Learning to Simulate Self-driven Particles System with Coordinated Policy Optimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

Self-driven Particles (SDP) models a category of multi-agent systems that commonly exist in everyday life, such as flocking birds and traffic of vehicles. In SDP model, each agent pursues its own goal and constantly changes its cooperative or competitive behaviors with its nearby agents. It makes simulating SDP a longstanding challenge. While manually-designed controllers have been able to emulate complex behaviors of SDP, building such controllers is time-consuming and the emergent behaviors are often neither realistic nor resilient. Reinforcement learning provides an appealing alternative for automating the development of the controller for SDP. However, previous multi-agent reinforcement learning (MARL) methods define the agents to be teammates or enemies before hand, which fail to capture the essence of SDP where the agents vary to be cooperative or competitive even within one episode. To simulate SDP with MARL, a key challenge is to coordinate agents' behaviors while still maximizing individual objectives. Taking traffic simulation as the testing bed, in this work we develop a novel MARL method called Coordinated Policy Optimization (CoPO), which incorporates social psychology principle to learn neural controller for SDP. Experiments show that the proposed method can achieve superior performance compared to MARL baselines in various metrics. Noticeably the trained vehicles exhibit complex and diverse social behaviors that improve performance and safety of the population as a whole.

1 Introduction

Self-driven Particles (SDP) models a wide range of multi-agent systems (MAS) in nature and human society. In SDP, individual agent pursues its own goal and interacts with each other following simple local alignment, and then the population exhibits complex collective behaviors [45]. We commonly see the phenomena of collective behaviors, such as the flocking birds [4], molecular motors [6], human crowd [17, 16], and the traffic system [20]. To understand and simulate such phenomena, researchers have developed a number of SDP models. For example, simple-rule based models [7] or Hydrodynamic equations based models [2, 19] can simulate the SDP very well in an unconstrained environment with random movement and resemble complex behaviors such as schooling fish [12] and marching locusts [5]. However, in a more structured environment such as a particular traffic scene where the interactions of agents are time-varying and the environment is non-stationary, it is difficult to design manual controllers or use rules to recover the underlying collective behaviors.

When the interactive environment is available, reinforcement learning becomes a promising approach to learn the controllers or behavior models for actuating the SDP. Recently, many multi-agent reinforcement learning (MARL) methods have been developed to play competitive multi-player games, such as Hide and Seek [1], Football [23], Go and other board games [33], and StarCraft [32]. However, it is challenging to apply the existing MARL to simulate SDP systems. That's because

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

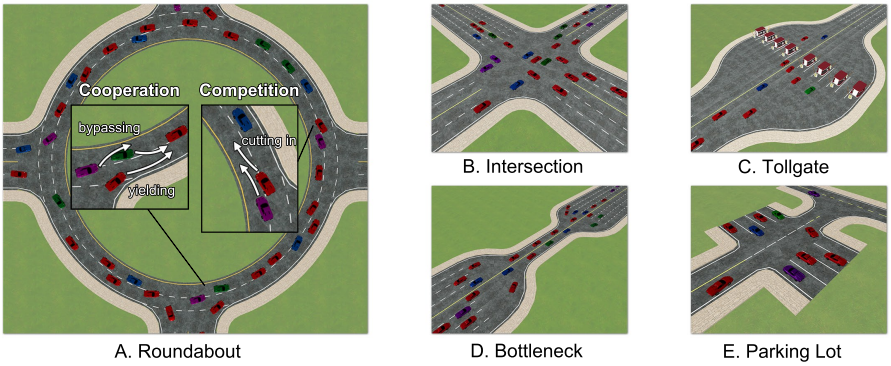


Figure 1: The five environments used in our evaluation. **A** highlights two cooperative and competitive events in the vehicle interactions.

each constituent agent in a SDP system is self-interested and the relationship between agents is constantly changing. As illustrated in Fig. 1**A**, the vehicles in the traffic system demonstrate social behaviors that are either cooperative or competitive depending on the situations. Meanwhile, the cooperation and the competition naturally emerge as the outcomes of self-driven instincts and the multi-agent interaction. Thus it is difficult to generate those social behaviors through a top-down design. Furthermore, most of the MARL methods assume that the role of each agent is pre-determined and fixed. In cooperative MARL methods [31, 39, 9, 8], all agents need to cooperate to maximize a joint scalar reward. However, the credit decomposition problem centered in such setting is not the major issue of simulating SDP systems, because each agent in SDP has its individual objective and the credit decomposition is no longer needed. On the other hand, directly applying independent learning [41, 34] will lead to a group of agents with unreasonable behaviors that are aggressive or egocentric, since each agent is trained to maximize its own reward.

In order to simulate SDP systems, the coordination is the major issue. The coordination of agents in MARL has been previously studied in the context of mixed cooperative and competitive tasks [25, 36]. However, seldom works focus on the simulation of SDP systems. Take the microscopic traffic simulation as an example, existing works focus on learning high-level controller to indirectly actuate vehicles [46], or using independent learner to train policies executing in the environments engaging a limited set of agents on simple scenes [30, 54], without considering the coordination problem. Instead, we aim to control all the vehicles in the scene by operating on the low-level continuous control space. This allows a higher degree of freedom to learn diverse behaviors but poses a challenging continuous control problem. In this work we propose a novel MARL method called **Co**ordinated **P**olicy **O**ptimization (CoPO) to facilitate the coordination of agents at both local and global levels. To evaluate the proposed method, as illustrated in Fig. 1, we construct five typical traffic environments as the testing bed. These environments contain rich interactions between agents and complex road structures. Besides, we develop three task-agnostic metrics to characterize different aspects of the learned population. We show that the proposed CoPO method can achieve superior performance compared to the baselines. Noticeably, collective social behaviors that resemble real-world patterns emerge in the trained population.

