overline{y}_t^k = y_t^k - d_{safe}$, $\underline{y}_t^k = y_t^k + d_{safe}$. Here, d_{safe} is the safe distance between vehicles, c_j are weights to trade off among risks, and $\eta > 0$ is a certain threshold to mark the expected zone or not. The proactive and passive agents/vehicles will learn the optimal policy $\pi^k(\cdot|s)$ that maximizes the total discounted reward to guide the lane change. Notice that the reward of each agent depends on the state (and thus action) of the other agent, which effectively leads to a strategic interaction among the agents and allows to study the cooperative lane change using the Markov game framework.

V. Lane change schemes

In the section, we consider two lane change schemes based on RL. In the first one (called asynchronous scheme), each vehicle treats the other vehicle as part of the environment and carries out RL as in the standard single-agent RL setting. Such a scheme often leads to vehicle collision, which motivates us to consider another scheme (called synchronous scheme) where vehicles carry out RL to solve for the resulting Markov game. This scheme takes into consideration the strategic behavior (being reward maximizer) of other vehicles, and leads to safe lane change.

A. Asynchronous scheme

Each vehicle considers all other vehicles as part of the environment, and carries out (single-agent) Deep Q-learning as in Algorithm 1. As each vehicle's action does not take into account the strategic behavior of the other vehicle, the asynchronous scheme may lead to vehicle collision. This is confirmed by the experiments; see Section VI.

B. Synchronous scheme

Motivated by [9][8], we use Markov game to model and guide the lane process.

Definition 1: A two-agent Markov game is defined as $\mathcal{G} := \{\mathcal{N}, S, (A^k)_{k\in\mathcal{N}}, P, (r^k)_{k\in\mathcal{N}}, (\pi^k)_{k\in\mathcal{N}}\}$, where $\mathcal{N} = \{1,2\}$, S and $(A^k)_{k\in\mathcal{N}}$ are the state space and action spaces,

Algorithm 1 Deep Q-learning algorithm [14]

Initialization of $Q(s,a;\theta)$ and the replay buffer $\mathcal{D}$.
for episode $1:M$ **do**
  Randomly choose a starting state s_0.
  for $t = 1, \cdots, T$ **do**
    Choose action $a_t = \arg\max_a Q(s_t,a;\theta)$ with probability $1-\epsilon$. Otherwise, make a random choice.
    Take action a_t in simulator and add (s_t,a_t,r_t,s_{t+1}) to the replay buffer.
    Take a sample (s,a,r,s') from $\mathcal{D}$.
    Update neural network by $\theta \leftarrow \theta - \alpha\nabla_\theta L(\theta)$ where $y = r + \gamma \max_{a'} Q(s',a';\theta)$ or $y = r$ if the process is terminated.
  end for
end for

$\pi^k = \{\pi_t^k\}_{t=1,\cdots T}$ is the strategy of agent k., $r^k(s,(a^1,a^2))$ is the reward function of agent k and $P(\cdot|s,(a^1,a^2))$ is the transition function.

At time t, agent k obseves state s_t, takes action $a_t^k \sim \pi_t^k(\cdot|s_t)$ according to its strategy π^k consisting of the policies π_t^k, and receives the reward $r_t^k(s_t,(a_t^1,a_t^2))$, while the state $s_{t+1} \sim P(\cdot|s_t,(a_t^1,a_t^2))$. In Markov game, the agents will adjust their strategies to reach an Nash equilibrium.

Definition 2: A Nash equilibrium of two-agent Markov game $\mathcal{G}$ is a pair of strategy (π^{1^*},π^{1^*}) such that $\forall s \in S$,

$$V^1(s|\pi^{1^*},\pi^{2^*}) \geq V^1(s|\pi^1,\pi^{2^*}), \forall \pi^1, \quad (6)$$
$$V^2(s|\pi^{1^*},\pi^{2^*}) \geq V^2(s|\pi^{1^*},\pi^2), \forall \pi^2, \quad (7)$$

where V^k is the state value function defined by equation (4).

