Sample and Communication Efficient Fully Decentralized MARL Policy Evaluation via a New Approach: Local TD update

Hairi
University of Wisconsin-Whitewater
Whitewater, USA
hairif@uww.edu

Zifan Zhang
North Carolina State University
Raleigh, USA
zzhang66@ncsu.edu

Jia Liu
The Ohio State University
Columbus, USA
liu@ece.osu.edu

ABSTRACT

In actor-critic framework for fully decentralized multi-agent reinforcement learning (MARL), one of the key components is the MARL policy evaluation (PE) problem, where a set of N agents work cooperatively to evaluate the value function of the global states for a given policy through communicating with their neighbors. In MARL-PE, a critical challenge is how to lower the sample and communication complexities, which are defined as the number of training samples and communication rounds needed to converge to some ϵ-stationary point. To lower communication complexity in MARL-PE, a "natural" idea is to perform multiple local TD-update steps between each consecutive rounds of communication to reduce the communication frequency. However, the validity of the local TD-update approach remains unclear due to the potential "agent-drift" phenomenon resulting from heterogeneous rewards across agents in general. This leads to an interesting open question: *Can the local TD-update approach entail low sample and communication complexities?* In this paper, we make the first attempt to answer this fundamental question. We focus on the setting of MARL-PE with average reward, which is motivated by many multi-agent network optimization problems. Our theoretical and experimental results confirm that allowing multiple local TD-update steps is indeed an effective approach in lowering the sample and communication complexities of MARL-PE compared to consensus-based MARL-PE algorithms. Specifically, the local TD-update steps between two consecutive communication rounds can be as large as $O(1/\epsilon^{1/2}\log(1/\epsilon))$ in order to converge to an ϵ-stationary point of MARL-PE. Moreover, we show theoretically that in order to reach the optimal sample complexity, the communication complexity of local TD-update approach is $O(1/\epsilon^{1/2}\log(1/\epsilon))$.

CCS CONCEPTS

• **Theory of computation → Distributed algorithms**; • **Computing methodologies → Multi-agent systems**.

KEYWORDS

Multi-agent reinforcement learning, policy evaluation, TD learning, sample and communication complexities

ACM Reference Format:
Hairi, Zifan Zhang, and Jia Liu. 2024. Sample and Communication Efficient Fully Decentralized MARL Policy Evaluation via a New Approach: Local TD update. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 16 pages.

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand.

1 INTRODUCTION

1) **Background and Motivation:** With the recent success of reinforcement learning (RL) techniques in the dynamic decision-making process [26], MARL, a natural extension of RL to multi-agent systems, has also received increasing attention. Compared to traditional RL, the richness of multi-agent systems has given rise to far more diverse problem settings in MARL, including cooperative, competitive, and mixed MARL (see [34] for an excellent survey). In this paper, we are interested in *fully decentralized cooperative* MARL, which has found a wide range of applications in the field of networked large-scale systems, such as power networks [3, 22], autonomous driving [23, 33], wireless network [31] and so on. A defining feature of fully decentralized cooperative MARL is that all agents in the system collaborate to learn a joint policy to maximize long-term system-wide total rewards through communicating with each other. However, due to the decentralized nature (i.e., lack of a centralized infrastructure) of fully decentralized cooperative MARL, the collaboration between the agents can only rely on some speical algorithmic designs to induce a "consensus" that can be reached by all agents.

In a consensus-based actor-critic framework, one of the key components is the MARL policy evaluation (PE) problem, where a set of N agents work cooperatively to evaluate the value function of the global states for a given joint policy. Just as the PE problem in single-agent RL, temporal difference (TD) learning [25] has been the prevailing method for MARL-PE thanks to its simplicity and effectiveness. Simply speaking, the key idea of TD learning is to learn the value function by using the Bellman equation to bootstrap from the current estimated value function.

However, as mentioned earlier, the decentralized nature of the MARL-PE problem necessitates communication among agents for TD learning. Hence, a critical challenge in consensus-based MARL-PE is how to lower the *sample and communication complexities*, which are defined as the required number of training samples and rounds of communications between neighboring agents to converge to an ϵ-stationary point of the MARL-PE problem.

To lower communication complexity for solving MARL-PE problems, a "natural" idea is to use an "infrequent communication" approach where we perform multiple local TD-update steps between each consecutive rounds of communication to reduce the communication frequency. However, the validity of the "local TD-update" approach remains unclear due to the potential "agent-drift" phenomenon resulted from heterogeneous rewards across agents (more on this soon). This leads to two interesting open questions:

1) *Can the local TD-update approach achieve low sample and communication complexities for solving MARL-PE?*
2) *If the answer to 1) is "yes," how does the local TD-steps approach perform in comparison to other approaches?*

In this paper, we make the first attempt to answer the above open questions. However, unlike conventional MARL research that adopts discounted reward, in this paper, we are particularly interested in the cooperative MARL setting with *average reward* [7, 20, 28, 29, 35]. The average reward setting of MARL-PE is motivated by and highly relevant for many multi-agent and network optimization problems that care about "average performances" (e.g., average throughput, average latency, and average energy consumption in multi-hop wireless networks).

2) **Technical Challenges:** Answering Questions 1) and 2) above is highly non-trivial due to several technical challenges in the convergence analysis of the local TD-update approach. Notably, it is easy to see that the structure of TD learning in consensus-based cooperative MARL resembles that of decentralized stochastic gradient descent (DSGD) method in consensus-based decentralized optimization[13, 17, 18]. Thus, it is tempting to believe that one can borrow convergence analysis techniques of DSGD and apply them in TD learning. However, despite such similarities, there also exist significant differences between TD learning in MARL and DSGD.

- *Structural Differences:* First, we note that TD learning is *not* a true gradient-based method since TD error is not a gradient estimator of any static objective function which is well-defined in a consensus-based decentralized optimization problem. Also, in decentralized optimization, the gradient terms are often assumed to be bounded. However, when using approximation for value function in TD learning, TD-errors can not be assumed to be bounded without further assuming that the approximation parameters lie in some compact set.
- *Markovian Noise in TD Learning:* In RL/MARL problems, there exists an underlying Markovian dynamic process across time steps, where the state distribution may differ at different time steps. By contrast, in decentralized optimization, it is often safe to assume that the data at each agent are independently distributed. Thus, it is not possible to directly apply convergence analysis techniques of decentralized optimization in TD learning for MARL-PE. The coupling and dependence among samples renders the convergence analysis of TD learning in MARL far more challenging.
- *"Agent-Drift" Phenomenon:* Due to heterogeneity nature of the rewards across agents, executing multiple local TD-update steps would inevitably pull the value functions toward the direction of local value functions rather than the global value function, leading to the "agent-drift" phenomenon. Hence, it is unclear under such "tug of war" whether local TD-update steps help or hurt the convergence of TD learning in MARL-PE. Because of the agent-drift effect, the number of local TD update steps has to be chosen judiciously to mitigate the potentially large divergence of the value functions among agents between consecutive communication rounds.