2 Related Work

Multi-Agent Reinforcement Learning. Three typical task settings are explored in MARL [18]: the fully cooperative task, the fully competitive task, and the mixed cooperative and competitive task. Many works focus on the problems in cooperative tasks such as credit assignment [31, 39, 9, 8, 53] and communication [10, 44]. In competitive tasks, the challenges include modeling the opponents [11] and generating meaningful opponent via self-play [15, 47, 1]. In mixed cooperative and competitive task [28, 25, 36], the roles of the agents are mostly predefined and fixed at run-time. Instead, in this work we hope to simulate SDP systems without imposing any constraint on the roles of the agents thus there is more space for emergent social behaviors. Apart from the fixed role problem, previous methods like MADDPG [28], QMIX [31], QTRAN [38], and COMA [9] are only applicable to discrete action space. We instead focus on learning policies for continuous control.

The independent PPO [34] and the Mean Field MARL [50] are two recent methods applied for continuous control in MARL. The Mean Field MARL method proposed by Yang *et al.* [50] approximates the value of global reward $Q(s, \mathbf{a})$, $\mathbf{a} = [a_1, ..., a_N]$ by agent-independent value functions $Q_i(s, a_i, \bar{a}_i^{\mathrm{N}})$, where $\bar{a}_i^{\mathrm{N}}$ is the average action of the nearby agents around agent i. Following similar design but in the context of simulating SDP, we use the mean field to compute neighborhood reward and use that reward to enforce local coordination, instead of using mean field to factorize joint value via averaging actions done in [50].

Traffic simulation. Simulating the macroscopic and microscopic traffic systems has been studied for decades [21]. As for the macroscopic traffic system where the dynamics of individuals is simple and the high-level characters of flow such as the traffic throughput are more concerned, early works focus on designing controllers to direct a fleet of vehicles into a dense and stable platoon [37, 3]. CityFlow [51] and FLOW [49] use RL agents to steer the low-level controllers of vehicles in order to investigate the traffic phenomenon in large-scale macroscopic traffic flow. For microscopic system, researchers also use behavioral model [43] or hand-designed control laws [22, 48] to control individual vehicles. Zhou *et al.* [54] propose SMARTS simulator to investigate the interaction of RL agents and social vehicles in atomic traffic scenes. Pal *et al.* [30] find the emergent road rules of MARL agents trained by simple independent learner, which are also shown in our experiments. In this work, we work on the traffic environments with complex real-world scenarios such as tollgate and parking lot. The traffic flow can contain more than 40 vehicles. In contrast to previous work, we also propose novel method to improve the collective motions that better simulate SDP systems.

3 Preliminaries

3.1 Problem Setting

SDP system has two distinguishable properties: (1) the individual agent can vary to be cooperative or competitive with others within one episode, (2) the individual agent only actively interacts with its nearby agents.

To accommodate these properties, we formulate a SDP system as a Dec-POMDP [14] with a tuple $G =< \mathcal{E}, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rho_0, \mathcal{O}, \mathcal{Z}, \gamma >$. $\mathcal{E}$ is the set of the agent indices. We define the active agent list at each environmental time step t as $\mathcal{E}_t = \{i_{1,t}, ..., i_{k_t,t}\} \subset \mathcal{E}$ wherein k_t is the number of active agents at this step. We consider a partially observable setting in which each agent can not access the environmental state s_t and instead draws individual observation $o_{i,t}$ according to the local observation function $o_{i,t} = \mathcal{Z}_i(s_t) : \mathcal{S} \to \mathcal{O}$. The joint action space $\mathcal{A}_t \subset \mathcal{A}$ is composed of the union of all the active agents' action spaces $\bigcup_{i=1}^{k_t} \mathcal{A}_{i,t}$. Each agent i chooses its action $a_{i,t} \in \mathcal{A}_{i,t}$ following its policy $a_{i,t} \sim \pi_i(\cdot|o_t)$. These actions form a joint action $\mathbf{a}_t = \{a_{i,t}\}_{i=1}^{k_t} \in \mathcal{A}_t$ causing a transition on the environmental state following the state transition distribution $\mathcal{P}(s_{t+1}|s_t, \mathbf{a}_t)$. Each agent receives their *individual reward* from its reward function $r_{i,t} = \mathcal{R}_i(s_t, \mathbf{a}_t)$. ρ_0 is the initial state distribution and γ is the discount factor. So in test time our system is decentralized, where each agent is fed with its local observation and operated by its policy for decision making.

Here we define the *environmental episode* $\tau = \{(s_t, \mathbf{a}_t, r_{1,t}, ..., r_{k_t,t})\}_{t=0}^{T}$, where T is the environmental horizon. An environmental episode thus contains a set of *agent episodes*: $\{(o_{i,t}, a_{i,t}, r_{i,t})\}_{t=t_i^s}^{t_i^e}$, wherein t_i^s and t_i^e denote the environmental steps when agent i enters and terminates, respectively. Suppose the policy π_i is parameterized by θ_i, the objective for each agent can be defined as the discounted sum of individual reward (namely the individual return): $J_i^I(\theta_i) = \mathbb{E}_\tau[R_i(\tau)]$, wherein $R_i(\tau) = \sum_{t=t_i^s}^{t_i^e} \gamma^{t-t_i^s} r_{i,t}$.