We see that at a Nash equilibrium each agent's strategy is the best response to the other's strategy. We consider only stationary strategies, and the existence of Nash equilibrium is ensured by the following theorem.

Theorem 3: (Filar and Vrieze [34], Theorem 4.6.4) A Markov game with stationary strategy has at least one Nash equilibrium.

Further, consider the optimal action-state function

$$Q_\pi^{k^*}(s_t,(a_t^1,a_t^2)) = r_t^k(s_t,(a_t^1,a_t^2)) + \gamma \mathbf{E}_{a_{t+1}^1\sim\pi^{1^*},a_{t+1}^2\sim\pi^{2^*},s_{t+1}} Q_{\pi^*}^k(s_{t+1},(a_{t+1}^1,a_{t+1}^2)). \quad (8)$$

The agent k will take action based on $\pi^k(s_t)$ and $Q_\pi^k(s_t,(a_t^1,a_t^2))$ at each time t, which leads to a bimatrix game.

Definition 4: A bimatrix game is defined as $\{\mathcal{N},(M^k)_{k\in\mathcal{N}},(\pi^k)_{k\in\mathcal{N}}\}$, where the set of agents $\mathcal{N} = \{1,2\}$, payoff $(M^k)_{k\in\mathcal{N}}$ are $|A^1| \times |A^2|$ matrices with the (i,j)-th entry $M^k(i,j) = Q_{\pi^k}^k(s_t,(a_i^1,a_j^2))$, and π^k is the strategy of agent k. The mixed strategy Nash equilibrium of this bimatrix game is a pair of probabilities (μ^{1^*},μ^{2^*}) satisfying

$$\mu^{1^*} M^1 \mu^{2^*} \geq \mu^1 M^1 \mu^{2^*}, \forall \mu^1 \in PD(A^1), \quad (9)$$
$$\mu^{1^*} M^2 \mu^{2^*} \geq \mu^{1^*} M^2 \mu^2, \forall \mu^2 \in PD(A^2). \quad (10)$$

By modeling each decision making step as a bimatrix game, the whole Markov game consists of a sequence of bimatrix games. Theorem 3 of [9] ensures that the Nash equilibrium $(\pi^1(s_t), \pi^2(s_t))$ of the bimatrix game $(Q^1(s_t), Q^2(s_t))$ is also part of the Nash equilbrium of the whole Markov game. If $\pi^{k^*} = \{\pi^{k^*}(\bar{s}_\ell)\}_{\ell=1,\cdots,|S|}$ is the Nash equilibrium of the Markov game $\mathcal{G}$, then there exists one $(\pi^{1^*}(\bar{s}'), \pi^{2^*}(\bar{s}'))$ that is also the Nash equilibrium of the bimatrix game $(Q^1(s_t), Q^2(s_t))$ with $s_t = \bar{s}'$. The state function value $V^k(s|\pi^{1^*}, \pi^{2^*}) = \pi^{1^*}(s)Q^{k^*}(s)\pi^2(s)$ with the starting state s, and the action-state function value (8) is given by

$$Q_\pi^{k^*}(s_t, (a_t^1, a_t^2)) = r_t^k(s_t, (a_t^1, a_t^2)) + \gamma \mathbf{E}_{s_{t+1}}[V^k(s_{t+1}|\pi^{1^*}, \pi^{2^*})]. \quad (11)$$

After taking action $a_t^k \sim \pi^k(s_t)$, the updated payoff matrix will be the Nash equilibrium $(\pi^1(s_{t+1}), \pi^2(s_{t+1}))$ of the bimatrix game $(Q_t^1(s_{t+1}), Q_t^2(s_{t+1}))$ instead of independent update [9],

$$Q_{t+1}^k(s_t,(a^1,a^2)) = (1-\alpha_t)Q_t^k(s_t,(a^1,a^2)) + \alpha_t[r_{t+1}^1 + \gamma\pi^1(s_{t+1})Q_t^k(s_{t+1})\pi^2(s_{t+1})]. \quad (12)$$

Notice that the Q-values with different state-action pairs are difficult to calculate directly. Instead, we can use neural network as the estimator of the Q-table: $Q^k(s) \approx Q^k(s;\theta^k)$. This leads to the Algorithm 2.