3) **Main Results and Contribution:** The main contribution of this paper is that we overcome the above challenges in analyzing the upper bounds of the sample and communication complexities for the local TD-update approach in cooperative fully decentralized MARL-PE. By doing so, we shed light on the effect of local TD-update steps in the consensus-based TD learning in MARL-PE with average reward. We summarize our main results in this paper as follows:

- Both theoretically and empirically, we show that allowing multiple local TD-update steps is indeed a valid approach that can significantly lower communication complexities of MARL-PE compared to vanilla consensus-based decentralized TD learning algorithms [5, 6, 35]. Specifically, we show that under the condition of achieving $\mathcal{O}(1/\epsilon \log^2(1/\epsilon))$ sample complexity (which differs from the state-of-the-art sample complexity only by a log factor), the local TD-update approach can allow up to $\mathcal{O}(1/\epsilon^{1/2}\log(1/\epsilon))$ local TD-update steps and the communication complexity upper bound is $\mathcal{O}(1/\epsilon^{1/2}\log(1/\epsilon))$. Compared to vanilla algorithms, this improves the communication complexity by a factor of $\mathcal{O}(1/\epsilon^{1/2})$.
- In comparison with another notable batching approach, we show that the local TD-update approach not only matches the communication complexity of the batching approach, but also achieves a better sample complexity than that of the batching approach [7] by a factor of $\mathcal{O}(1/\epsilon^{1/2})$ in average reward setting. Our extensive empirical results also verify the performance of the local TD-update approach and confirm our theoretical results compared to the vanilla TD learning and batching approaches with both synthetic and real-world datasets.

The rest of the paper is organized as follows. In Section 2, we review the literature to put our work in comparative perspectives. In Section 3, we present the system model and formulation of the MARL-PE problem in the average reward setting. In Section 4, we introduce the decentralized TD learning algorithm with multiple local TD-update steps for MARL-PE. In Section 5, we provide the theoretical convergence analysis for the decentralized TD learning algorithm with multiple local TD-update steps. In addition, we provide comparisons of both sample and communication complexities of the proposed local TD-update approach with other methods. Section 6 presents numerical results and Section 7 concludes this paper. Due to space limitation, some proof details and additional experiments are relegated to the supplementary material .

2 RELATED WORK

In this section, we provide an overview on two lines of research that are related to this work: i) multi-agent reinforcement learning policy evaluation; and ii) single-agent RL policy evaluation.

1) Multi-agent reinforcement learning policy evaluation: To our knowledge, the work in [35] proposed the first fully decentralized multi-agent actor-critic algorithm using TD learning in the critic step, which solves the PE problem in average reward setting. However, the convergence results for both its critic and actor steps are asymptotic. Finite-time analysis of MARL-PE problem using distributed TD learning algorithm has been first studied in [5] under the i.i.d. sampling assumption, and later the work in [6] generalized the result to Markovian sampling assumption only in

discounted reward settings. In [14], a compressed algorithm is proposed where, instead of sending a vector, only a single entry is sent during communication. However, their communication complexity (i.e., the number of communication rounds) remains the same as sample complexity and the convergence is only asymptotic. In [2], a lazy communication algorithm is proposed assuming a central controller, which is different from the fully decentralized setting that we consider in this paper.

It is worth noting that many of the above existing distributed TD learning algorithms [5, 6, 35] for MARL-PE perform *frequent* consensus rounds (i.e., one round of communication per local TD update) to share the value functions among neighbors. Specifically, in these algorithms, agents share the value functions to their neighbors in every sampling step, which causes the communication complexity to be the same as the sample complexity. In this paper, we consider an infrequent communication framework that allows the agents to do multiple local TD-update steps and communicate with the neighbors once every $K(\gg 1)$ rounds. In [4, 7], complete actor-critic algorithms have been proposed and the batching approach has been used in the critic step, which corresponds to MARL-PE, in discounted and average reward respectively. In this batching approach [7], consensus is performed in every $M = O(1/\epsilon)$ samples, which in return only requires $O(1/\epsilon^{1/2}\log(1/\epsilon))$ communication complexity. Detailed discussions on the comparison of the local TD approach and batching approach is provided in Section 5.3.

We also remark that there exists another class of approaches [11, 16, 21, 30, 36] that solve the MARL-PE problem by formulating MARL-PE into optimizing projected Bellman error or its variants, where the proposed algorithms require frequent communications. This class of algorithms do not use the on-policy TD learning approach as we do in our paper. In [9], the paper optimizes communication in order to comply with the bandwidth restriction and minimize the collision between pair-wise channels. However, this work adopts centralized learning and distributed execution paradigm, where in our paper, the learning process is fully decentralized.

2) Single-agent reinforcement learning policy evaluation: For single-agent RL, policy evaluation problems have been extensively studied in terms of asymptotic convergence [27–29] for both discounted and average reward settings, later finite-time convergence under i.i.d. sampling assumption [10] and under Markovian sampling assumption using different techniques [1, 24] in discounted reward setting. Further, using batching TD learning [32] yields state-of-the-art sample complexity $O(1/\epsilon\log(1/\epsilon))$ in the discounted reward setting. For average reward setting, [19] yields a sample complexity of $O(1/\epsilon^2\log^3(1/\epsilon))$, where the sample complexity is worse than that in our multi-agent setting. To the best of our knowledge, the sample complexity of $O((1/\epsilon)\log^2(1/\epsilon))$ in [24] is the state-of-the-art sample complexity for the single agent average-reward RL policy evaluation problem. However, there is no notion of "communication with other agents" due to the single-agent nature. Thus, results in this area, though related, are not directly comparable to our work in terms of communication complexity.

3 DISTRIBUTED POLICY EVALUATION IN MULTI-AGENT REINFORCEMENT LEARNING

Throughout this paper, $\|\cdot\|$ denotes the ℓ_2-norm for vectors and the ℓ_2-induced norm for matrices. $\|\cdot\|_F$ denotes the Frobenius norm for matrices. $(\cdot)^T$ denotes the transpose for a matrix or a vector.

