

Scalable Model-based Policy Optimization for Decentralized Networked Systems

Yali Du^{1*}, Chengdong Ma^{2*}, Yuchen Liu⁵, Runji Lin³, Hao Dong⁵, Jun Wang⁴, Yaodong Yang^{5,†}

Abstract— Reinforcement learning algorithms require a large amount of samples; this often limits their real-world applications on even simple tasks. Such a challenge is more outstanding in multi-agent tasks, as each step of operation is more costly, requiring communications or shifting or resources. This work aims to improve data efficiency of multi-agent control by model-based learning. We consider networked systems where agents are cooperative and communicate only locally with their neighbors, and propose the decentralized model-based policy optimization framework (DMPO). In our method, each agent learns a dynamic model to predict future states and broadcast their predictions by communication, and then the policies are trained under the model rollouts. To alleviate the bias of model-generated data, we restrain the model usage for generating myopic rollouts, thus reducing the compounding error of model generation. To pertain the independence of policy update, we introduce extended value function and theoretically prove that the resulting policy gradient is a close approximation to true policy gradients. We evaluate our algorithm on several benchmarks for intelligent transportation systems, which are connected autonomous vehicle control tasks (Flow and CACC) and adaptive traffic signal control (ATSC). Empirical results show that our method achieves superior data efficiency and matches the performance of model-free methods using true models.

The source code of our algorithm and baselines can be found at <https://github.com/PKU-MARL/Model-Based-MARL>.

I. INTRODUCTION

Many real world problems, such as autonomous driving, wireless communications, multi-player games can be modeled as multi-agent reinforcement learning (MARL) problems, where multiple autonomous agents coexist in a common environment, aiming to maximize its individual or team reward in the long term by interacting with the environment and other agents. Unlike single-agent tasks, multi-agent tasks are more challenging, due to partial observations and unstable environments when agents update

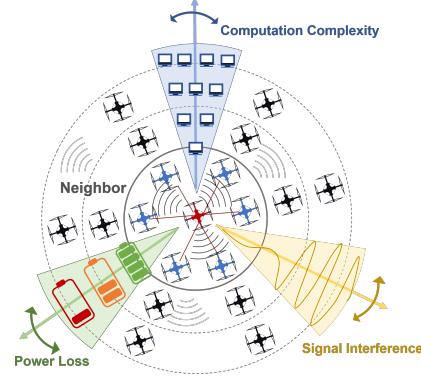


Fig. 1. Communication relationship of UAVs formation and disadvantages of massive communication operations.

their policies simultaneously. Therefore, there are hardly any one-fits-all solutions for MARL problems. For example, in networked systems control (NSC) [1], agents are connected via a stationary network and perform decentralized control based on its local observations and messages from connected neighbors. Examples include connected vehicle control [2], traffic signal control [1], etc.

Despite the emergence of many MARL algorithms [1], [3], [4], [5], the data efficiency problem is often underestimated [6]. On the one hand, in MARL tasks, agents are often connected via communication or coordination for cooperative control, making the decision making more complicated than single agent RL. On the other hand, many real applications demand high sample efficiency, preventing RL being applied. For example, in networked storage operation, one step of operations can be costly with loads shifted across a network and possible power loss [7]. Figure 1 shows that in multiple unmanned aerial vehicle (UAV) formation, when executing military tasks, each step of communication operations will cause power loss and reduce the endurance of UAVs. Frequent and massive communication operations also increase the probability of receiving signal interference for UAVs. In contrast, model-based RL exploits an estimated dynamic model, is empirically more data-efficient than model-free approaches [8], [9]. While the success of model-based RL (MB-RL) has been witnessed in many single agent RL tasks [10], [11], [12], [13], the understanding of its MARL counterpart is still limited. Existing MB-MARL algorithms either limit their field of research on specific scenario, e.g. two-player zero-sum Markov game [14], or tabular RL case [15]. MB-MARL for multi-agent MDPs is still an open problem to be solved [16], with profound challenges such as scalability issues caused by large state-action space and

*The first two authors contributed equally. †Corresponding author.

¹ Yali Du is with King’s College London, 30 Aldwych, London, UK
yali.du@kcl.ac.uk

² Chengdong Ma is with Xiamen University, 422 Siming Road, Xiamen, China mачengdong@stu.xmu.edu.cn. Work done as a research intern at Peking University

³ Runji Lin is with Chinese Academy of Sciences, 95 Zhongguancun East Road, Beijing, China linrunji2021@ia.ac.cn

⁴ Jun Wang is with University College London, 66-72 Gower street, London, UK jun.wang@cs.ucl.ac.uk

⁵ Yuchen Liu is with Peking University, 5 Yiheyuan Road, Haidian District, Beijing, China liuyuchen18@pku.edu.cn

⁵ Hao Dong is with CFCS, School of CS, Peking University hao.dong@pku.edu.cn

⁵ Yaodong Yang is with Institute for AI, Peking University & BIGAI yaodong.yang@pku.edu.cn

incomplete information of other agents' state or actions [17].

In this paper, we restrict our attention to networked system control, where agents are able to communicate with others for the objective of cooperative control, and propose the first decentralized model-based algorithm for networked systems, coined as **Decentralized Model-based Policy Optimization** (DMPO). It is worth noting that the networked systems here are in a broad sense. Actually, DMPO can be widely used to solve the control and decision-making problems of general multi-agent systems. Similarly, "decentralized" is in a broad sense for any multi-agent system in general. Our purpose is to improve the performance of the overall system when the information acquisition of a single agent is very limited. In DMPO, we use localized models to predict future states, and use communication to broadcast their predictions. To alleviate the issue of compounding model error, we adopt a branching strategy [18], [12] by replacing few long-horizon rollouts with many short-horizon rollouts to reduce compounding error in model-generated rollouts. In the policy optimization part, we use decentralizd PPO [19] with a localized extended value function. Theoretically, we prove that the policy gradient computed from extended value function is a close approximation to the true gradient. Empirically, we evaluate our method on adaptive traffic signal control (ATSC) and connected autonomous vehicle (CAVs) control tasks [1], [20], which are extensively studied intelligent transportation systems.

In summary, our contributions are three-fold. Firstly, we propose an algorithmic framework, DMPO, for decentralized model-based reinforcement learning for networked systems. Secondly, we integrate branched rollout to reduce compounding error in model-based rollouts and extended value function to reduce the complexity of computing policy gradient. We theorize that the resulting policy gradient is a close approximation to true policy gradient. Lastly, extensive results on intelligent transportation tasks demonstrate the superiority of DMPO in terms of sample efficiency and performance.