Algorithm 2 Deep bimatrix Q-learning for vehicle k

1: Build neural networks $Q^k(\cdot;\theta^k)$ with output size $|A_1| \times |A_2|$ and replay buffer $\mathcal{D}$.
2: Build the initial $Q^{\mathcal{N}/k}(\cdot;\lambda)$ for the other vehicle.
3: **for** episode $1 : M$ **do**
4: Vehicle k chooses a starting state s_0^k randomly.
5: **for** $t = 1, \cdots, T$ **do**
6: Calculates $(\pi^1(s_t), \pi^2(s_t))$.
7: Two vehicles reach $(\pi^1(s_t), \pi^2(s_t))$ consensus.
8: Chooses action by $\pi^k(s_t)$ with probability $1-\epsilon$. Otherwise, vehicle k chooses action randomly.
9: Take actions in the simulator and add $(s_t, (a_t^1, a_t^2), r_t^k, s_{t+1})$ into replay buffer $\mathcal{D}$.
10: **while** One minibatch sample from $\mathcal{D}$ **do**
11: For each $(s, (a^1, a^2), r^k, s')$ in the minibatch, calculate target y^k as follows

$$y^k = \begin{cases} r^k + \gamma\pi^1(s')Q_t^k(s';\theta^k)\pi^2(s'), & \text{not terminated,} \\ r^k, & \text{terminated.} \end{cases}$$

12: Use stochastic gradient descent to update θ^k with the cost function $(y^k - Q^k(s,(a^1,a^2);\theta^k))^2$.
13: Update $Q_t^{\mathcal{N}/k}(s;\lambda)$.
14: **end while**
15: **end for**
16: **end for**

Here $(\pi^1(s_t), \pi^2(s_t))$ is the mixture Nash equilibrium of bimatrix game $(Q^k(s_t;\theta^k), Q^{\mathcal{N}/k}(s_t;\lambda))$. We apply Lemke-Howson algorithm to caculate the mixture Nash equilibrium of the bimatrix game. $Q^k(s_t;\theta^k)$ is the vehicle k's Q-values and $Q^{\mathcal{N}/k}(s_t;\lambda)$ is the estimate of the other vehicle's Q-values. $Q^{\mathcal{N}/k}(s_t;\lambda)$ can be build by neural network and the data (e.g., $r_t^{\mathcal{N}/k}$ or s_t or $Q^{\mathcal{N}/k}(s_t;\theta^{\mathcal{N}/k})$) sent by the other vehicle $\mathcal{N}/k$. Based on the received data, λ can be updated using the gradient method or replaced by the received value of $Q^{\mathcal{N}/k}(s_t;\theta^{\mathcal{N}/k})$ value. In our experiments, we use the received $Q^{\mathcal{N}/k}(s_t;\theta^{\mathcal{N}/k})$ from the other vehicle.

C. Convergence of Algorithm 2

Based on [9], the following theorem ensures the convergence of the algorithm 2.

Theorem 5: (Theorem 4 in [9]) Given a Markov game $\mathcal{G} := \{\mathcal{N}, S, (A^k)_{k\in\mathcal{N}}, P, (r^k)_{k\in\mathcal{N}}, (\pi^k)_{k\in\mathcal{N}}\}$ as in Definition 1. If we take the iterative update (12) through Algorithm 2, it will converge to the Nash equilibrium (π^{1^*}, π^{2^*}) of $\mathcal{G}$.