3.1 System Model

Consider a multi-agent system with N agents, denoted by $\mathcal{N} = \{1, \cdots, N\}$, operating in a networked environment. Let $\mathcal{E}$ be the edge set for a given network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. To formulate our MARL problem and facilitate our subsequent discussions, we first define the notion of networked multi-agent Markov decision process (MDP) in the average reward setting as follows.

Definition 1 (Networked Multi-Agent MDP)**.** Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a communication network that connects N agents. A networked multi-agent MDP is defined by following five-tuple:

$$(\mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, P, \{r^i\}_{i\in\mathcal{N}}, \mathcal{G}),$$

where $\mathcal{S}$ is the global state space, $\mathcal{A}^i$ is the action set for agent i. Let $\mathcal{A} = \prod_{i\in\mathcal{N}}\mathcal{A}^i$ be the joint action set of all agents. $P : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to [0, 1]$ is the global state transition function and $r^i : \mathcal{S}\times\mathcal{A}$ is the local reward function for agent i.

In this paper, we assume that the global state space $\mathcal{S}$ is finite. We also assume that at time step $t \geq 0$, all agents can observe the current global state s_t. However, each agent can only observe its own reward r^i_{t+1}, i.e., agents do not observe or share rewards with other agents. Each agent $i \in \mathcal{N}$ receives a deterministic reward $r^i(s, a)$ given the global state s and joint action a [1].

In our MARL system, each agent chooses its action following its local policy π^i that is conditioned on the current global state s, i.e., $\pi^i(a^i|s)$ is the probability for agent i to choose an action $a^i \in \mathcal{A}^i$. Then, the joint policy $\pi : \mathcal{S}\times\mathcal{A} \to [0, 1]$ can be written as $\pi(a|s) = \prod_{i\in\mathcal{N}}\pi^i(a^i|s)$.

The global long-term average reward for a given joint policy π in average reward setting is defined as follows:

$$\begin{aligned} J_\pi &= \lim_{T\to\infty}\frac{1}{T}\mathbb{E}\left(\sum_{t=0}^{T-1}\frac{1}{N}\sum_{i\in\mathcal{N}} r^i_{t+1}\right) \\ &= \sum_{s\in\mathcal{S}} d(s)\sum_{a\in\mathcal{A}}\pi(a|s)\cdot\bar{r}(s, a), \end{aligned} \tag{1}$$

where $d(\cdot)$ is the steady state distribution, which is guaranteed to exist due to the Assumption 1 below, and $\bar{r}(s, a) = \frac{1}{N}\sum_{i\in\mathcal{N}} r^i(s, a)$. In other words, in the average reward setting, J_π evaluates the performance of the given policy π at steady state as given in (1).

3.2 Technical Assumptions

We now state the following assumptions for the MARL system described above.

Assumption 1. For the given policy π, we assume the induced Markov chain $\{s_t\}_{t\geq 0}$ is irreducible and aperiodic.

[1] For simplicity of the presentation, we assume that the rewards are deterministic. For more general stochastic rewards, the results are straightforward.

Assumption 2. The reward r_t^i is uniformly bounded by a constant $r_{\max} > 0$ for any $i \in \mathcal{N}$ and $t \geq 0$.

Assumption 3. Let A be a consensus weight matrix for a given communication network $\mathcal{G}$. There exists a positive constant $\eta > 0$ such that $A \in \mathbb{R}^{N\times N}$ is doubly stochastic and $A_{ii} \geq \eta$, $\forall i \in \mathcal{N}$. Moreover, $A_{ij} \geq \eta$ if i, j are connected, otherwise $A_{ij} = 0$.

Assumption 4. The global value function is parameterized by linear functions, i.e., $V(s; w) = \phi(s)^\top w$ where

$$\phi(s) = [\phi_1(s), \cdots, \phi_n(s)]^\top \in \mathbb{R}^n$$

is the feature vector associated with the state $s \in \mathcal{S}$. We typically assume the dimension of the vector is smaller than the cardinality of the state space, i.e. $n < |\mathcal{S}|$. The feature vectors $\phi(s)$ are uniformly bounded for any $s \in \mathcal{S}$. Without loss of generality, we assume that $\|\phi(s)\| \leq 1$. Furthermore, the feature matrix $\Phi \in \mathbb{R}^{|\mathcal{S}|\times n}$ is full column rank. Also, for any $u \in \mathbb{R}^n$, $\Phi u \neq \mathbf{1}$, where $\mathbf{1}$ is an all-one vector.

Assumption 1 guarantees that there exists a unique stationary distribution over $\mathcal{S}$ for the induced Markov chain by the given policy π. In other words, it guarantees that the steady state distribution $d(\cdot)$ induced by the policy π is well defined. Assumption 2 is common in the RL literature (see, e.g., [5, 32, 35]) and easy to be satisfied in many practical MDP models with finite state and action spaces. Assumption 3 is standard in the distributed multi-agent optimization literature [17]. This assumption says that non-zero entries of the weight matrix A needs to be lower bounded by a positive value η. Note that this characterization of the weight matrix is a rich representation, as for the same graph/topology $\mathcal{G}$, the weights can vary, which correspond to different consensus effects. Assumption 4 on features is standard and has been widely adopted in the literature, e.g., [7, 19, 24, 28, 35]. The goal of this assumption is to approximate the value function as follows:

$$V(s) \approx V(s; w) = \phi(s)^\top w$$

where $\phi(s)$ is the aforementioned feature vector associated with state $s \in \mathcal{S}$.