II. RELATED WORK

Reinforcement learning has achieved remarkable success in many decision making tasks [21], [6], [22], [23]. Due to low data efficiency, model-based methods are widely studied as a promising approach for improving sample efficiency [24], [8], [11], [12].

Due to the growing need of multi-agent decision making, such as traffic light control, wireless communications, and multi-player video games, many efforts have been poured in designing MARL algorithms. One line of work focuses on centralized training decentralized execution (CTDE) framework, including policy gradients methods COMA [25], MADDPG [26] and LIIR [27], and value factorization methods VDN [28], QMIX [29], QTRAN [30], etc. In large scale multi-agent systems, however, centralized training might not scale [23], and fully decentralized algorithms are favoured. [31] proposed an algorithm of NSC that can be proven to converge under linear approximation. [3] proposed truncated

policy gradient, to optimize local policies with limited communication. Baking in the idea of truncated Q -learning in [3], we generalize their algorithm to deep RL, rather than tabular RL. Factoring environmental transition into marginal transitions can be seen as factored MDP. [32] used Dynamic Bayesian Network to predict system transition. [33] proposed a tabular RL algorithm to ensure policy improvement at each step. However, our algorithm is a deep RL algorithm, enabling better performance in general tasks. [34], [35], [36], [37], [38] communicate with other agents by learning some hidden information. In comparison, our algorithm only needs to obtain the state of the neighbors.

Early attempts on model-based MARL learning are restricted to special settings. For example, [39] solved single-controller-stochastic games, which is a certain type of two-player zero-sum game; [14] proved that model-based method can be nearly optimally sample efficient in two-player zero-sum Markov games; [40] constructs dynamics and opponents model in a decentralized manner, but it assumes full observability of states and the scalability is not sufficiently verified. [15] extended model-based prioritized sweeping into a MARL scenario, but only restricted to tabular reinforcement learning algorithm, thus unable to deal with more general Markov decision making tasks.

In contrast to existing works, this work tackles the more general multi-agent MDP under partial observability, and proposes the first fully decentralized model-based reinforcement learning algorithm.

III. PROBLEM SETUP

In this section, we introduce multi-agent networked MDP and model-based networked system control.

A. Networked MDP

We consider environments with a graph structure. Specifically, n agents coexist in an underlying undirected and stationary graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Agents are represented as a node in the graph, therefore $\mathcal{V} = \{1, \dots, n\}$ is the set of agents. $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ comprises the edges that represent the connectivity of agents. Agents are able to communicate along the edges with their neighbors. Let N_i denote the neighbor of the agent i including i itself. Let N_i^κ denote the κ -hop neighborhood of i , i.e. the nodes whose graph distance to i is less than or equal to κ . For the simplicity of notation, we also define $N_{-i}^\kappa = \mathcal{V} \setminus N_i^\kappa$.

The corresponding networked MDP is defined as $(\mathcal{G}, \{\mathcal{S}_i, \mathcal{A}_i\}_{i \in \mathcal{V}}, p, r)$. Each agent i has its local state $s_i \in \mathcal{S}_i$, and performs action $a_i \in \mathcal{A}_i$. The global state is the concatenation of all local states: $s = (s_1, \dots, s_n) \in \mathcal{S} := \mathcal{S}_1 \times \dots \times \mathcal{S}_n$. Similarly, the global action is $a = (a_1, \dots, a_n) \in \mathcal{A} := \mathcal{A}_1 \times \dots \times \mathcal{A}_n$. For the simplicity of notation, we define s_{N_i} to be the states of i 's neighbors. The transition function is defined as: $p(s'|s, a) : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$. Each agent possess a localized policy $\pi_i^{\theta_i}(a_i | s_{N_i})$ that is parameterized by $\theta_i \in \Theta_i$, meaning the local policy is dependent only on states of its neighbors and itself. We use $\theta = (\theta_1, \dots, \theta_n)$ to denote the tuple of localized policy parameters, and

Algorithm 1: DMPO framework

Input: rollout length T

- 1: Initialize the model p^{ψ_i} , actor π^{θ_i} and critic V^{ϕ_i} .
- 2: Initialize replay buffers \mathcal{D}_i^E and \mathcal{D}_i^M .
- 3: **for** N epochs **do**
- 4: Take action in environment according to $\pi^{\theta_i}, i \in \mathcal{V}$;
 add to $\mathcal{D}_i^E, i \in \mathcal{V}$
- 5: Train p^{ψ_i} on \mathcal{D}_i^E by maximizing likelihood, $i \in \mathcal{V}$.
- 6: $D_{i,0}^M = \emptyset$.
- 7: **for** B steps **do**
- 8: **for** M rollouts **do**
- 9: Sample s_i^t from $\mathcal{D}_i^E, i \in \mathcal{V}$ // branching
- 10: Generate T -step rollout initing from $\{s_i^t\}_{i \in \mathcal{V}}$ by
 policy π^{θ_i} and model $p^{\psi_i}, i \in \mathcal{V}$
- 11: Append trajectories to \mathcal{D}^M
- 12: **for** G gradient steps **do**
- 13: Update policies θ_i and critics ϕ_i on model data
 sampled from \mathcal{D}^M

$\pi^\theta(a|s) = \prod_{i=1}^n \pi_i^{\theta_i}(a_i|s_{N_i})$ denote the joint policy. Each agent has a reward functions that depends on local state and action: $r_i(s_i, a_i)$, and the global reward function is defined to be the average reward $r(s, a) = \frac{1}{n} \sum_{i=1}^n r_i(s_i, a_i)$. The goal of reinforcement learning is to find a policy π^θ that maximizes the discounted reward,

$$\pi^{\theta^*} = \arg \max_{\pi^\theta} \mathbb{E}_{\pi^\theta} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right], \quad (1)$$

where $\gamma \in (0, 1)$ is the temporal discount factor. The value function is defined as

$$V(s) = \mathbb{E}_{\pi^\theta} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s^0 = s \right] = \frac{1}{n} \sum_{i=1}^n V_i(s). \quad (2)$$

In the last step, we have defined $V_i(s)$, which is the value function for individual reward r_i .