Proof: By the Conditional Averaging Lemma in [35], $Q_{t+1}^k(s_t,(a_t^1,a_t^2)) = (1-\alpha_t)Q_t^k(s_t,(a_t^1,a_t^2)) + \alpha_t[r_t^k + \gamma\pi^1(s_{t+1})Q_t^k(s_{t+1})\pi^2(s_{t+1})]$ will converge to

$$\begin{aligned} & \bar{T}^k Q^k(s_t,(a_t^1,a_t^2)) \\ = \; & \mathbf{E}_{s_{t+1}}[r_t^k + \gamma\pi^1(s_{t+1})Q_t^k(s_{t+1})\pi^2(s_{t+1})] \quad (13) \\ = \; & \sum_{s_{t+1}} P(s_{t+1}|s_t,(a_t^1,a_t^2))[r_t^k + \gamma\pi^1(s_{t+1})Q_t^k(s_{t+1})\pi^2(s_{t+1})] \\ = \; & \sum_{s_{t+1}} P(s_{t+1}|s_t,(a_t^1,a_t^2))T^k Q^k(s_t), \end{aligned}$$

where $T^k Q^k(s_t) = r_t^k + \gamma\pi^1(s_{t+1})Q_t^k(s_{t+1})\pi^2(s_{t+1})$. By Theorem 9 in the Appendix, T^k is a contraction mapping, and so is $\bar{T}^k$. So the update (12) will converge to a fixed point of $\bar{T}^k$. ∎

VI. Experiments and Evaluations

A. Testbed

In order to test this method we use a physics engine-based [36] high-fidelity model of a four wheel drive, $\frac{1}{8}^{th}$ scale vehicle. The dynamics of the system are given as:

$$\boldsymbol{x}'(t) = \boldsymbol{\Phi}(\boldsymbol{x}(t), \boldsymbol{F}(t), \boldsymbol{u}(t), \boldsymbol{p}), \quad (14)$$

in which $\boldsymbol{\Phi}(\cdot)$ is the dynamical model, $\boldsymbol{x}(t) \in \mathbb{R}^n$ is the state of the system, $\boldsymbol{x}'(t)$ is the derivative of the state, $\boldsymbol{F}(t)$ consolidates external forces and collisions applied to the system from e.g. the road surface, $\boldsymbol{u}(t) \in \mathbb{R}^m$ is a control input vector including steering and acceleration signals, and $\boldsymbol{p} \in \mathbb{R}^w$ is model parameter vector to be calibrated. In order to simulate simulated the behavior of a physical system, we calibrate model parameters $\boldsymbol{p}$ to the physical platform by using an online calibration algorithm [37]. By achieving low uncertainty on model parameters, it is observed that the simulated vehicle can exhibit virtually identical behavior as the actual vehicle.

By providing the actions based on learned policies $\pi^k(s_t)$ for multiple simulated vehicles and stepping the simulator forward, state updates can be obtained and update the observation s_{t+1}, which then propagate Q-value updates from Eq. (12) leading to direct policy updates via Algorithm 2.

B. Experiments

We implement the two schemes V-A and V-B using the lane change structure in Fig. 1. In this diagram, *car0* is the proactive vehicle trying to change to lane-3, *car2* is the passive vehicle creating merging space, *car1* is the reference vehicle, and *car3, car4, car5* are neighbors. The neural network settings of each agent are the same, using one hidden layer consisting of 512 units. The training machine consists of a 2.7GHz Intel Core i5 processor with 16GB of RAM. *car0* and *car2* are trained by randomly assigning initial configurations. After training, the tests call the learned policy for determining the controls to be applied to the individual vehicles.