4 DECENTRALIZED TD LEARNING WITH LOCAL TD-UPDATE STEPS FOR MARL-PE

In this section, we introduce the decentralized TD learning algorithm with local TD-update steps (i.e., infrequent communication), which is illustrated in Algorithm 1 [2]. Given a joint policy π, the goal of the MARL-PE in the decentralized setting is that the agents collaborate in a consensus manner to characterize the global value function. Specifically, each agent i maintains a value function approximation parameter w^i locally, which estimates the global value function as follows:

$$V(s; w^i) = \phi(s)^\top w^i.$$

The local TD-update algorithm for MARL-PE contains two loops. The outer loop is the communication rounds, where consensus update (Line 12 in Algorithm 1) is performed for L rounds in total. The inner loop is local TD-update steps (Line 10 in Algorithm 1), which are executed K times in between consecutive communication rounds. Locally, each agent performs local TD-updates within each communication round $l \in \{0, \cdots, L-1\}$ as follows:

$$w_{l,k+1}^i = w_{l,k}^i + \beta \cdot \delta_{l,k}^i \cdot \phi(s_{l,k}), \tag{2}$$

where $\beta > 0$ is the constant step size and $\delta_{l,k}^i$ is the local TD error, which is defined as follows

$$\delta_{l,k}^i := r_{l,k+1}^i - \mu_{l,k}^i + \phi(s_{l,k+1}) w_{l,k}^i - \phi(s_{l,k}) w_{l,k}^i,$$

and $\mu_{l,k}^i$ tracks the local average reward, which is updated as follows

$$\mu_{l,k+1}^i = (1-\beta)\mu_{l,k}^i + \beta r_{l,k+1}^i. \tag{3}$$

We remark that Eq. (3) is the *key difference* between the average reward setting and the conventional discounted reward setting in MARL-PE. In the discounted reward setting, there is no μ^i-terms. The use of the μ^i-term is to keep track of the local average reward for agent i. Surprisingly, we will show later that consensus and finite-time convergence results on w^i parameters can be obtained without performing consensus on these μ^i terms. We also note that each execution of Eq. (2) is considered performing one local TD learning step. Within each inner loop, this local TD update step is performed K times.

Due to the privacy of the reward signals in the fully decentralized setting, the agents are unable to access the rewards of any other agents, let alone the average rewards. Therefore, communication/sharing of the value function approximation parameters among the neighbors is necessary [4, 5, 7, 35]. This step is often referred to as *consensus* update, which is defined as follows:

$$w_{l+1,0}^i = \sum_{j\in\mathcal{N}_i} A_{ij} w_{l,K}^j, \tag{4}$$

where $\mathcal{N}_i$ denotes the set of neighbors for agent i. In other words, after performing K local TD-update steps, each agent shares its parameter with the neighbors, receives the ones from the neighbors, and then updates its own parameter in a weighted aggregation as shown in Eq. (4).

We note that in our algorithm, the infrequent communication is achieved by agents communicating with neighbors periodically with the period being K. We also note that when $K = 1$, our algorithm reduces to the vanilla distributed TD learning algorithm [5, 6, 35]. Therefore, the vanilla distributed TD learning can be viewed as a special case of our proposed algorithm.

5 CONVERGENCE ANALYSIS OF THE LOCAL TD-UPDATE APPROACH FOR MARL-PE

In this section, we present the convergence results for Algorithm 1, which further imply both the sample and communication complexities of the local TD-update approach for MARL-PE. To characterize the convergence, we define the following quantities:

$$\Psi := \mathbb{E}[(\phi(s') - \phi(s))\phi^\top(s)] \quad \text{and}$$
$$b := \frac{1}{N}\mathbb{E}[\phi(s)(\sum_{i\in\mathcal{N}} r^i(s,a) - J_\pi)], \tag{5}$$

where J_π is defined in Eq. (1). The expectations in Eq. (5) are taken over the steady state distribution induced by the given joint policy, which is guaranteed to exist due to Assumption 1, stationary action

[2] For simplicity, we present TD(0) in our paper, the algorithm and theoretical results can be generalized to TD(λ) straightforwardly.

Algorithm 1: Decentralized TD Learning with periodic local TD-update steps

Input : Initial state s_0, $\pi = \{\pi^i | i \in \mathcal{N}\}$, feature map ϕ, initial parameters $\{w^i_{0,0}, \mu^i_{0,0} | i \in \mathcal{N}\}$, step size β, communication round number L, local step number K

for $l = 0, \cdots, L-1$ **do**
  $s_{l,0} = s_{l-1,K}$ (when $l = 0$ and $k = 0$, $s_{l,k} = s_0$);
  for $k = 0, \cdots, K-1$ **do**
    for *all* $i \in \mathcal{N}$ **do in parallel**
      Execute action $a^i_{l,k} \sim \pi^i(\cdot|s_{l,k})$;
      Observe the state $s_{l,k+1}$ and reward $r^i_{l,k+1}$;
      Update $\delta^i_{l,k} \leftarrow r^i_{l,k+1} - \mu^i_{l,k} + \phi(s_{l,k+1})^T w^i_{l,k} - \phi(s_{l,k})^T w^i_{l,k}$;
      Update $\mu^i_{l,k+1} \leftarrow \beta r^i_{l,k+1} + (1-\beta)\mu^i_{l,k}$;
      Local TD-update Step: $w^i_{l,k+1} \leftarrow w^i_{l,k} + \beta \delta^i_{l,k} \cdot \phi(s_{l,k})$;
    end
  end
  for *all* $i \in \mathcal{N}$ **do in parallel**
    Consensus Update: $w^i_{l+1,0} \leftarrow \sum_{j \in \mathcal{N}_i} A(i,j) \cdot w^j_{l,K}$;
  end
end

Output: $\{w^i_{L,0} | i \in \mathcal{N}\}$

policy $a \sim \pi(\cdot|s)$ and state transition probability $s' \sim P(\cdot|s,a)$. Furthermore, we define

$$w^* = -\Psi^{-1} b, \tag{6}$$

where the invertibility is due to Ψ being negative definite [7, 19, 28]. Consequently, $\forall s, \forall k \geq \tau(\beta)$, we define mixing time $\tau(\beta)$ as the time index k that satisfies the following relationship:

$$\|\Psi - \mathbb{E}[(\phi(s_{k+1}) - \phi(s_k))\phi^\top(s) | s_0 = s]\| \leq \beta, \tag{7}$$

where the expectation is taken over appropriate distributions. We note that under the Assumption 1, by [12, Theorem 4.9], the Markov chain mixes at a geometric rate, which implies $\tau(\beta) = O(\log \frac{1}{\beta})$.

5.1 Supporting Lemmas

Before presenting our main theorem, we introduce two useful lemmas. Our strategy of convergence analysis is to divide the convergence error into two parts. They are the consensus error, which is defined as the agent's parameters deviation from the average parameter, and convergence error of the average parameter to the solution of the ODE in Eq. (6).