B. Model-based RL

Let $V^{\pi, \hat{p}}$ be the value function of the policy on the estimated model \hat{p} . Towards optimizing $V^{\pi, p}(s)$, a common solution in single agent RL is to build a lower bound as follows and maximize it iteratively [11]:

$$V^{\pi, p}(s) \geq V^{\pi, \hat{p}}(s) - D(\hat{p}_i, \pi_i), \quad (3)$$

where $D(\hat{p}, \pi) \in \mathbb{R}$ bounds the discrepancy between $V_i^{\pi, p}$ and $V_i^{\pi, \hat{p}}$ and can be defined as $D(\hat{p}_i, \pi_i) = \alpha \cdot \mathbb{E}[\|\hat{s}_{i,t+1} - s_{i,t+1}\|]$, where α is a hyperparameter.

IV. DECENTRALIZED MODEL-BASED POLICY OPTIMIZATION

In this section, we formally present DMPO, which is a decentralized model-based reinforcement learning algorithm. Three key components are localized models, decentralized policies and extended value functions.

A. Modeling Networked System

Networked system may have some extent of locality, meaning in some cases, local states and actions do not affect the states of distant agents. In such systems, environmental transitions can be factorized, and agents are able to maintain local models to predict future local states. The factorization is given below.

$$p(s, a) = \prod_{i=1}^n p_i(s'_i | s_{N_i^\kappa}, a_i). \quad (4)$$

While N_i is often not known, we employ the κ -neighbor N_i^κ to approximate the true dynamic by $\hat{p}(s, a) = \prod_{i=1}^n \hat{p}_i(s'_i | s_{N_i^\kappa}, a_i)$, where \hat{p}_i is usually parameterized by unknown variables that we denote as $\psi_i, i \in \mathcal{V}$. Larger κ leads to better approximation of model p , but also more computation overhead. In the following presentation, we would use \hat{p}_i and p^{ψ_i} interchangeably.

For each agent i , we solve the following problem:

$$\pi_i^{k+1}, p_i^{k+1} = \operatorname{argmax}_{\pi_i, p_i} V^{\pi, p} - D(\hat{p}_i, \pi_i). \quad (5)$$

Let $\hat{s}_{i,t+1} = p^{\psi_i}(s_{N_i, t}, a_i)$. We want to minimize $\|\hat{s}_{i,t+1} - s_{i,t+1}\|$. With trajectories sampled from the true environment by π_i , each agent locally updates its model \hat{p}_i . With a localized model \hat{p}_i , agent i learns to update π_i to maximize the reward.

For the training of policies and models, we maintain two data buffers, \mathcal{D}^E for trajectories generated by true environment p and \mathcal{D}^M for data generated by the learned model. The trajectories here consist of s, a, s', r, d , where d is binary, indicating whether the task is completed or reaches maximum episode length. The architecture of our framework is presented in Figure 2. Below we present the details of model and policy updates.

B. Update policies

To optimize the policies, we need to adopt an algorithm that can exploit network structure. Whilst remaining decentralized. Independent RL algorithms that observe only local states are fully decentralized, but they often fail to learn an optimal policy. Centralized algorithms that utilize centralized critics often achieve better performance than decentralized algorithms, but they might not scale to large environments where communication costs are expensive.

For each agent i , denote parameterized policy π^{θ_i} and critic V^{ϕ_i} for fitting optimal policy π_i^* and critic $V_i(s)$. Let $\{(s_{i,\tau}, a_{i,\tau}, r_{i,\tau})\}_{i \in \mathcal{V}, \tau \in \mathcal{B}}$ a minibatch sample from \mathcal{D}^M under policies $\pi^{\theta_i}, i \in \mathcal{V}$. Define the advantage function $\hat{A}^{(t)} = r^{(t)} + \gamma V(s^{(t+1)}) - V(s^{(t)})$. As it is not easy to obtain the true value $V_i(s)$, we adopt an *extended value function*, which is defined as

$$V_i(s_{N_i^\kappa}) = \mathbb{E}_{s_{N_i^\kappa}} \left[\sum_{t=0}^{\infty} r_i^t | s_{N_i^\kappa}^0 = s_{N_i^\kappa} \right], i \in \mathcal{V}. \quad (6)$$

Note that $V_i(s_{N_i^\kappa})$ is a good approximation of $V_i(s)$, with the discrepancy decreasing exponentially with κ . We defer the discussion to Section V.

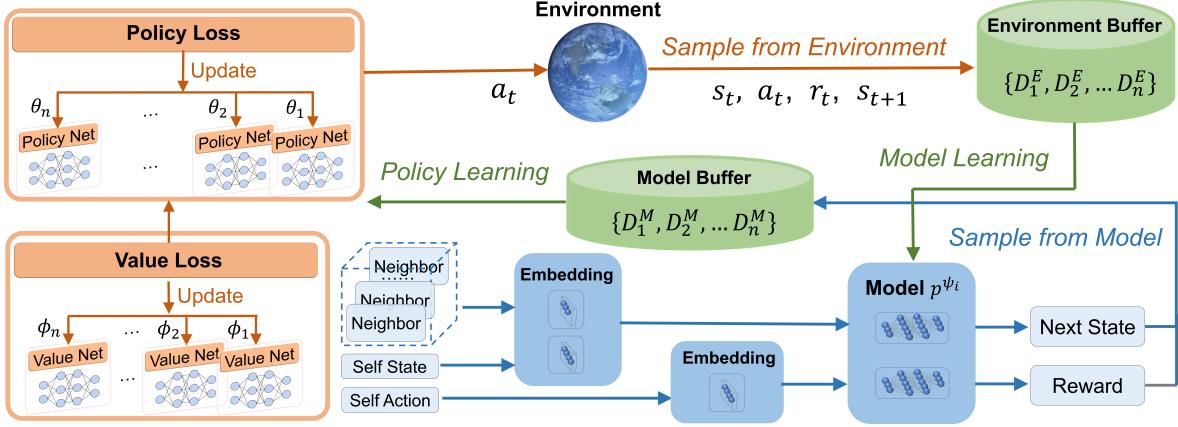


Fig. 2. Architecture of DMPO. Experience data of state transition and rewards from interaction between the environment and the agent are used to train the environment model. Then DMPO can use the large amount of data generated by the interaction between the model and the agent to improve the sampling efficiency in the training process.