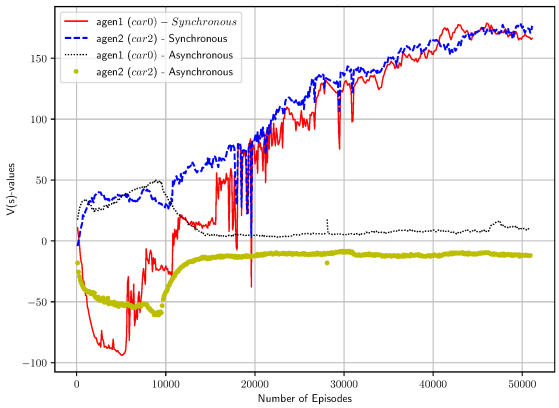
The performance of this method is shown in Fig. 2. In addition to the termination conditions like success and collision, there are additional training constraints. For instance, in our experiments, $|v_t^k|$ is bounded and each vehicle has a safety distance between its neighbors that must be upheld. $car2$ should not fall too far behind $car5$, otherwise it is presumed that traffic would be deleteriously affected. The merging moment configuration includes the safe relative speed between $car0/car2$ and $car1$, while $car2$ keeps a certain merging gap with $car1$.

Fig. 2a shows the state function values $(V^1(s), V^2(s))$ starting from the state s of the two schemes. During training, the random initialization for s was reused 5 times in each episode. For V-B, both $V^k(s)$ reached a steady state policy after ≈40,000 training episodes. For V-A, $V^k(s)$ reached steady-state in fewer training episodes than scheme V-B. Even so, both schemes eventually resulted in a stable policy $\{\pi^k\}_{k=1,2}$. After ≈20,000 episodes training, scheme V-B achieved higher $V^k(s)$ values than V-A, meaning that scheme V-B resulted in generally more optimal strategies.

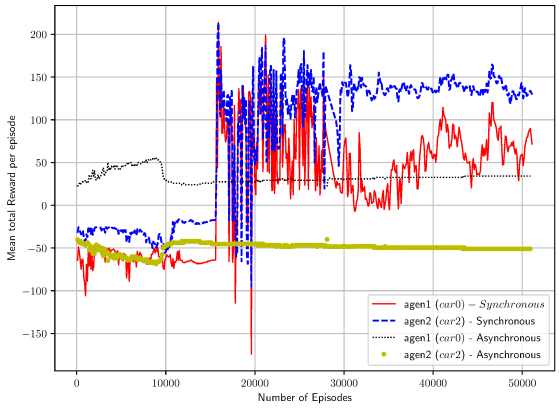
Fig. 2b shows the mean of total reward $\sum_{t=0}^{T} \gamma^t r_t$ per episode for the same initial state s. We can see that, after training, scheme V-A reaches stability faster than scheme V-B; however, after 40,000 training episodes, scheme V-B achieves higher total rewards, which suggests even stronger preferences for scheme V-B.

Both of Fig. 2a and 2b show an advantage of scheme V-B. One of the reasons is that each agent always tries to find the best strategy corresponding to the other one's strategy based on the Q-tables $\{Q^k(s;\theta^k)\}$. The adjustment of equilibrium solutions (π^1, π^2) will not stop until they reach the Nash equilibrium. Compared with scheme V-A, the efficient usage of the interaction in scheme V-B makes agents learn their policies better.

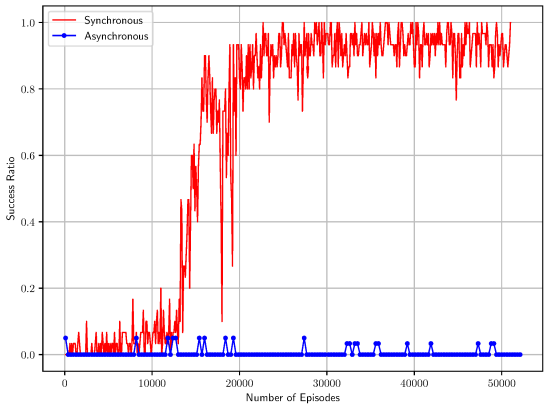
Fig. 2c shows results for the successful merging ratio, which is the ratio of the number of arrivals for expected merging moment over the total number of tests. In each episode, the tests were conducted 30 times with randomly selected initial states. From this, it is clear that scheme V-B's successful ratio increases after 12k episodes training. After that, this ratio begins to plateau after 20k episodes training. Meanwhile, most of the tasks are failed for scheme V-A. It is a more convincing proof for the advantages of scheme V-B.