First, we define the average of the parameters to be $\bar{w}_{l,k} = \frac{1}{N}\sum_{i \in \mathcal{N}} w^i_{l,k}$ for any communication round $l \in \{0, \cdots, L-1\}$ and local step $k \in \{0, \cdots, K-1\}$ and similarly $\bar{\mu}_{l,k} = \frac{1}{N}\sum_{i=1}^N \mu^i_{l,k}$. Then, we define the consensus error for agent i as:

$$Q^i_{l,k} := w^i_{l,k} - \bar{w}_{l,k} \tag{8}$$

and the matrix form is $Q_{l,k} = [Q^1_{l,k}, \cdots, Q^N_{l,k}] \in \mathbb{R}^{n \times N}$.

We provide an upper bound for the consensus error generated by Algorithm 1 in the following lemma.

Lemma 1. *Suppose that Assumptions 2–4 hold. For the consensus error generated by Algorithm 1, if* $\beta K \leq \min\{\frac{1}{2}, \frac{\eta^{N-1}}{4(1-\eta^{N-1})}\}$, *it then holds that*

$$\|Q_{L,0}\| \leq \kappa_1 \rho^L \|Q_{0,0}\| + \frac{\kappa_2 \beta K}{1-\rho}, \tag{9}$$

where $\kappa_1 = \frac{2N^2(1+\eta^{-(N-1)})}{1-\eta^{N-1}}$, $\kappa_2 = 8(1+\eta^{-(N-1)})N^{\frac{5}{2}} r_{\max}$ *and* $\rho := (1+4\beta K)(1-\eta^{N-1})$. *By the condition on* βK, *we have* $0 < \rho < 1$.

The first term in Lemma 1 shows that even if the parameters are not set to be the same initially, the effect of the initial consensus error will vanish exponentially fast as the round of communication L goes to infinity. The second term is linear with respect to βK, which resembles the constant term in optimization using stochastic gradient descent (SGD) with constant step-sizes. This product term dictates the consensus error and the error level that the algorithm converges to, see discussion on Figure 2b for more details. Next, we provide a lemma that characterizes the convergence of the average parameter $\bar{w}_{l,k}$ to the TD fixed point defined in Eq. (6).

Lemma 2. *Suppose Assumptions 1-4 hold. For the w-parameters generated by Algorithm 1, we have following result for the average of the w-parameters:*

$$\begin{aligned}
&\mathbb{E}[\|\bar{w}_{L,0} - w^*\|^2] \\
\leq& c_2(1-c_1\beta)^{KL-\tau(\beta)}\left(\sqrt{\|\bar{w}_{0,0} - w^*\|^2 + (\bar{\mu}_{0,0} - J_\pi)^2} \right. \\
&\left. + \frac{r_{\max}}{3}\right)^2 + c_3 \beta \tau(\beta),
\end{aligned} \tag{10}$$

where $c_1, c_2, c_3 > 0$ *are constants that are independent of step-size* β, *local TD-update step* K *and communication round* L; *and* $\tau(\beta) = O(\log \frac{1}{\beta})$ *is the mixing time. The specified expressions of the constants* c_1, c_2, *and* c_3 *can be found in supplementary material.*

The average parameter $\bar{w}_{L,0} = \frac{1}{N}\sum_{i \in \mathcal{N}} w^i_{L,0}$ corresponds to the updates after $K \times L$ samples and L communication rounds. Lemma 2 shows that $\bar{w}_{L,0}$ converges to solution of the ODE with the rate given by the right-hand-side (RHS) of Eq. (10).

5.2 Main Results

Now, we state the main convergence result of Algorithm 1:

Theorem 1. *Suppose that Assumptions 1-4 hold. For the given policy, consider the output parameters* $\{w^i_{L,0} | i \in \mathcal{N}\}$ *generated by Algorithm 1. If* $\beta K \leq \min\{\frac{1}{2}, \frac{\eta^{N-1}}{4(1-\eta^{N-1})}\}$, *it then follows that:*

$$\begin{aligned}
&\mathbb{E}\left[\sum_{i=1}^N \|w^i_{L,0} - w^*\|^2\right] \leq 2n\left(\kappa_1 \rho^L \|Q_{0,0}\| + \frac{\kappa_2 \beta K}{1-\rho}\right)^2 \\
&+ 2N\left(c_2(1-c_1\beta)^{KL-\tau(\beta)}(\sqrt{\|\bar{w}_{0,0} - w^*\|^2 + (\bar{\mu}_{0,0} - J_\pi)^2} \right. \\
&\left. + \frac{r_{\max}}{3})^2 + c_3\beta\tau(\beta)\right),
\end{aligned} \tag{11}$$

where $\kappa_1, \kappa_2, c_1, c_2, c_3 > 0, 0 < \rho < 1$ *are constants, and* $\bar{w}_{0,0} = \frac{1}{N}\sum_{i \in \mathcal{N}} w^i_{0,0}$, $\bar{\mu}_{0,0} = \frac{1}{N}\sum_{i \in \mathcal{N}} \mu^i_{0,0}$ *and* $Q_{0,0}$ *is the initial consensus*

error defined in Eq. (8). Furthermore, by letting

$$\beta=\Theta(\epsilon\log^{-1}(1/\epsilon)), K=\Theta(1/\epsilon^{1/2}\log(1/\epsilon)), L=\Theta(1/\epsilon^{1/2}\log(1/\epsilon)),$$

we have $\mathbb{E}[\sum_{i=1}^{N}\|w_{L,0}^{i}-w^{*}\|^{2}] = O(\epsilon)$. *The sample complexity is* $KL = O(1/\epsilon\log^{2}(1/\epsilon))$ *and the communication complexity is* $L = O(1/\epsilon^{1/2}\log(1/\epsilon))$.

Note that due to the use of a double-loop structure in Algorithm 1, the parameter $w_{L,0}^{i}$ of agent i corresponds to the result after $K\times L$ samples. We remark that to the best of our knowledge, the state-of-the-art sample complexity for the average reward RL in single agent setting is $O((1/\epsilon)\log^{2}(1/\epsilon))$ [24]. The sample complexity of our algorithm in decentralized multi-agent setting, *matches this sample complexity in the single-agent setting.*

5.3 Discussion

In this section, we provide a comparison of the proposed local TD-update step approach with vanilla and batching approaches in terms of both sample and communication complexities.

1) Sample complexity in comparison with single agent setting: The sample complexity of our algorithm matches the state-of-the-art sample complexity in the single-agent setting. Also, compared to the single-agent discounted reward policy evaluation [32] (a batching method) and its multi-agent counterpart [4], the sample complexity of local TD-update only differs by a log factor. We note that, in [4, 7, 32], the algorithms are complete actor-critic algorithms. Thus, we only compare our results with their policy evaluation counterparts(i.e., critic steps).