To generate the objective for extended value function, or return R_i , we use reward-to-go technique. However, because model rollout is short, standard reward-to-go returns would get a biased estimation of $V_i(s)$, a.k.a. compounding error of model generation. To resolve this issue, we add the value estimation of the last state to the return. The target of $V_i(s_{N_i^\kappa}^t)$ is

$$R_i^t = \sum_{l=0}^{T-t-1} \gamma^l r_i^{t+l} + V^{\phi_i}(s_{N_i^\kappa}^T). \quad (7)$$

The loss of value function is defined as

$$\mathcal{L}(\psi_i) = \frac{1}{|\mathcal{B}|} \sum_{\tau \in \mathcal{B}} (V^{\phi_i}(s_{N_i^\kappa}^\tau) - R_i^\tau)^2. \quad (8)$$

Empirically, we make use of communication within κ -hop neighbors and generate an estimation of global value function as

$$\tilde{V}_i(s_{N_i^\kappa}^t) = \frac{1}{n} \sum_{j \in N_i^\kappa} V_j(s_{N_j^\kappa}^t) \quad (9)$$

The advantage is thus defined as $\hat{A}_t = r_i^t + \gamma \tilde{V}_i(s_{N_i^\kappa}^{t+1}) - \tilde{V}_i(s_{N_i^\kappa}^t)$. The loss function of a DMPO agent is defined as

$$\mathcal{L}(\theta_i) = \frac{1}{|\mathcal{B}|} \sum_{\tau \in \mathcal{B}} (-\log \pi^{\theta_i}(a_{i,\tau} | s_{N_i,\tau}) \hat{A}_{i,\tau} + \beta H(\pi^{\theta_i})). \quad (10)$$

We replace $\log \pi^\theta$ with $\frac{\pi^\theta}{\pi^{\theta_{\text{old}}}}$ and adopt a PPO agent [19] to constrain the policy shift in implementations.

In general, larger κ leads to better results on model approximation and policy learning, but also more communication costs. In algorithmic solutions, computation cost cannot be ignored as larger κ leads to more complex models or policies architectures, posing difficulties on training. We discuss more about this in experiments.

C. Update Model

To perform decentralized model-based learning, we let each agent maintain a localized model. The localized model can observe the state of κ -hop neighbor and the action of itself, and the goal of a localized model is to predict the

information of the next timestep, including state, reward. This process is denoted by $p^{\psi_i}(s'_i, r'_i | s_{N_i^\kappa}, a_i)$. In practice, the data are all stored locally by each agent. We minimize the following objective to update models.

$$\mathcal{L}(\psi_i) = \frac{1}{|\mathcal{B}|} \sum_{\tau \in \mathcal{B}} \|\hat{s}_{i,\tau+1} - s_{i,\tau+1}\|^2. \quad (11)$$

Scaling model-based methods into real tasks can result in decreased performance, even if the model is relatively accurate. One main reason is the compound modeling error when long model rollouts are used, and model error compounds along the rollout trajectory, making the trajectory ultimately inaccurate. To reduce the negative effect of model error, we adopt a branched rollout scheme proposed in [12], [18]. In branched rollout, model rollout starts not from an initial state, but from a state that is randomly selected from the most recent environmental trajectory τ . Additionally, the model rollout length is fixed to be T , as indicated in line 10 in Algorithm 1. Line 4-5 of Algorithm 1 describe the model training steps.

V. THEORETICAL ANALYSIS

In this section, we discuss that the extended value function $V_i(s_{N_i^\kappa})$ is a good approximation of the real value function. We formally state the result in Theorem 1 and defer the proof to Appendix.

Theorem 1: Define $V_i(s_{N_i^\kappa}) = \mathbb{E}_{s_{N_i^\kappa}} [\sum_{t=0}^{\infty} r_i^t | s_{N_i^\kappa}, a_t = s_{N_i^\kappa}]$, and $V_i(s) = \mathbb{E}[\sum_{t=0}^{\infty} r_i^t | s = s]$, then

$$|V_i(s) - V_i(s_{N_i^\kappa})| \leq \frac{r_{\max}}{1-\gamma} \gamma^\kappa. \quad (12)$$

Remark 1: Recall that $V(s) = \frac{1}{n} \sum_{i=1}^n V_i(s)$. From Eq. (12), it is easy to obtain the following result,

$$|V(s) - \frac{1}{n} \sum_{i=1}^n V_i(s_{N_i^\kappa})| \leq \frac{r_{\max}}{1-\gamma} \gamma^\kappa, \quad (13)$$

which indicates that the global value function $V(s)$ can be approximated by the average of localized value functions.

In policy optimization, value functions are used for calculating advantages $\hat{A}^{(t)}$, and we have shown that $V(s)$ can be estimated with the average of localized value functions $\frac{1}{n} \sum_{i=1}^n V_i(s_{N_i^\kappa})$. In practice, an agent might not get the value function of distant agents and can only access the value function of its κ -hop neighbors. However, we can prove that $\tilde{V}_i = \frac{1}{n} \sum_{j \in N_i^\kappa} V_j(s_{N_i^\kappa})$ is already very accurate for calculating the policy gradient for agent i . Theorem 2 formally states this result and the proof is deferred to Appendix.

Theorem 2: Let $\hat{A}_t = r^{(t)} + \gamma V(s^{(t+1)}) - V(s^{(t)})$ be the TD residual, and $g_i = \mathbb{E}[\hat{A}\nabla_{\theta_i} \log \pi_i(a|s)]$ be the policy gradient. If \tilde{A}_t and \tilde{g}_i are the TD residual and policy gradient when value function $V(s)$ is replaced by $\tilde{V}_i(s) = \frac{1}{n} \sum_{j \in N_i^\kappa} V_j(s_{N_i^\kappa})$, we have:

$$|g_i - \tilde{g}_i| \leq \frac{\gamma^{\kappa-1}}{1-\gamma} [1 - (1-\gamma^2) \frac{N_i^\kappa}{n}] r_{\max} g_{\max}, \quad (14)$$

where r_{\max} and g_{\max} denote the upper bound of the absolute value of reward and gradient, respectively.

Remark 2: Theorem 2 justifies that the policy gradients computed based on the sum of the neighboring extended value functions is a close approximation of true policy gradients. The power of this theorem is that the extended value function $V_i(s_{N_i^\kappa})$ requires only the neighboring information, thus easier to approximate and scalable. Despite the reduction in computation, the difference between the approximated and true gradient in Eq. (14) is small.