(a) $\{V^k(s)\}_{k=1,2}$ values



(b) Mean total reward per episode



(c) Successful merging ratio

Fig. 2: Synchronous/Asynchronous scheme performance

Based on the above evaluation, scheme V-B appears to bring higher state function values and total rewards. More importantly, it brings bigger chance to find the proper merging moment. Videos demonstrating the performance among

different training episodes are provided (V-B scheme link and V-A scheme link). These results suggest that the synchronous cooperative scheme has potential to assist autonomous vehicles in finishing maneuvers strategically after sufficient training.

VII. CONCLUSIONS

In our paper, we have introduced the proactive-passive pair of vehicles with one reference vehicle as the lane change structure. Because the non-coordination scheme for lane change task does not give the expected performance, we propose a scheme considering Deep Q-learning and the Markov game together. The resulting performance on the physics engine based high fidelity model shows that the proposed scheme allows the vehicles reach the proper merging configuration starting from different states. This scheme could be used to build the model-free controller to help autonomous vehicles finish tasks after training. In the future, we are going to consider how to improve the training process so that it can avoid obvious dangerous behaviors.

APPENDIX

To prove Theorem 5, we have the following assumptions and result.

Assumption 6: The learning rates α_t satisfy $\sum_t \alpha_t = \infty$, $\sum_t \alpha_t^2 < \infty$.

Assumption 7: (Assumption 1 in [9]) Given a bimatrix game $(Q^1(s), Q^2(s))$, its mixture Nash equilibrium is a pair of probabilities (μ^{1^*}, μ^{2^*}) satisfies one of the following properties:

1) Property1: The equilibrium is social optimal:
$$\mu^{1^*} Q^k \mu^{2^*} \geq \mu^1 Q^k \mu^2, \forall \mu^1 \in PD(A^1), \mu^2 \in PD(A^2).$$
2) Property2: The equilibrium is not social optimal:
$$\mu^{1^*} Q^1 \mu^{2^*} \leq \mu^{1^*} Q^1 \mu^2, \forall \mu^2 \in PD(A^2),$$
$$\mu^{1^*} Q^2 \mu^{2^*} \leq \mu^1 Q^2 \mu^2, \forall \mu^1 \in PD(A^1).$$

Assumption 8: (Assumption 2 in [38]) For all bimatrix game $(Q_t^1(s), Q_t^2(s))$ in the Markov game $\mathcal{G}$ at time t, they should all satisfy Property 1 in Assumption 7 or all satisfy Property 2 in Assumption 7.

With these assumptions, we can prove that the Q value update operation is a contraction mapping,

Theorem 9: (Lemma 3 in [9]) Define a mapping T^k such that $T^k Q^k(s) = r + \gamma \pi^1(s) Q^k(s) \pi^2(s)$ where $(\pi^1(s), \pi^2(s))$ is the Nash equilibrium of the bimatrix game $(Q^1(s), Q^2(s))$. Then T^k is a contraction mapping.

Proof: Without loss of generality, consider two bimatrix games $(Q^1(s), Q^2(s))$ and $(Q^{1'}(s), Q^{2'}(s))$ with $T^k Q^k(s) \geq T^k Q^{k'}(s)$, and denote the corresponding mixture Nash equilibriums by $(\pi^1(s), \pi^2(s))$ and $(\pi^{1'}(s), \pi^{2'}(s))$ respectively. For agent k,

$$\begin{aligned} &T^k Q^k(s) - T^k Q^{k'}(s) \\ =&(r + \gamma \pi^1(s) Q^k(s) \pi^2(s)) - (r + \gamma \pi^{1'}(s) Q^{k'}(s) \pi^{2'}(s)) \\ =&\gamma(\pi^1(s) Q^k(s) \pi^2(s) - \pi^{1'}(s) Q^{k'}(s) \pi^{2'}(s)) \end{aligned}$$