3.2 Individual Policy Optimization

A simple approach for MARL is the independent policy optimization (IPO) [34] which maximizes individual objective as if in the single-agent environment. The policy gradient methods are the appropriate solution, since it updates the policy based on the latest data, resulting in the fast adaptation to the variation of other agents. Assuming the individual value function as $V_{i,t}^I = V_i^I(s_t) = \mathbb{E}[\sum_{t'=t}^{t_i^e} \gamma^{(t'-t)} r_{i,t}]$ and the advantage function as $A_{i,t}^I = A_i^I(s_t, a_{i,t}, \mathbf{a}_{\neg i,t}) = r_{i,t} + \gamma V_i^I(s_{t+1}) - V_i^I(s_t)$, wherein $\mathbf{a}_{\neg i,t} = \{a_i\}_{j=1, j\neq i}^{k_t}$ is the shorthand of other agents' actions, the policy gradient

becomes [40]:

$$\nabla_{\theta_i} J_i^I(\theta_i) = \mathop{\mathbb{E}}_{(s,\mathbf{a})} [\nabla_{\theta_i} \log \pi_{\theta_i}(a_i|s) A_i^I(s, a_i, \mathbf{a}_{\neg i})]. \tag{1}$$

As a common practice, in PPO [35] the clipped importance sampling factor $\rho = \pi_{i,\text{new}}(a_i|s)/\pi_{i,\text{old}}(a_i|s)$ is used to mitigate the distribution shift when the policy is updated for several epochs, wherein $\pi_{i,\text{old}}$ is the behavior policy that generate samples and $\pi_{i,\text{new}}$ is the latest policy parameterized by θ_i. We call the resulting objective as the surrogate objective:

$$\hat{J}_i^I(\theta_i) = \mathop{\mathbb{E}}_{(s,\mathbf{a})} \min(\rho A_i^I, \text{clip}(\rho, 1-\epsilon, 1+\epsilon) A_i^I). \tag{2}$$

ϵ is hyper-parameter. By conducting the stochastic gradient ascent on the surrogate objective w.r.t. the policy parameters, the expected individual return can be improved. In continuous control RL problem, both the state space and action space are high-dimensional. Therefore V_i^I is approximated using a neural network whose input is typically the local observation, though in the centralized training and decentralized execution (CTDE) framework [28], the value function takes global information as input, which concatenates all agents' local observations.

4 Coordinated Policy Optimization

In a SDP system each agent has its individual objective. However, if we simply maximize each individual reward, the system will have sub-optimal solutions where, for example, the agents will become aggressive and egocentric, jeopardizing the performance of the population and leading to critical failures. On the contrary, if we apply cooperative learning schemes [31, 39] and consider the summation of individual reward as the joint objective, the trained agents will exhibit unreasonable behaviors such as sacrificing oneself to improve group reward, which is not expected in SDP systems. To find a balance, as illustrated in Fig. 2 and Algorithm 1, we develop a novel MARL algorithm called **Co**ordinated **P**olicy **O**ptimization (**CoPO**) to facilitate the bi-level coordination of agents to learn the controllers of the SDP systems. Following the CTDE framework, during centralized training, we first propose individual learning objectives by local coordination (Sec. 4.1), a mechanism inspired by the fact that each individual agent is mostly affected by its nearby agents [50]. We coordinate agents' objectives in neighborhood by mixing rewards following a common social psychological principle. We further design the meta-learning technique to optimize the local coordination process, leading to the global coordination (Sec. 4.2).

4.1 Local Coordination

IPO only maximizes the individual objective, leading to selfish policies that damage the overall performance of the population. However, animal groups often demonstrate a certain level of cooperative behavior that not only benefits others, but also results better individual utility. Thus we incorporate the ring measure of social value orientation [27, 36], a common metric from social psychology, into our formulation of local coordination. Concretely, we define the *Local Coordination Factor* (LCF) as a degree $\phi \in [-90^\circ, 90^\circ]$ describing an agent's preference of being selfishness, cooperative, or competitive.[1] We then define the *neighborhood reward*:

$$r_{i,t}^N = \frac{\sum_{j \in \mathcal{N}_{d_n}(i,t)} r_{j,t}}{|\mathcal{N}_{d_n}(i,t)|}, \text{ wherein } \mathcal{N}_{d_n}(i,t) = \{j : ||\text{Pos}(i) - \text{Pos}(j)|| \leq d_n\}. \tag{3}$$

$\mathcal{N}_{d_n}(i,t)$ defines the neighborhood of agent i within the radius d_n at step t. Note that the neighborhood of agent is time-varying, therefore the the neighborhood reward can not be simply computed from a fixed set of agents across the episode. We define the *coordinated reward* weighted by LCF as:

$$r_{i,t}^C = \cos(\phi) r_{i,t} + \sin(\phi) r_{i,t}^N. \tag{4}$$

The LCF allows the local coordination in the optimization process. Suppose the agents maximize the coordinated reward with $\phi = 0^\circ, 90^\circ, -90^\circ$ respectively, then the agents should become egoistic, altruistic, or sadistic respectively. Illustratively, Fig. 3 demonstrates an interesting preliminary result

[1] The social value orientation can be in range $[-180^\circ, 180^\circ]$. We only take the meaningful half of the range.

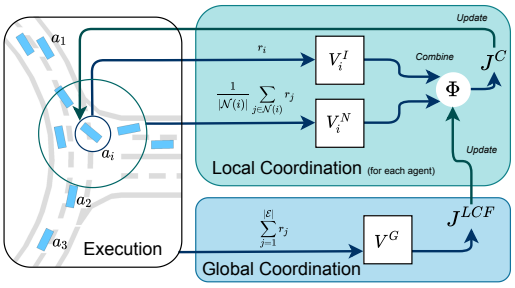


Figure 2: The framework of the CoPO method.