2) Communication and sample complexity in comparison with vanilla approach: In the local TD-update algorithm, between consecutive communication rounds, the number of local TD-update steps for each agent can be $K = O(1/\epsilon^{1/2}\log(1/\epsilon))$. This improved the communication complexity of vanilla distributed TD algorithms [5, 6, 35] by a factor of $K = O(1/\epsilon^{1/2}\log(1/\epsilon))$. The communication complexity of the local TD-update is $L = O(1/\epsilon^{1/2}\log(1/\epsilon))$. In terms of sample complexity, both approaches require a sample complexity of $O(1/\epsilon\log^{2}(1/\epsilon))$. This is because as we set local step $K = 1$ of local TD approach, it reduces to the vanilla approach.

3) Communication and sample complexities in comparison with batching approach: It is worth noting that *"batching"* [7] is another natural TD learning approach that can achieve infrequent communication among agents via locally updating value function parameters using a batch of $M(\geq 1)$ samples, then performing consensus. Specifically, instead of repeatedly updating w^{i} for each sample locally as in Line 10 in Algorithm 1, at each communication round $l \in \{0,\cdots,L-1\}$, the batching approach performs the following update:

$$\tilde{w}_{l}^{i} \leftarrow w_{l}^{i} + \frac{1}{M}\sum_{\tau=0}^{M-1}\delta_{l,\tau}^{i}(w_{l}^{i})\cdot\phi(s_{l,\tau}),$$

which is followed by a consensus update same as Line 12 in Algorithm 1 for $\{\tilde{w}_{l}^{i}\}_{i=1}^{N}$. The full algorithm description of the batching approach can be found in [7, Algorithm 1]. The key difference between batching and local TD-update approaches is that the w-parameters are updated repeatedly with each sample in local TD-update, whereas in batching, the w-parameters are updated *only once* through a batch of samples.

Table 1: Comparison of sample and communication complexities.

Approaches	Sample Complexity	Communication Complexity
Vanilla	$O(1/\epsilon\log^{2}(1/\epsilon))$	$O(1/\epsilon\log^{2}(1/\epsilon))$
Batching	$O(1/\epsilon^{3/2}\log(1/\epsilon))$	$O(1/\epsilon^{1/2}\log(1/\epsilon))$
Local TD	$O(1/\epsilon\log^{2}(1/\epsilon))$	$O(1/\epsilon^{1/2}\log(1/\epsilon))$

Under the average reward setting, the local TD-update approach achieves the same communication complexity. However, the local TD-update approach outperforms the batching approach in terms of sample complexity. Specifically, the sample complexity upper bound of the local TD-update approach is $O(1/\epsilon\log^{2}(1/\epsilon))$. In contrast, the sample complexity of the batching approach is $O(1/\epsilon^{3/2}\log(1/\epsilon))$, which is worse than that of the local TD-update approach by a factor of $O(1/\epsilon^{1/2}/\log(1/\epsilon))$.

To conclude the comparisons, we list the sample and communication complexities of different approaches in Table 1.

6 EXPERIMENTAL RESULTS

In this section, we conduct numerical experiments to compare our proposed algorithm, TD learning with local steps, with vanilla TD learning [5, 6, 35] and the batch TD learning [4, 7] in both synthetic settings as in [35] and cooperative navigation tasks as in [15].

6.1 Performance with Synthetic Experiments

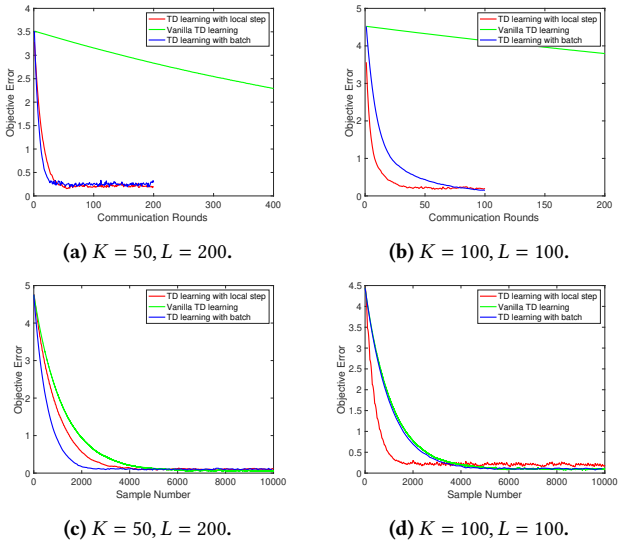
1) Synthetic Experiment Setup: We consider the same setting as in Section 6.1 of [35]. There are $N = 20$ agents, each of which has a binary-valued action space, i.e., $\mathcal{A}^{i} = \{0,1\}$ for all $i \in \mathcal{N}$. There are $|\mathcal{S}| = 10$ states. The entries in the transition matrix are uniformly sampled from the interval $[0,1]$ and normalized to be stochastic. For each agent i and global state action pair (s,a), the reward $r^{i}(s,a)$ is sampled uniformly from $[0,4]$ and the instantaneous rewards $\{r_{t}^{i}\}$ are sampled uniformly within the set $[r^{i}(s,a)-0.5, r^{i}(s,a)+0.5]$. The policy considered in the simulation is $\pi^{i}(\cdot|s) = 0.5$ for all $i \in \mathcal{N}$, $s \in \mathcal{S}$. The entries of feature matrix Φ are sampled uniformly at random from $[0,1]$ with feature dimension $n = 5$ and ensured to be full rank and satisfy Assumption 4. In addition, we set each feature vector to be of unit length. The network topology is chosen as a ring network with diagonal elements being 0.4 and off-diagonal elements being 0.3. The simulation results are averaged over 10 trials. We choose the step sizes for our algorithm to be 0.005, vanilla TD to be 0.1, and batch TD to be 0.1. We note that these step sizes are chosen to be best for the corresponding algorithms.

The objective error is defined as the normalized version of convergence term (LHS of Eq. (11)), i.e., the sample mean errors divided

by the number of agents N and the dimension number n:

$$\text{Objective Error} := \text{sample average of } \frac{\sqrt{\sum_{i=1}^{N} \|w_{l,k}^i - w^*\|^2}}{nN} \text{ for 10 trials.}$$

We remark that due to the fact that the transition matrix is not dependent on joint action, the steady state distribution can be computed and so is the value of w^*, whose definition is in Eq. (6).