By Assumption 8, we have two cases:

1) If $(\pi^1(s), \pi^2(s))$ and $(\pi^{1'}(s), \pi^{2'}(s))$ are social optimal: we have $\pi^{1'}(s) Q^{k'}(s) \pi^{2'}(s) \geq \pi^1(s) Q^{k'}(s) \pi^2(s)$, so that

$$\begin{aligned} &T^k Q^k(s) - T^k Q^{k'}(s) \\ \leq&\gamma(\pi^1(s) Q^k(s) \pi^2(s) - \pi^1(s) Q^{k'}(s) \pi^2(s)) \\ \leq&\gamma \|\pi^1(s)\| \|Q^{k'}(s) - Q^{k'}(s)\| \|\pi^2(s)\| \end{aligned}$$

Notice that $\gamma \in (0,1)$ and $\|\pi^{1'}(s)\| \in [0,1], \|\pi^{2'}(s)\| \in [0,1]$. So T^k is a contraction mapping.

2) If $(\pi^1(s), \pi^2(s))$ and $(\pi^{1'}(s), \pi^{2'}(s))$ are not social optimal: We consider different agents separately:

- For agent-1. As $(\pi^{1'}(s), \pi^{2'}(s))$ is a Nash equilibrium, we have $\pi^{1'}(s) Q^{1'}(s) \pi^{2'}(s) \geq \pi^1(s) Q^{1'}(s) \pi^{2'}(s)$. Thus,
$$\begin{aligned} &T^1 Q^1(s) - T^1 Q^{1'}(s) \\ \leq&\gamma(\pi^1(s) Q^1(s) \pi^2(s) - \pi^1(s) Q^{1'}(s) \pi^{2'}(s)). \end{aligned}$$
Since $(\pi^1(s), \pi^2(s))$ is not social optimal, agent-1 will receive higher payoff for any other policies of the agent-2, which means
$$\pi^1(s) Q^1(s) \pi^2(s) \leq \pi^1(s) Q^1(s) \pi^{2'}(s).$$
Thus,
$$\begin{aligned} &T^1 Q^1(s) - T^1 Q^{1'}(s) \\ \leq&\gamma(\pi^1(s) Q^1(s) \pi^2(s) - \pi^1(s) Q^{1'}(s) \pi^{2'}(s)) \\ \leq&\gamma(\pi^1(s) Q^1(s) \pi^{2'}(s) - \pi^1(s) Q^{1'}(s) \pi^{2'}(s)) \\ \leq&\gamma \|\pi^1(s)\| \|Q^{k'}(s) - Q^{k'}(s)\| \|\pi^{2'}(s)\|. \end{aligned}$$
- For agent-2. Similarly, We have
$$\begin{aligned} &T^2 Q^2(s) - T^2 Q^{2'}(s) \\ \leq&\gamma(\pi^1(s) Q^k(s) \pi^2(s) - \pi^{1'}(s) Q^{2'}(s) \pi^2(s)) \\ \leq&\gamma(\pi^{1'}(s) Q^2(s) \pi^2(s) - \pi^{1'}(s) Q^{2'}(s) \pi^2(s)) \\ \leq&\gamma \|\pi^{1'}(s)\| \|Q^{k'}(s) - Q^{k'}(s)\| \|\pi^2(s)\|. \end{aligned}$$

We see that T^k is a contraction mapping. ■

REFERENCES

[1] G.-L. Chang and Y.-M. Kao, "An empirical investigation of macroscopic lane-changing characteristics on uncongested multilane freeways," *Transportation Research Part A: General*, vol. 25, no. 6, pp. 375–389, 1991.

[2] J. A. Laval and C. F. Daganzo, "Lane-changing in traffic streams," *Transportation Research Part B: Methodological*, vol. 40, no. 3, pp. 251–264, 2006.

[3] W. Knospe, L. Santen, A. Schadschneider, and M. Schreckenberg, "A realistic two-lane traffic model for highway traffic," *Journal of Physics A: Mathematical and General*, vol. 35, no. 15, p. 3369, 2002.