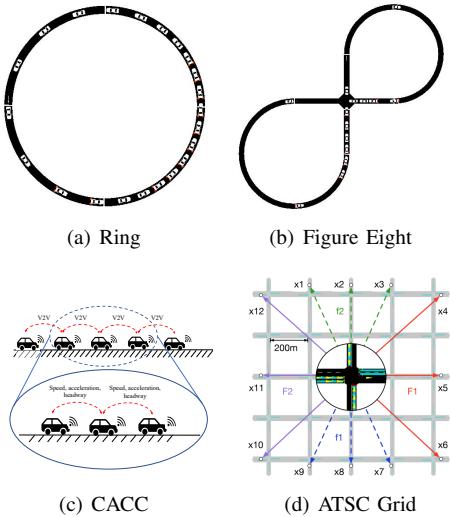


Fig. 3. Visualization of CACC, Flow and ATSC environments. (a) Vehicles travel in a ring to reduce stop-and-go waves. (b) Vehicles travel in a figure eight shaped road section to learn the behavior at an intersection. (c) A line of vehicles that need to keep a stable velocity and desired headway. (d) Synthetic traffic grid that need to learn how to minimize traffic congestion.

VI. EXPERIMENTS

We evaluate DMPO on existing multi-agent environments of intelligent transport systems, which are Connected Autonomous Vehicles including Flow [20] and Cooperative Adaptive Cruise Control (CACC) [1], and Adaptive Traffic

TABLE I
HYPERPARAMETERS FOR DMPO.

	Catch.	Slow.	Fig.Eight	RingAtt.	ATSC
lr of V_i	3e-4	3e-4	5e-5	5e-4	5e-4
lr of π	3e-4	3e-4	5e-5	5e-4	5e-4
lr of p_i	3e-4	3e-4	5e-4	5e-4	2e-4
π_i Net.	[64,64]	[64,64]	[64,64]	[64,64]	[128,128]
V_i Net.	[64,64]	[64,64]	[64,64]	[64,64]	[128,128]
p_i Net.	[16,16]	[16,16]	[16,16]	[16,16]	[64,64]
κ	2	2	3	3	1
Rollout	25	25	25	25	25

Signal Control (ATSC) [1]. The number of agents in these systems is 8, 14, 22 and 25 respectively, and the complexity of the networked systems gradually increases. Figure 3 gives visualization for the environments used in the experiments.

We use three-layered MLP layers for policy, critic and model prediction networks. For policy and critic, all hidden layers are set up to 64 hidden units for Flow and CACC, and 128 for ATSC. For model networks, all hidden layers are set to 16 hidden units for Flow and CACC, and 64 for ATSC. κ for policy and model are set up to 1 in all tasks. For critics, κ is set to 2 for CACC, 3 for Flow and 1 for ATSC. The key hyperparameters for DMPO are summarized in Table I.

A. Baselines

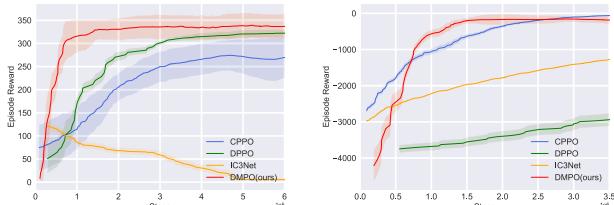
We evaluate the following algorithms in experiments.

- CPPO [19], [20]: Centralized PPO learns a centralized critic $V_i(s)$. This baseline aims to analyze the performance when κ is set to be arbitrarily huge, and is used in [20] as a benchmark algorithm for networked system control.
- DPPO [19]: Decentralized PPO learns an independent actor and critic for each agent. We implement it by using the neighbor's state for extended value estimation.
- IC3Net [41]: A communication-based multi-agent RL algorithm. The agents maintain their local hidden states with a LSTM kernel, and actively determine the communication target. Compared with DPPO, IC3Net employs communication, whereas DPPO agents only observe the states of their neighbors.
- DMPO (our method): DMPO is a decentralized and model-based algorithm based on DPPO. The extended value function is based on κ -hop neighbors.

B. Connected Autonomous Vehicles

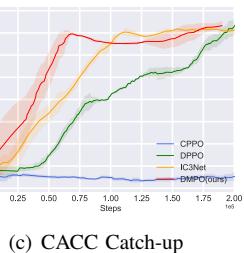
Cooperative Adaptive Cruise Control CACC consists of two scenarios: Catch-up and Slow-down. The objective of CACC is to adaptively coordinate a platoon of 8 vehicles to minimize the car-following headway and speed perturbations based on real-time vehicle-to-vehicle communication. The state of each agent consists of headway h , velocity v , acceleration a , and is shared to neighbors within two steps. The action of each agent is to choose appropriate hyperparameters ($\alpha^\circ, \beta^\circ$) for each OVM controller [42].

Flow environments This task consists of Figure Eight and Ring Attenuation. The objective is to achieve a target speed and avoid collision, which is similar to CACC. The

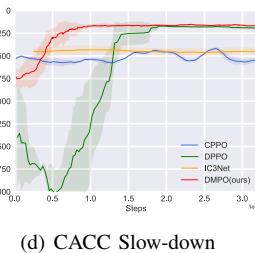


(a) Figure Eight

(b) Ring Attenuation



(c) CACC Catch-up



(d) CACC Slow-down

Fig. 4. Training curves on multi-agent environments. Solid curves depict the mean of trails, and shaded region correspond to standard deviation.

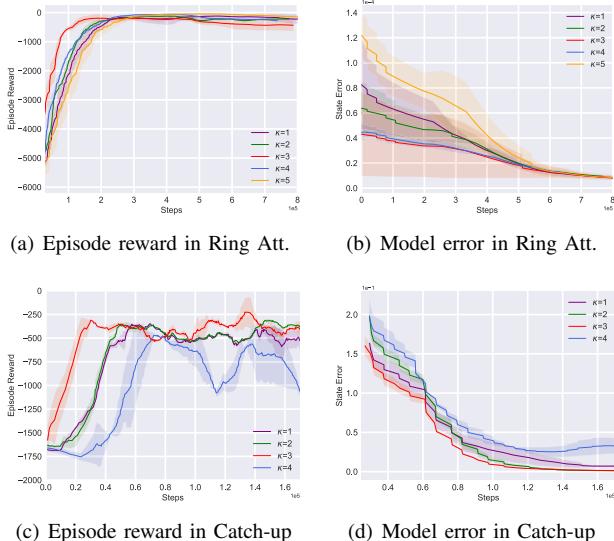
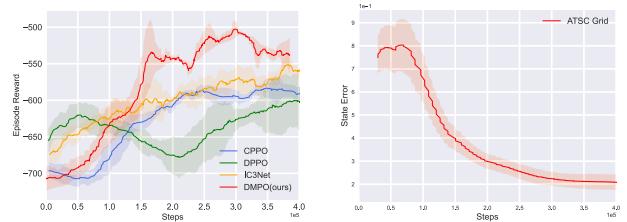


Fig. 5. Training performance on Ring Attenuation and CACC Catch-up under different choices of κ . (a) and (c) report training reward; (b) and (d) report the state error.

road network has a shape of ring or figure “eight”. The figure eight network, previously presented in [43], acts as a closed representation of an intersection. The state consists of velocity and position for the vehicle. The action is the acceleration of the vehicle. In the perspective of a networked system, we assume that the vehicles are connected with the preceding and succeeding vehicle, thus resulting in a loop-structured graph.