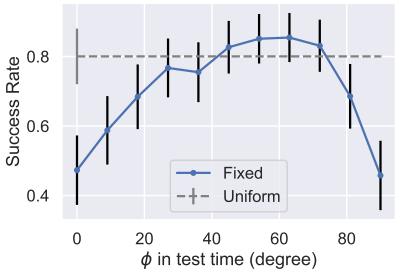


Figure 3: The test-time performance of a population trained with uniformly sampled LCF.

on a simple version of Roundabout environment. We train all policies $\pi_i(\cdot|o_{i,t}, \phi), i = 1, ...$ with an uniformly sampled $\phi \sim \mathcal{U}(0°, 90°)$ for each episode and test population controlled by the policies with shared and fixed LCF. The trained population shows a peak of success rate at certain LCF at test time. This result suggests the IPO might lead to sub-optimal policies since the individual objectives never consider local coordination with other agents.

In order to incorporate the LCF into training process to improve the population performance, apart from the value function approximator for the individual reward (Sec. 3.2), we use another value function to approximate the discounted sum of the neighborhood reward $V_{i,t}^N = \mathbb{E}[\sum_t^{t_i^e} \gamma^{t-t_i^s} r_{i,t}^N]$ and the neighborhood advantage is therefore calculated as: $A_{i,t}^N = r_{i,t}^N + \gamma V_{i,t+1}^N - V_{i,t}^N$. The coordinated advantage can be derived as: $A_{\Phi,i,t}^C = \cos(\phi) A_{i,t}^I + \sin(\phi) A_{i,t}^N$. We now have the *coordinated objective*, which takes the utilities of neighbors into consideration, as follows:

$$J_i^C(\theta_i, \Phi) = \mathop{\mathbb{E}}_{(s,\mathbf{a})}[\min(\rho A_{\Phi,i}^C, \text{clip}(\rho, 1-\epsilon, 1+\epsilon) A_{\Phi,i}^C)]. \tag{5}$$

We follow the similar training procedure as IPO introduced in Sec. 3.2. The objectives of the agent itself as well as its neighbors can be improved consistently. To improve diversity, we use distributional LCF $\phi \sim \mathcal{N}(\phi_\mu, \phi_\sigma^2)$, where $\Phi = [\phi_\mu, \phi_\sigma]^\top$ are the parameters of LCF distribution.

4.2 Global Coordination

The major challenge of local coordination is the selection of LCF. In this section, we introduce how we enable global coordination with bi-level optimization method that computes a meta-gradient of the global objective w.r.t. Φ. We first define the objective that indicates the performance of a population:

$$J^G(\theta_1, \theta_2, ...) = \mathop{\mathbb{E}}_{\tau}[\sum_{i \in \mathcal{E}_\tau} \sum_{t=t_i^s}^{t_i^e} r_{i,t}] = \mathop{\mathbb{E}}_{\tau}[\sum_{t=0}^{T} \sum_{i \in \mathcal{E}_{\tau,t}} r_{i,t}], \tag{6}$$

wherein $\mathcal{E}_{\tau,t}$ denotes the occurred agents at environmental time step t of the episode τ. Here we remove γ since it holds ambiguous meaning when the start steps of agents are not aligned. However, Eq. 6 can not be directly optimized when computing policy gradients of individual policies' parameters. To overcome such problem, we factorize $J^G(\theta_1, ...)$ to individual global objectives $J_i^G(\theta_i)$ such that maximizing the objective for each agent i is equivalent to maximizing Eq. 6. The *individual global objective* is defined as:

$$J_i^G(\theta_i) = \mathop{\mathbb{E}}_{\tau}[\sum_{t=t_i^s}^{t_i^e} \frac{\sum_{j \in \mathcal{E}_{\tau,t}} r_{j,t}}{|\mathcal{E}_{\tau,t}|}]. \tag{7}$$

Proposition 1 (Global objective factorization)**.** *Suppose each agent $i \in \mathcal{E}$ can maximize its individual global objective J_i^G, then the original global objective J^G is maximized.*

We prove such factorization does not affect the optimality of global objective in Appendix. To improve global objective via optimizing LCF, we need to compute the gradient of Eq. 7 w.r.t. Φ. Denote the parameters of policies before and after optimizing Eq. 5 as θ_i^{old} and θ_i^{new} respectively, we

Algorithm 1: The Procedure of CoPO

Input: Maximal iterations N; Environmental horizon T; Number of training epochs in each iteration K_p, K_Φ for updating policy and Φ, respectively; Number of mini batches in each epoch M.

Initialize environment, data buffer D, policy network π_θ, individual value network V_η^I, neighborhood value network V_ψ^N, global value network V_ω^G, LCF distribution $\mathcal{N}(\phi_\mu, \phi_\sigma^2)$.
for $I = 1, ..., N$ **do**
  Clear buffer D and store current policy $\pi_{\text{old}} \leftarrow \pi_\theta$.
  for $t = 0, ..., T$ **do**
    Assign each newly spawned agent i with $\phi_i \sim \mathcal{N}(\phi_\mu, \phi_\sigma^2)$.
    $\mathbf{a}_t = [a_{1,t}, ..., a_{k_t,t}], a_{i,t} \sim \pi_\theta(\cdot|o_{i,t}), \forall i = 1, ..., k_t$.
    Step the environment with $\mathbf{a}_t$ and store the tuple $(s_t, \mathbf{a}_t, r_{1,t}, ..., o_{1,t}, ...)$ to D.
  for $k = 1, ..., K_p$ **do**
    Shuffle D and slice it into M batches.
    Update π_θ, V_η^I, V_ψ^N and V_ω^G following Eq. 5 for each mini batch.
  for $k = 1, ..., K_\Phi$ **do**
    Update Φ following Eq. 11 with D.

can compute the gradient as:

$$\nabla_\Phi J_i^G(\theta_i^{\text{new}}) = \underbrace{\nabla_{\theta_i^{\text{new}}} J_i^G(\theta_i^{\text{new}})}_{\text{1st Term}} \underbrace{\nabla_\Phi \theta_i^{\text{new}}}_{\text{2nd Term}}. \tag{8}$$