[4] P. Hidas, "Modelling lane changing and merging in microscopic traffic simulation," *Transportation Research Part C: Emerging Technologies*, vol. 10, no. 5, pp. 351–371, 2002.

[5] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[6] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.

[7] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems," *The Knowledge Engineering Review*, vol. 27, no. 1, pp. 1–31, 2012.

[8] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of the eleventh international conference on machine learning*, vol. 157, 1994, pp. 157–163.

[9] J. Hu, M. P. Wellman *et al.*, "Multiagent reinforcement learning: theoretical framework and an algorithm." in *ICML*, vol. 98, 1998, pp. 242–250.

[10] S. Singh, D. Litman, M. Kearns, and M. Walker, "Optimizing dialogue management with reinforcement learning: Experiments with the njfun system," *Journal of Artificial Intelligence Research*, vol. 16, pp. 105–133, 2002.

[11] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 3. IEEE, 2004, pp. 2619–2624.

[12] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "Autonomous inverted helicopter flight via reinforcement learning," in *Experimental Robotics IX*. Springer, 2006, pp. 363–372.

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[15] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[16] K. Li and J. Malik, "Learning to optimize," *arXiv preprint arXiv:1606.01885*, 2016.

[17] ——, "Learning to optimize neural nets," *arXiv preprint arXiv:1703.00441*, 2017.

[18] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio, "An actor-critic algorithm for sequence prediction," *arXiv preprint arXiv:1607.07086*, 2016.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[20] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[21] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[22] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PloS one*, vol. 12, no. 4, p. e0172395, 2017.

[23] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," *arXiv preprint arXiv:1705.08926*, 2017.

[24] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *arXiv preprint arXiv:1706.02275*, 2017.

[25] N. Usunier, G. Synnaeve, Z. Lin, and S. Chintala, "Episodic exploration for deep deterministic policies: An application to starcraft micromanagement tasks," *arXiv preprint arXiv:1609.02993*, 2016.

[26] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multiagent control using deep reinforcement learning," in *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2017)*, 2017.

[27] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," *arXiv preprint arXiv:1703.04908*, 2017.

[28] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, J. Perolat, D. Silver, T. Graepel *et al.*, "A unified game-theoretic approach to multiagent reinforcement learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4193–4206.

[29] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[30] J. Schulman, "Optimizing expectations: From deep reinforcement learning to stochastic computation graphs," 2016.

[31] X.-G. Li, B. Jia, Z.-Y. Gao, and R. Jiang, "A realistic two-lane cellular automata traffic model considering aggressive lane-changing behavior of fast vehicle," *Physica A: Statistical Mechanics and its Applications*, vol. 367, pp. 479–486, 2006.

[32] T. Toledo, H. Koutsopoulos, and M. Ben-Akiva, "Modeling integrated lane-changing behavior," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1857, pp. 30–38, 2003.

[33] H. Bi, T. Mao, Z. Wang, and Z. Deng, "A data-driven model for lane-changing in traffic simulation." in *Symposium on Computer Animation*, 2016, pp. 149–158.

[34] J. Filar and K. Vrieze, *Competitive Markov decision processes*. Springer Science & Business Media, 2012.

[35] C. Szepesvári and M. L. Littman, "A unified analysis of value-function-based reinforcement-learning algorithms," *Neural computation*, vol. 11, no. 8, pp. 2017–2060, 1999.

[36] E. Coumans, "Bullet physics engine," *Open Source Software: http://bulletphysics. org*, vol. 1, 2010.

[37] S. Aghli and C. Heckman, "Online system identification and calibration of dynamic models for autonomous ground vehicles," *IEEE International Conference on Robotics and Automation (to appear)*, 2018.

[38] M. Bowling, "Convergence problems of general-sum multiagent reinforcement learning," in *ICML*. Citeseer, 2000, pp. 89–94.