Training Results Figure 4 shows the results of episode reward v.s. number of training steps of different algorithms. From the results, we observe that our method achieves highest data efficiency, converging within $1e5$ training steps, exceedingly outperforms CPPO and DPPO.

The comparison between DMPO and DPPO can be viewed



(a) Episode reward in ATSC Grid

(b) Model error in ATSC Grid

Fig. 6. Training curves on ATSC-Grid environment and state error in DMPO algorithm.

as an ablation study of model usage. In figure eight, DMPO increases sample efficiency at the beginning, but as the task becomes difficult, the sample efficiency of our method decreases. In a relatively easy task, ring attenuation, our method increased sample efficiency massively, compared with its model-free counterpart.

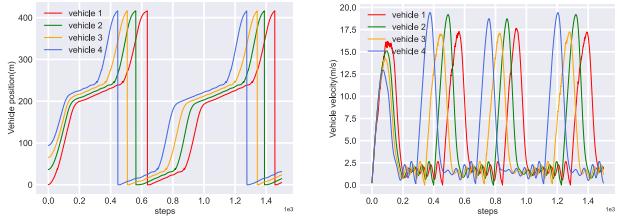
The comparison between the performance of CPPO and DMPO or DPPO can be viewed as an ablation study of extended value function. From the result in four environments, we observe that the performance of CPPO does not exceed that of the algorithms that use extended value function. In this way, we conclude that by using an extended value function, a centralized algorithm can be decomposed into a decentralized algorithm, but the performance would not drop significantly.

Figure 5 shows training curves and the accuracy of our model in predicting the state during training under different choices of κ . The state error is defined as the MSE loss. From the figures, we conclude that neighborhood information is accurate enough for a model to predict the next state in these environments, but the effect of different κ on state prediction is different. When the network structure of DMPO is fixed, the sampling efficiency will increase and the error of our model in predicting the state will decrease with the increase of κ . But when κ is greater than a certain value, the sampling efficiency will decrease and the error of our model in predicting the state will decrease due to the limited fitting ability of the network. The optimal choice of κ is 3 under our network structure in both Ring Attenuation and CACC Catch-up.

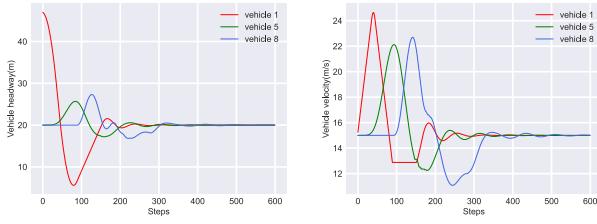
C. Adaptive Traffic Signal Control

The objective of ATSC is to adaptively adjust signal phases to minimize traffic congestion based on real-time road-traffic measurements. Here we use the scenario: a 5×5 synthetic traffic grid. The traffic grid is composed of two-lane roads. In the traffic grid, the peak hour traffic dynamics are simulated by a collection of four time-variant traffic flows, including loading and recovery phases. The state of each agent is all the twelve flows, and all agents have the same action space, which is a set of five pre-defined signal phases.

Training results Figure 6 shows training curves and the accuracy of our model in predicting the state during training in a relatively hard task with more agents, higher dimensional state space and more complex scenario settings. In ATSC, compared to the three model-free baselines, we conclude that the existence of the model increased sample efficiency



(a) Position profiles in Figure Eight (b) Velocity profiles in Figure Eight



(c) Headway profiles in CACC Catch-up (d) Velocity profiles in CACC Catch-up

Fig. 7. Execution performance of the final decision models.

massively. With training, the model error gradually decreases smoothly, indicating that the training process of the model is satisfactory and our model is more and more accurate in predicting future states.

D. Execution Results

Figure 7 shows execution performance of the trained policies based on DMPO. In (a) and (b), we show the position and velocity profiles of four adjacent vehicles 1, 2, 3, 4. Our models control the formation of queues to cross intersection with an orderly process of accelerating to the target velocity and then decelerating to the safe velocity near 0 m/s. We conclude that our models control vehicles to obey traffic rules while improving the efficiency of the overall traffic flow.

In CACC Catch-up, we plot the headway and velocity profiles of three vehicles 1,5,8 at the head, middle and tail.

The headway stabilizes at 20 m which matches the target headway. The velocity stabilizes at 15 m/s, matching the target velocity. Both the velocity and headway of three vehicles with the same interval in queue can be stably controlled around the target.

VII. CONCLUSIONS

In this paper, we propose DMPO, a model-based and decentralized multi-agent RL framework. To preserve the independence of policy learning, we introduce extended value function and the resulting policy gradient is proven to be a close approximation to true policy gradient. Through extensive experiments in several tasks in networked systems, we show that our algorithm matches the performance of some state-of-art multi-agent algorithms and achieves higher data efficiency. From the results, we also conclude that using extended value function instead of centralized value function did not sacrifice performance massively, yet it makes our algorithm scalable. One limitation of this work is that the system is assumed to be factorizable into independent

components, which may not hold in practice. We leave the more general scenarios for future work.