The 1st term resembles the policy gradient but replaces objective with J_i^G:

$$\text{1st Term} = \mathop{\mathbb{E}}_{(s,\mathbf{a})\sim\theta_i^{\text{old}}}[\nabla_{\theta_i^{\text{new}}} \min(\rho A^G(s,\mathbf{a}), \text{clip}(\rho, 1-\epsilon, 1+\epsilon)A^G(s,\mathbf{a}))]. \tag{9}$$

We use an extra global value function V^G to estimate the value of global reward, then compute the global advantage A^G following similar equation of neighborhood advantage. The samples $(s, \mathbf{a})$ are generated by the behavior policy θ_i^{old}. By following first-order Taylor expansion on θ_i^{old}, the 2nd term can be computed as follows. Note that $\nabla_{\theta_i^{\text{old}}} J_i^C(\theta_i^{\text{old}}, \Phi)$ recovers the vanilla policy gradient in Eq. 1.

$$\text{2nd Term} = \nabla_\Phi(\theta_i^{\text{old}} + \alpha \nabla_{\theta_i^{\text{old}}} J_i^C(\theta_i^{\text{old}}, \Phi)) = \alpha \mathop{\mathbb{E}}_{(s,\mathbf{a})\sim\theta_i^{\text{old}}}[\nabla_{\theta_i^{\text{old}}} \log \pi_{\theta_i^{\text{old}}}(a_i|s) \nabla_\Phi A_{\Phi,i}^C]. \tag{10}$$

Averaging over all agents, we can combine Eq. 9 and Eq. 10 and derive the LCF objective as:

$$J^{LCF}(\Phi) = \mathop{\mathbb{E}}_{i} \mathop{\mathbb{E}}_{(s,\mathbf{a})\sim\theta^{\text{old}_i}} \{[\nabla_{\theta_i^{\text{new}}} \min(\rho A^G, \text{clip}(\rho, 1-\epsilon, 1+\epsilon)A^G)][\nabla_{\theta_i^{\text{old}}} \log \pi_{\theta_i^{\text{old}}}(a_i|s)] A_{\Phi,i}^C\}. \tag{11}$$

We then conduct stochastic gradient ascent on the LCF objective w.r.t. Φ with learning rate α. Algorithm 1 describes the overall procedure of CoPO.

In our implementation, we follow the commonly used parameter sharing technique [42] so that all agents share the same set of neural networks. The scalability of our method is thus guaranteed. As a summary, our method employs totally 4 neural networks: the policy network, the individual value network, the neighborhood value network, and the global value network. The input to all networks is always the local observation of the agent. Our further study (see Appendix) shows that centralized critic method does not yield better performance and usually causes unstable training. The reason might be the time-varying input to the centralized critics, that the combined observation of nearby agents is always changing and doesn't hold a temporal consistency across the episodes.

5 Experiments

5.1 Evaluation Platform

We develop a set of physics-based 3D traffic environments to evaluate MARL methods for simulating SDP systems. The environments are developed in Panda3D [13] and Bullet Engine, which can run 80

fps on a standard workstation[2]. Examples of the environments are shown in Fig. 1. The descriptions and typical settings of the five traffic environments are as follows:

Roundabout: A four-way roundabout with two lanes. 40 vehicles spawn during environment reset. This environment includes merge and split junctions.

Intersection: An unprotected four-way intersection allowing bi-directional traffic as well as U-turns. Negotiation and social behaviors are expected to solve this environment. We initialize 30 vehicles.

Tollgate: Tollgate includes narrow roads to spawn agents and ample space in the middle with multiple tollgates. The tollgates create static obstacles where the crashing is prohibited. We force agent to stop at the middle of tollgate for 3s. The agent will fail if they exit the tollgate before being allowed to pass. 40 vehicles are initialized. Complex behaviors such as deceleration and queuing are expected. Additional states such as whether vehicle is in tollgate and whether the tollgate is blocked are given.

Bottleneck: Complementary to Tollgate, Bottleneck contains a narrow bottleneck lane in the middle that forces the vehicles to yield to others. We initialize 20 vehicles.

Parking Lot: A compact environment with 8 parking slots. Spawn points are scattered in both parking lots or in external roads. 10 vehicles spawn initially and need to navigate toward external roads or enter parking lots. In this environment, we allow agents to back their cars to spare space for others. Maneuvering and yielding are the key to solve this task.

Those environments cover several scenarios in previous simulators [54, 30] but are extended with real-world scenarios rather than atomic scenes. Besides, our environments support dense traffic flow where all vehicles are controlled through the low-level continuous signals. All the environments can be further concatenated to form more complex tasks for the future study of multi-task learning.

We initialize a given number of vehicles in random spawn points, and assign a destination to each of them randomly. Agents are terminated in three situations: reaching the destinations (called *success*), crashing with others or driving out of the road (called *failure*). The new vehicles spawn immediately after existing agents being terminated. Meanwhile, the terminated vehicles remains static for 10 steps (2s) in the original positions, creating impermeable obstacles. Crashing to dead vehicles is also considered failure. This design enables the total number of vehicles to vary in time and exceed the the initial number of agents, which mimics real-world situation and brings more challenges.

In all environments, the local observation of each vehicle contains (1) current states such as the steering, heading, velocity and relative distance to boundaries *etc.*, (2) the navigation information that guides the vehicle toward the destination, and (3) the surrounding information encoded by a vector of 72 Lidar-like distance measures of the nearby vehicles. Vehicles are controlled by continuous steering and acceleration signals. The reward function only contains a dense driving reward for longitudinal movement to the destination and a sparse reward that incentives or penalizes the terminations.