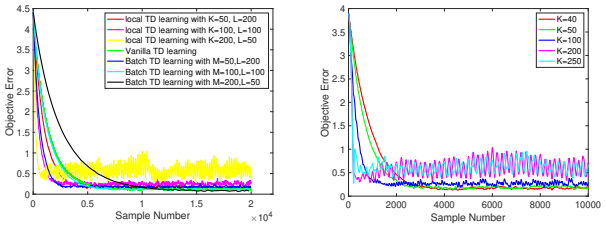
(a) $K = 50, L = 200$. **(b)** $K = 100, L = 100$.

(c) $K = 50, L = 200$. **(d)** $K = 100, L = 100$.

Figure 1: Convergence with respect to the number of communication rounds and samples.

2) Convergence Performance: In Fig. 1, the y-axis is the normalized convergence error of the LHS of Eq. (11) and the x-axes are the numbers of communication rounds in Figure 1a,1b and sample numbers in Figs. 1c and 1d. For fair comparisons between the local TD-update and batching approaches, we keep the local TD-update step number and batch size to be the same for the majority of the comparisons except for Fig. 2a, where we compare the results for various local TD-update step numbers and batch sizes.

In Fig. 1a, we illustrate the convergence results with respect to the communication rounds for all three algorithms, where the local TD-update step $K = 50$ for the local TD-update approach and the batch size is 50 for batch algorithm. Under such a setting, both local TD-update and batched TD algorithms perform consensus communication every 50 samples. We can see that within 200 communication rounds, both local TD-update and batching algorithms converge to a very similar error level, yet the vanilla TD algorithm does not converge even after 400 rounds of communication. Between local TD-update and batching, both algorithms perform similarly, which means similar communication rounds to converge. In Fig. 1b, when local TD-update step $K = 100$ and the batch size is 100, the local TD-update approach requires the least amount of communication rounds to converge compared to the batching approach. On the other hand, local TD-update again performs significantly better compared to vanilla TD. In Fig. 1d and 1c, we illustrate the corresponding convergence results with respect



(a) Different (K, M, L) values. **(b) The impacts of K.**

Figure 2: Convergence comparisons with different settings of (K, L) and the impact of local TD-update steps K on convergence performance.

to the number of samples. We can see that vanilla TD eventually converges but requires consensus operation at every sample. Fig. 1 verifies the theoretical analysis that allowing local TD-update steps does reduce the number of communication rounds compared to vanilla TD. In addition, the communication rounds of local TD-update algorithm is similar in the setting of Fig. 1a and significantly better in the setting of Fig. 1b.

In addition, we compare local TD-update approach under different number of local TD-update steps K and communication rounds L with batching approach under different batch sizes M and communication rounds L in Fig. 2a. In general, the local TD-update approach converges faster than the batching approach, but with a slightly larger objective error. As the number of local TD-update steps increases, the convergence speed of the local TD-update approach converges also increases, but the objective error becomes larger. This verifies the "agent-drift" phenomenon. In contrast, as the batch size increases, the convergence speed becomes slower, and the objective error continues to improve.

3) Impacts of the Number of Local TD-Updates: Next, we further investigate the effect of the number of local TD-update steps on the convergence of the local TD-update approaches and the agent-drift phenomenon. In Fig. 2b, we vary the number of local steps from $K = 40$ to $K = 250$. There are two interesting observations from our experiments. First, the initial dropping of objective error increases as the number of local TD-update steps increases. For example, when $K = 100$ or larger, the curves drop much more rapidly in the beginning compared to the curves with a smaller K. Second, the objective error floor increases as the number of local steps increases. For example, when $K \leq 100$, the objective error floor is relatively low and stable. However, as K increases to 200 or 250, the objective error floor also increases with a larger oscillation magnitude. This observation is consistent with our theoretical analysis in Lemma 1, where the second term on the RHS of Eq. (9) is proportional to the product of step size β and local TD-update step K. This term indicates that the objective error will only converge to neighborhood of zero, whose size depends on βK. As a result, for a larger K-value, the objective error will oscillate with a larger magnitude. This is similar to the constant error term in the convergence of the dencentralized SGD method [17]. Also, the agent-drift phenomenon worsens as the number of local TD-update steps increases, which can be seen by the result of $K \geq 200$ in Fig. 2b. To summarize, under a fixed step size, more local TD-update steps

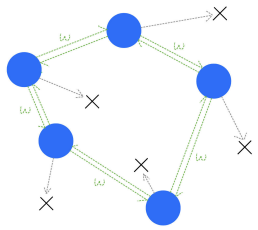


Figure 3: A cooperative navigation task.

improve the initial convergence speed, but will eventually result in a larger objective error floor.

6.2 Performance with Cooperative Navigation

As illustrated in Fig. 3, in the cooperative navigation task [15, 35], the agents (blue circles) are trained to cover the landmarks (crosses). Agents observe positions of all other agents and all landmarks and collaboratively cover the landmarks while avoiding collisions. The rewards for agents are defined through the proximity to the nearest landmarks. Unlike the synthetic experiments, the fixed point of the corresponding ODE as in Eq. (6) is difficult to compute. Thus, we use the mean squared Bellman error(MSBE) as the performance metric. Due to space limitation, we relegate some experimental results to our online technical report [8], including discussions on various network typologies, local TD-update steps, batch sizes, step sizes, and consensus error metrics.

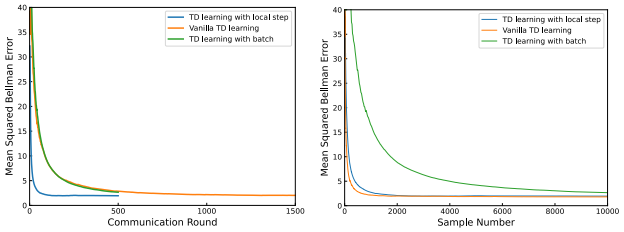
1) Experiment Setup and Performance Metrics: We consider a cooperative navigation task that is adapted from one of the multi-agent environments [15]. There are $N = 9$ agents in total, and the goal is to cover 9 landmarks collaboratively. Each agent chooses from the action space $\mathcal{A}^i$ ={no action, move left, move right, move down, move up} based on the given policy π. The policy considered in the simulation is $\pi^i(\cdot|s) = 0.2$ for all actions and $i \in \mathcal{N}$, $s \in \mathcal{S}$, i.e. uniformly random policy. The local rewards are given by the distance between the agents and the nearest goal landmarks. However, if the agents collide with each other, a penalty will incur. The agents are trained to cover landmarks and reach the destination, while avoiding to collide with other agents, and the entire learning process is fully decentralized. The feature dimension here is $n = 36$, which includes all agents' self positions, landmark relative positions, and other agent relative positions. We choose step sizes for the TD-update and the vanilla TD approaches to be both 0.1. We note that such step sizes are chosen for the best performance for the corresponding algorithms.