APPENDIX

A. Proof of Theorem 1

Proof: In [4], it was proven that if $s_{N_i^\kappa}$ and $a_{N_i^\kappa}$ are fixed, then no matter how distant states and actions changes, Q -function will not change significantly:

$$|Q_i(s_{N_i^\kappa}, a_{N_i^\kappa}, s_{N_{-i}^\kappa}, a_{N_{-i}^\kappa}) - Q_i(s_{N_i^\kappa}, a_{N_i^\kappa}, s'_{N_{-i}^\kappa}, a'_{N_{-i}^\kappa})| \leq \frac{r_{\max}}{1-\gamma} \gamma^\kappa.$$

As value function is the expectation of Q -function

$$\begin{aligned} V_i(s) &= \mathbb{E}_{a \sim \pi} Q_i(s, a) \\ V_i(s_{N_i^\kappa}) &= \mathbb{E}_{a \sim \pi} Q_i(s_{N_i^\kappa}, a_{N_i^\kappa}), \end{aligned}$$

we have,

$$\begin{aligned} |V_i(s) - V_i(s_{N_i^\kappa})| &= |\mathbb{E}_{a \sim \pi} Q_i(s, a) - \mathbb{E}_{a \sim \pi} Q_i(s_{N_i^\kappa}, a_{N_i^\kappa})| \\ &\leq \mathbb{E}_{a \sim \pi} |Q_i(s, a) - Q_i(s_{N_i^\kappa}, a_{N_i^\kappa})| \\ &\leq \frac{r_{\max}}{1-\gamma} \gamma^\kappa, \end{aligned}$$

which concludes the proof. ■

B. Proof of Theorem 2

Proof: The difference of the gradients is written as

$$\begin{aligned} g_i - \tilde{g}_i &= \mathbb{E}(\hat{A} - \tilde{A}) \nabla_{\theta_i} \log \pi_i(a_i | s_{N_i}) \\ &= \frac{1}{n} \mathbb{E} \left[\sum_{j \notin N_i^\kappa} \hat{A}_j \right] \nabla_{\theta_i} \log \pi_i(a_i | s_{N_i}) \\ &\quad + \frac{1}{n} \mathbb{E} \left[\sum_{j \in N_i^\kappa} (\hat{A}_j - \tilde{A}_j) \right] \nabla_{\theta_i} \log \pi_i(a_i | s_{N_i}) \\ &= \frac{1}{n} \mathbb{E} \left[\sum_{j \notin N_i^\kappa} (r_j + \gamma V_j(s') - V_j(s)) \right] \nabla_{\theta_i} \log \pi_i(a_i | s_{N_i}) \\ &\quad + \frac{1}{n} \mathbb{E} \sum_{j \in N_i^\kappa} [(r_j + \gamma V_j(s') - V_j(s)) \\ &\quad - (r_j + \gamma V_j(s'_{N_i^\kappa}) - V_j(s_{N_i^\kappa}))] \nabla_{\theta_i} \log \pi_i(a_i | s_{N_i}) \\ &= L_1 + L_2. \end{aligned} \quad (15)$$

Note that for any function $b(s)$, $\mathbb{E}[b(s) \nabla_{\theta_i} \log \pi^{\theta_i}(a_i | s_{N_i})] = 0$. Therefore, L_2 in Equation (15) becomes:

$$\begin{aligned} |L_2| &\leq \frac{1}{n} \mathbb{E} \sum_{j \in N_i^\kappa} \gamma |V_j(s') - V_j(s'_{N_i^\kappa})| \|\nabla_{\theta_i} \log \pi_i(a_i | s_{N_i})\| \\ &\leq \frac{|N_i^\kappa| \gamma^{\kappa+1}}{n} r_{\max} g_{\max}. \end{aligned} \quad (16)$$

For L_1 , note that $r_j + \gamma V_j(s') - V_j(s) = -V_j(s) + r_j + \mathbb{E} \sum_{t=1}^{\kappa-2} \gamma^t r_j^t + \gamma^{\kappa-1} V_j(s^{\kappa-1})$. And in an independent network system (where Eq.(4) holds), $s_j^t, a_j^t, t = 1, \dots, \kappa-2$ is not affected by policy π_i if $j \notin N_i^\kappa$, we have that

$$\begin{aligned} |L_1| &\leq \frac{1}{n} \mathbb{E} \sum_{j \notin N_i^\kappa} |\gamma^{\kappa-1} V_j(s^{\kappa-1})| \|\nabla_{\theta_i} \log \pi_i(a_i | s_{N_i})\| \\ &\leq (1 - \frac{|N_i^\kappa|}{n}) \frac{\gamma^{\kappa-1}}{1-\gamma} r_{\max} g_{\max}. \end{aligned} \quad (17)$$

Put Equation (16) and (17) together, we have

$$\begin{aligned}
|g_i - \tilde{g}_i| &\leq |L_1| + |L_2| \\
&\leq \frac{|N_i^\kappa|}{n} \frac{\gamma^{\kappa+1}}{1-\gamma} r_{\max} g_{\max} + \left(1 - \frac{|N_i^\kappa|}{n}\right) \frac{\gamma^{\kappa-1}}{1-\gamma} r_{\max} g_{\max} \\
&= \frac{\gamma^{\kappa-1}}{1-\gamma} [1 - (1-\gamma^2) \frac{|N_i^\kappa|}{n}] r_{\max} g_{\max}.
\end{aligned}$$