Evaluating a population simulated by MARL is an open question. In this work, we define a set of general-purpose metrics that can characterize different aspects of the SDP systems. We choose three episodic metrics with minimal assumptions on the tasks. We evaluate the competence of population with *success rate*. It is the ratio of the number of agents that reach their goals over the total occurred agents in one episode. We also evaluate the *efficiency* of the population. It indicates the difference between successes and failures in a unit of time $(N_{\text{success}} - N_{\text{failure}})/T$. It is possible for a population to achieve high *success rate* but has low *efficiency*, because the agents in the population are running in very low velocity. The third metric we adopt here is the *safety*. It is defined as the total number of critical failures for all agents in one episode.

5.2 Experiment Setting

We compare our method with the following baselines: We first compare the independent policy optimization (*IPO*) method [34] using PPO [35] as the individual learners. We further encode the nearby agents' states into the input of value functions (centralized critics) following the idea of Mean Field MARL [50] and form the mean field policy optimization (*MFPO*). We also examine different variants of centralized critics, but find MFPO yields best performance (see Appendix). The curriculum learning (*CL*) [29] is also a baseline, where we chunk the training into 4 phases and gradually increase initial agents in each episode from 25% to 100% of the target number.

[2]Environments and evaluation protocols will be open-sourced.

Figure 4: Performance of the trained populations from different MARL methods.

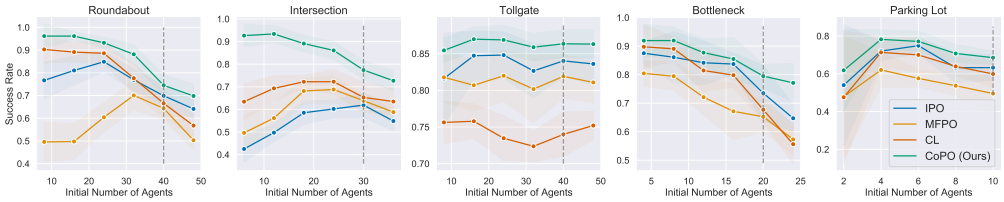


Figure 5: The success rate of different trained populations with varying number of initial agents at test time. The gray lines denote the initial number of agents in the training environments.

Table 1: Success rate of different approaches.

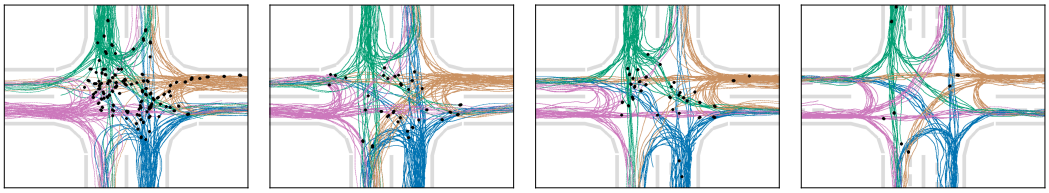
	Roundabout	Intersection	Tollgate	Bottleneck	Parking Lot
IPO	70.81 $\pm$1.95	60.47 $\pm$5.79	82.90 $\pm$2.81	72.43 $\pm$3.79	61.05 $\pm$2.81
MFPO	64.27 $\pm$3.68	67.74 $\pm$4.19	81.05 $\pm$3.07	67.40 $\pm$4.77	53.96 $\pm$4.65
CL	65.48 $\pm$3.96	62.03 $\pm$4.41	73.72 $\pm$3.46	68.81 $\pm$4.39	60.62 $\pm$2.25
CoPO (Ours)	**73.67** $\pm$3.71	**78.97** $\pm$4.23	**86.13** $\pm$1.76	**79.68** $\pm$2.91	**65.04** $\pm$1.59

We conduct experiments on the aforementioned environments and algorithms using RLLib [26], a distributed learning system which allows large-scale parallel experiments. Generally, we host 4 concurrent trials in an Nvidia GeForce RTX 2080 Ti GPU. Each trial consumes 2 CPUs with 4 parallel rollout workers. Each trial is trained over roughly 1M environmental steps, which corresponds to about 55 hours in real-time transportation system and $+2000$ hours of individual driving experience (assume averagely 40 vehicles running concurrently). All experiments are repeated 8 times with different random seeds. Other hyper-parameters are given in Appendix.

5.3 Main Results

Table 1 shows the main results of all the evaluated methods. Because of the bi-level coordination in policy optimization, our CoPO achieves superior success rate in all 5 environments. Noticeably in the most difficult environment the unprotected Intersection, our method outperforms baselines with a substantial margin. That environment requires the population to cooperate frequently in order to determine who should go fist. The populations generated from other methods cause severe congestion in the middle of intersection which prevents any agent going through. See Sec. 5.4 for detailed visualization. Besides, we find the MFPO performs worse than the independent PPO in several environments. That might be because the neighborhood is changing all the time. Simply concatenating or averaging neighbors' observation as the additional input to the value functions makes the training unstable. A detailed comparison of different design choices in MFPO is given in Appendix. Fig. 4 presents the radar plots under the three metrics. In each environment, we normalize each metric to its maximal and minimal values in all evaluated episodes for all algorithms. For *safety*, we use negative values of total crashes in normalization. We can see that CoPO achieves superior results across all three metrics. Noticeably, CoPO has a substantial improvement in safety, showing the importance of bi-level coordination.

To further evaluate the generalization of the populations, we run the trained policies in the test environments with different initial numbers of agents, as shown in Fig. 5. We find that in Roundabout and Intersection, where the interactions are extremely intensive, IPO and MFPO seem overfitting to the initial number of agents. In the environment with sparse traffic, their performance is inferior to that in the environment with more agents. On the contrary, our method does not overfit these two



(a) IPO (b) MFPO (c) CL (d) CoPO (Ours)

Figure 6: Plot of trajectories of trained populations in Intersection. The dark spots indicate the locations where critical failures happen. CoPO produces safer and more coordinated traffic flow.

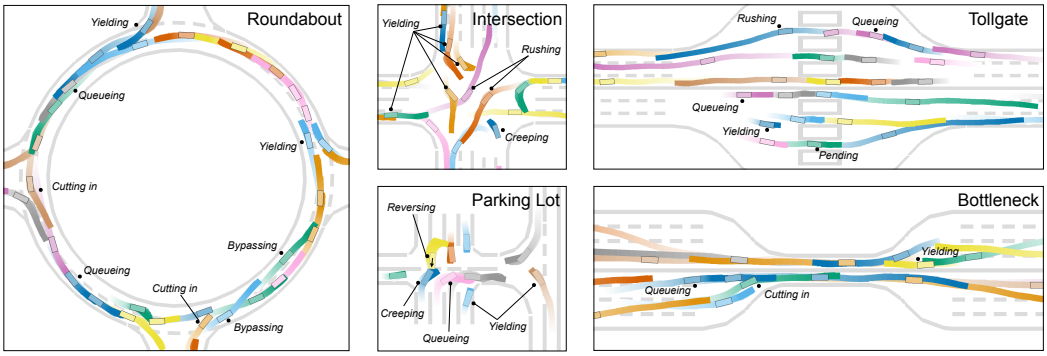


Figure 7: Visualizing the social behaviors of CoPO populations. Diverse behaviors emerge in CoPO populations, which are highlighted with black dots in the plot. We plot the past 25 and future 25 positions of each vehicle with different colors, where the luminosity of a trajectory decreases from light to dark, representing its past to future positions.

environments. In Parking Lot, it seems that all methods has some degree of overfitting to the initial number of agents. The results are consistent with the finding in [24], showing that strategies learned via a single instance of MARL algorithm can overfit to other agents' policies. It is left to future work on improving the generalization of MARL methods.

5.4 Behavioral Analysis

Fig. 6 plots the visualization of all the vehicle trajectories in different colors within one environmental episode, where the agents spawning at same entry share the same colors. The population trained from CoPO exhibits emerging collective behaviors, while the occurred critical failures are much less frequent than the other populations. The population trained from other methods tend to directly rush into the center of Intersection, causing severe congestion and frequent crashes. On the contrary, the population from CoPO tends to drive collectively and successfully avoid crashes through yielding, with superior success rate and high safety. The qualitative result suggests that CoPO can resemble the collective motions widely existing in SDP systems [52]. Fig. 7 further visualizes the temporal behaviors of CoPO agents in 5 environments. Each vehicle in the population tends to drive socially and react to its nearby vehicles. A set of diverse behaviors emerge through the interactions, including socially compliant behaviors such as yielding, queuing, and even reversing to leave more room to others, as well as more aggressive behaviors such as cutting in and rushing. The population trained from CoPO successfully reproduces diverse interaction behaviors in traffic systems.

6 Conclusion

In this work, we develop a novel MARL method called **Co**ordinated **P**olicy **O**ptimization (CoPO) to incorporate social psychology principle to learn neural controller of SDP. Experiments on microscopic traffic simulation show that the proposed method can learn a population which achieves superior performance over the previous MARL methods in three general-purpose metrics. Interestingly, we find the trained vehicles exhibits complex and socially compliant behaviors that improve the performance and safety of the population as a whole.

References

[1] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019.

[2] Eric Bertin, Michel Droz, and Guillaume Grégoire. Hydrodynamic equations for self-propelled particles: microscopic derivation and stability analysis. *Journal of Physics A: Mathematical and Theoretical*, 42(44):445001, 2009.

[3] Bart Besselink and Karl H Johansson. String stability and a delay-based spacing policy for vehicle platoons subject to disturbances. *IEEE Transactions on Automatic Control*, 62(9):4376–4391, 2017.

[4] Kunal Bhattacharya and Tamás Vicsek. Collective decision making in cohesive flocks. *New Journal of Physics*, 12(9):093019, 2010.

[5] Jerome Buhl, David JT Sumpter, Iain D Couzin, Joe J Hale, Emma Despland, Edgar R Miller, and Steve J Simpson. From disorder to order in marching locusts. *Science*, 312(5778):1402–1406, 2006.

[6] Debashish Chowdhury. Collective effects in intra-cellular molecular motor transport: Coordination, cooperation and competition. *Physica A: Statistical Mechanics and its Applications*, 372(1):84–95, 2006.

[7] András Czirók and Tamás Vicsek. Collective behavior of interacting self-propelled particles. *Physica A: Statistical Mechanics and its Applications*, 281(1-4):17–29, 2000.

[8] Yali Du, Lei Han, Meng Fang, Ji Liu, Tianhong Dai, and Dacheng Tao. Liir: Learning individual intrinsic reward in multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 32:4403–4414, 2019.

[9] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[10] Jakob N Foerster, Yannis M Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *arXiv preprint arXiv:1605.06676*, 2016.

[11] Jakob N Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. *arXiv preprint arXiv:1709.04326*, 2017.

[12] Jacques Gautrais, Christian Jost, and Guy Theraulaz. Key behavioural factors in a self-organised fish school model. In *Annales Zoologici Fennici*, volume 45, pages 415–428. BioOne, 2008.

[13] Mike Goslin and Mark R Mine. The panda3d graphics engine. *Computer*, 37(10):112–114, 2004.

[14] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 66–83. Springer, 2017.

[15] Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

[16] Dirk Helbing, Illés Farkas, and Tamas Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487–490, 2000.