-
- ## REFERENCES
- [1] T. Chu, S. Chinchali, and S. Katti, “Multi-agent reinforcement learning for networked system control,” in *ICLR*, 2020.
 - [2] I. G. Jin and G. Orosz, “Dynamics of connected vehicle systems with delayed acceleration feedback,” *Transportation Research Part C: Emerging Technologies*, vol. 46, pp. 46–64, 2014.
 - [3] G. Qu, Y. Lin, A. Wierman, and N. Li, “Scalable multi-agent reinforcement learning for networked systems with average reward,” *NeurIPS*, vol. 33, 2020.
 - [4] G. Qu, A. Wierman, and N. Li, “Scalable reinforcement learning of localized policies for multi-agent networked systems,” in *L4DC*. PMLR, 2020, pp. 256–266.
 - [5] C. Yu, A. Veliu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, “The surprising effectiveness of ppo in cooperative, multi-agent games,” *arXiv preprint arXiv:2103.01955*, 2021.
 - [6] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al., “Grandmaster level in starcraft ii using multi-agent reinforcement learning,” *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.
 - [7] J. Qin, Y. Chow, J. Yang, and R. Rajagopal, “Distributed online modified greedy algorithm for networked storage operation under uncertainty,” *IEEE Transactions on Smart Grid*, vol. 7, no. 2, pp. 1106–1118, 2015.
 - [8] M. Deisenroth and C. E. Rasmussen, “Pilco: A model-based and data-efficient approach to policy search,” in *ICML*. Citeseer, 2011, pp. 465–472.
 - [9] M. P. Deisenroth, G. Neumann, J. Peters, et al., “A survey on policy search for robotics,” *Foundations and trends in Robotics*, vol. 2, no. 1-2, pp. 388–403, 2013.
 - [10] J. Schriftwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al., “Mastering atari, go, chess and shogi by planning with a learned model,” *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.
 - [11] Y. Luo, H. Xu, Y. Li, Y. Tian, T. Darrell, and T. Ma, “Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees,” in *ICLR*, 2019.
 - [12] M. Janner, J. Fu, M. Zhang, and S. Levine, “When to trust your model: Model-based policy optimization,” in *NeurIPS*, 2019.
 - [13] A. S. Morgan, D. Nandha, G. Chalvatzaki, C. D’Eramo, A. M. Dollar, and J. Peters, “Model predictive actor-critic: Accelerating robot skill acquisition with deep reinforcement learning,” in *ICRA*. IEEE, 2021, pp. 6672–6678.
 - [14] K. Zhang, S. Kakade, T. Basar, and L. Yang, “Model-based multi-agent rl in zero-sum markov games with near-optimal sample complexity,” *NeurIPS*, vol. 33, 2020.
 - [15] E. Bargiacchi, T. Verstraeten, and D. Roijers, “Cooperative prioritized sweeping,” in *AAMAS 2021*. IFAAMAS, 2021, pp. 160–168.
 - [16] K. Zhang, Z. Yang, and T. Bašar, “Multi-agent reinforcement learning: A selective overview of theories and algorithms,” *Handbook of Reinforcement Learning and Control*, pp. 321–384, 2021.
 - [17] R. E. Wang, J. C. Kew, D. Lee, T.-W. E. Lee, T. Zhang, B. Ichter, J. Tan, and A. Faust, “Model-based reinforcement learning for decentralized multiagent rendezvous,” *arXiv preprint arXiv:2003.06906*, 2020.
 - [18] R. S. Sutton, “Integrated architectures for learning, planning, and reacting based on approximating dynamic programming,” in *Machine learning proceedings 1990*. Elsevier, 1990, pp. 216–224.
 - [19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
 - [20] E. Vinitsky, A. Kreidieh, L. Le Flem, N. Kheterpal, K. Jang, C. Wu, F. Wu, R. Liaw, E. Liang, and A. M. Bayen, “Benchmarks for reinforcement learning in mixed-autonomy traffic,” in *CoRL*. PMLR, 2018, pp. 399–409.
 - [21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, p. 529, 2015.
 - [22] D. Silver, T. Hubert, J. Schriftwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al., “A general reinforcement learning algorithm that masters chess, shogi, and go through self-play,” *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.
 - [23] L. Han, P. Sun, Y. Du, J. Xiong, Q. Wang, X. Sun, H. Liu, and T. Zhang, “Grid-wise control for multi-agent reinforcement learning in video game ai,” in *ICML*. PMLR, 2019, pp. 2576–2585.
 - [24] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.
 - [25] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, “Counterfactual multi-agent policy gradients,” in *AAAI*, 2018.
 - [26] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” in *NeurIPS*, 2017, pp. 6379–6390.
 - [27] Y. Du, L. Han, M. Fang, T. Dai, J. Liu, and D. Tao, “Liir: Learning individual intrinsic reward in multi-agent reinforcement learning,” in *NeurIPS*, 2019.
 - [28] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, et al., “Value-decomposition networks for cooperative multi-agent learning based on team reward,” in *AAMAS*, 2018, pp. 2085–2087.
 - [29] T. Rashid, M. Samvelyan, C. S. Witt, G. Farquhar, J. Foerster, and S. Whiteson, “Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning,” in *ICML*, 2018, pp. 4292–4301.
 - [30] K. Son, D. Kim, W. J. Kang, D. Hostallero, and Y. Yi, “Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning,” in *ICML*, 2019.
 - [31] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, “Fully decentralized multi-agent reinforcement learning with networked agents,” in *ICML*, 2018, pp. 5872–5881.
 - [32] C. Guestrin, D. Koller, and R. Parr, “Multiagent planning with factored mdps,” in *NeurIPS*, vol. 1, 2001, pp. 1523–1530.
 - [33] T. D. Simao and M. T. Spaan, “Safe policy improvement with baseline bootstrapping in factored environments,” in *AAAI*, vol. 33, no. 01, 2019, pp. 4967–4974.
 - [34] J. Ruan, Y. Du, X. Xiong, D. Xing, X. Li, L. Meng, H. Zhang, J. Wang, and B. Xu, “Gcs: Graph-based coordination strategy for multi-agent reinforcement learning,” in *AAMAS*, 2022.
 - [35] Y. Du, B. Liu, V. Moens, Z. Liu, Z. Ren, J. Wang, X. Chen, and H. Zhang, “Learning correlated communication topology in multi-agent reinforcement learning,” in *AAMAS*, 2021, pp. 456–464.
 - [36] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, “Learning to communicate with deep multi-agent reinforcement learning,” in *NeurIPS*, 2016, pp. 2137–2145.
 - [37] C. Zhang and V. Lesser, “Coordinating multi-agent reinforcement learning with limited communication,” in *AAMAS*, 2013, pp. 1101–1108.
 - [38] S. Sukhbaatar, A. Szlam, and R. Fergus, “Learning multiagent communication with backpropagation,” in *NeurIPS*, 2016, pp. 2252–2260.
 - [39] R. I. Brafman and M. Tennenholtz, “A near-optimal polynomial time algorithm for learning in certain classes of stochastic games,” *Artificial Intelligence*, vol. 121, no. 1-2, pp. 31–47, 2000.
 - [40] W. Zhang, X. Wang, J. Shen, and M. Zhou, “Model-based multi-agent policy optimization with adaptive opponent-wise rollouts,” in *IJCAI*, Z. Zhou, Ed. ijcai.org, 2021, pp. 3384–3391.
 - [41] A. Singh, T. Jain, and S. Sukhbaatar, “Learning when to communicate at scale in multiagent cooperative and competitive tasks,” *arXiv preprint arXiv:1812.09755*, 2018.
 - [42] M. Bando, K. Hasebe, A. Nakayama, A. Shibata, and Y. Sugiyama, “Dynamical model of traffic congestion and numerical simulation,” *Physical review E*, vol. 51, no. 2, p. 1035, 1995.
 - [43] C. Wu, A. Kreidieh, E. Vinitsky, and A. M. Bayen, “Emergent behaviors in mixed-autonomy traffic,” in *CoRL*. PMLR, 2017, pp. 398–407.