[17] Dirk Helbing, Joachim Keltsch, and Peter Molnar. Modelling the evolution of human trail systems. *Nature*, 388(6637):47–50, 1997.

[18] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797, 2019.

[19] Thomas Ihle. Kinetic theory of flocking: Derivation of hydrodynamic equations. *Physical Review E*, 83(3):030901, 2011.

[20] Venkatesan Kanagaraj and Martin Treiber. Self-driven particle model for mixed traffic and other disordered flows. *Physica A: Statistical Mechanics and its Applications*, 509:1–11, 2018.

[21] Matthew G Karlaftis and Eleni I Vlahogianni. Statistical methods versus neural networks in transportation research: Differences, similarities and some insights. *Transportation Research Part C: Emerging Technologies*, 19(3):387–399, 2011.

[22] Florian Knorr, Daniel Baselt, Michael Schreckenberg, and Martin Mauve. Reducing traffic jams via vanets. *IEEE Transactions on Vehicular Technology*, 61(8):3490–3498, 2012.

[23] Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4501–4510, 2020.

[24] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. *arXiv preprint arXiv:1711.00832*, 2017.

[25] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent reinforcement learning in sequential social dilemmas. *arXiv preprint arXiv:1702.03037*, 2017.

[26] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, pages 3053–3062, 2018.

[27] Wim BG Liebrand. The effect of social motives, communication and group size on behaviour in an n-person multi-stage mixed-motive game. *European journal of social psychology*, 14(3):239–264, 1984.

[28] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 2017.

[29] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

[30] Avik Pal, Jonah Philion, Yuan-Hong Liao, and Sanja Fidler. Emergent road rules in multi-agent driving environments. *arXiv preprint arXiv:2011.10753*, 2020.

[31] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4295–4304. PMLR, 2018.

[32] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[33] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[34] Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv e-prints*, pages arXiv–2011, 2020.

[35] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[36] Wilko Schwarting, Alyssa Pierson, Javier Alonso-Mora, Sertac Karaman, and Daniela Rus. Social behavior for autonomous vehicles. *Proceedings of the National Academy of Sciences*, 116(50):24972–24978, 2019.

[37] Steven E Shladover. Review of the state of development of advanced vehicle control systems (avcs). *Vehicle System Dynamics*, 24(6-7):551–595, 1995.

[38] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 5887–5896. PMLR, 2019.

[39] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[40] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[41] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

[42] Justin K Terry, Nathaniel Grammel, Ananth Hari, Luis Santos, and Benjamin Black. Revisiting parameter sharing in multi-agent deep reinforcement learning. 2021.

[43] Martin Treiber and Arne Kesting. The intelligent driver model with stochasticity-new insights into traffic flow oscillations. *Transportation research procedia*, 23:174–187, 2017.

[44] Simon Vanneste, Astrid Vanneste, Siegfried Mercelis, and Peter Hellinckx. Learning to communicate using counterfactual reasoning. *arXiv preprint arXiv:2006.07200*, 2020.

[45] Tamás Vicsek, András Czirók, Eshel Ben-Jacob, Inon Cohen, and Ofer Shochet. Novel type of phase transition in a system of self-driven particles. *Physical review letters*, 75(6):1226, 1995.

[46] Eugene Vinitsky, Aboudy Kreidieh, Luc Le Flem, Nishant Kheterpal, Kathy Jang, Cathy Wu, Fangyu Wu, Richard Liaw, Eric Liang, and Alexandre M Bayen. Benchmarks for reinforcement learning in mixed-autonomy traffic. In *Conference on Robot Learning*, pages 399–409, 2018.

[47] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[48] Meng Wang, Winnie Daamen, Serge P Hoogendoorn, and Bart van Arem. Connected variable speed limits control and car-following control with vehicle-infrastructure communication to resolve stop-and-go waves. *Journal of Intelligent Transportation Systems*, 20(6):559–572, 2016.

[49] Cathy Wu, Aboudy Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M Bayen. Flow: Architecture and benchmarking for reinforcement learning in traffic control. *arXiv preprint arXiv:1710.05465*, page 10, 2017.

[50] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 5571–5580. PMLR, 2018.

[51] Huichu Zhang, Siyuan Feng, Chang Liu, Yaoyao Ding, Yichen Zhu, Zihan Zhou, Weinan Zhang, Yong Yu, Haiming Jin, and Zhenhui Li. Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In *The World Wide Web Conference*, pages 3620–3624, 2019.

[52] Bolei Zhou, Xiaoou Tang, and Xiaogang Wang. Measuring crowd collectiveness. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3049–3056, 2013.

[53] Meng Zhou, Ziyu Liu, Pengwei Sui, Yixuan Li, and Yuk Ying Chung. Learning implicit credit assignment for cooperative multi-agent reinforcement learning. 2020.

[54] Ming Zhou, Jun Luo, Julian Villella, Yaodong Yang, David Rusu, Jiayu Miao, Weinan Zhang, Montgomery Alban, Iman Fadakar, Zheng Chen, Aurora Chongxi Huang, Ying Wen, Kimia Hassanzadeh, Daniel Graves, Dong Chen, Zhengbang Zhu, Nhat Nguyen, Mohamed Elsayed, Kun Shao, Sanjeevan Ahilan, Baokuan Zhang, Jiannan Wu, Zhengang Fu, Kasra Rezaee, Peyman Yadmellat, Mohsen Rohani, Nicolas Perez Nieves, Yihan Ni, Seyedershad Banijamali, Alexander Cowen Rivers, Zheng Tian, Daniel Palenicek, Haitham bou Ammar, Hongbo Zhang, Wulong Liu, Jianye Hao, and Jun Wang. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [N/A]

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]