As mentioned earlier, we adopt the mean squared Bellman error (MSBE) as our performance metric. Given w-parameters and samples (s_k, s_{k+1}), the empirical squared Bellman error (SBE) of the κ-th sample is defined as:

$$\begin{aligned}&\text{SBE}\left(\{w_\kappa^i\}_{i=1}^N, s_\kappa, s_{\kappa+1}\right)\\ &:= \frac{1}{N}\sum_{i\in\mathcal{N}}\left(\phi(s_\kappa)^T w_\kappa^i + \bar{\mu}_\kappa - \bar{r}_\kappa - \phi(s_{\kappa+1})^T w_\kappa^i\right)^2,\end{aligned}$$

where $\bar{r}_\kappa = \frac{1}{N}\sum_{i\in\mathcal{N}} r_\kappa^i$ and $\bar{\mu}_\kappa = \frac{1}{N}\sum_{i\in\mathcal{N}} \mu_\kappa^i$. Then, MSBE up to the k-th sample is defined as the average of SBEs over the history, which is as follows:

$$\text{MSBE} := \frac{1}{k}\sum_{\kappa=1}^{k}\text{SBE}\left(\{w_\kappa^i\}_{i=1}^N, s_\kappa, s_{\kappa+1}\right).$$



(a) Bellman error with respect to communication rounds. **(b) Bellman error with respect to the number of samples.**

Figure 4: Convergence in terms of the number of communication rounds and training samples.

2) Convergence Performance: In Fig. 4a and 4, we illustrate the results of MSBEs with respect to the number of communication rounds and training samples, where $N = 9$ agents are connected through an Erdos-Renyi (ER) network. We set the number of local TD-update steps and the batch size both to be 20 for the local TD-update and batching approaches, respectively. Similar to the synthetic experiments, all algorithms converge to similar levels of MSBE as shown in Fig. 4. This again verifies our theoretical analysis that allowing local TD-update steps and performing infrequent communications do not affect convergence. Moreover, in this setting, the local TD-update algorithm converges much faster in terms of the number of communication rounds. Specifically, in Fig. 4a, the local TD-update algorithm requires roughly 250 rounds of communication to converge, while both the batching and vanilla TD algorithms perform similarly and require more than 500 rounds of communication to converge.

7 CONCLUSION

In this paper, we investigated the question of whether the local TD-update approach can achieve low sample and communication complexities for multi-agent reinforcement learning policy evaluation (MARL-PE) under the average reward setting and, if so, how is the performance in comparison with other approaches under the average reward setting. Our theoretical analysis and experimental results show that the local TD-update approach can significantly lower the communication complexity compared to the vanilla TD learning. In addition, our theoretical analysis also shows that the number of local TD-update steps can be as large as $K = O(1/\epsilon^{1/2}\log(1/\epsilon))$ to converge to an ϵ-neighborhood of the solution of the corresponding ODE for MARL-PE. Compared with the batching approach for solving MARL-PE under average reward, the local TD-update approach achieves the same communication complexity as that of the batching approach, while enjoying a better sample complexity by a factor of $O(1/\epsilon^{1/2})$ than that of the batching approach. Our experimental results also verify our theoretical findings in both synthetic and real-world data settings.

Algorithm 2: Single Agent TD(0) Learning in Average Reward Setting

Input : Initial state s_0, π, feature map ϕ, initial parameters w_0, μ_0, step size β, traning iteration T

for $t = 0, \cdots, T-1$ **do**
  Execute action $a_t \sim \pi(\cdot|s_t)$;
  Observe the state s_{t+1} and reward r_{t+1};
  Update $\delta_t \leftarrow r_{t+1} - \mu_t + \phi(s_{t+1})^T w_t - \phi(s_t)^T w_t$;
  Update $\mu_{t+1} \leftarrow \beta r_{t+1} + (1-\beta)\mu_t$;
  TD Step: $w_{t+1} \leftarrow w_t + \beta\delta_t \cdot \phi(s_t)$;
end

Output : w_T

A SINGLE-AGENT POLICY EVALUATION CONVERGENCE UNDER THE AVERAGE REWARD SETTING

In this section, we provide the finite-time convergence result for single-agent RL in the average reward setting, as the update of average parameter $\bar{w}$ in Lemma 2 is essentially a centralized single-agent TD learning. The finite time convergence for a more general form of stochastic approximation has been established in [24]. We utilize such results by verifying the conditions in [24].

A.1 Single Agent RL in Average Reward Setting

We first describe the single agent TD(0) algorithm in the average reward setting in Algorithm 2.

Theorem 2. *Suppose $N = 1$ and Assumptions 1-4 hold. For the parameter generated by Algorithm 2, we have following results:*

$$\mathbb{E}[\|w_T - w^*\|^2] \leq c_2(1-c_1\beta)^{T-\tau(\beta)}(\sqrt{\|w_0 - w^*\|^2 + (\mu_0 - J_\pi)^2} + \frac{r_{\max}}{3})^2 + c_3\beta\tau(\beta), \tag{12}$$

where $c_1, c_2, c_3 > 0$ are constants that are independent of step size β and iteration number T; and $\tau(\beta) = O(\log\frac{1}{\beta})$ is the mixing time.

Proof. To apply Theorem 7 in [24], we need to verify the three conditions in [24, Section 2.1]. We have the following notations $Z_t = (s_t, a_t)$ and $r_{t+1} = r(Z_t)$. For TD(0) learning under the average reward setting, we have

$$\mu_{t+1} = \mu_t + \beta(r_{t+1} - \mu_t),$$
$$w_{t+1} = w_t + \beta(r_{t+1} - \mu_t + \phi^T(Z_{t+1})w_t - \phi^T(Z_t)w_t)\phi(Z_t).$$

Equivalently, the matrix form is

$$\begin{pmatrix}\mu_{t+1}\\ w_{t+1}\end{pmatrix} = \begin{pmatrix}\mu_t\\ w_t\end{pmatrix} + \beta\begin{pmatrix}-1 & 0\\ -\phi(Z_t) & \phi(Z_t)(\phi(Z_{t+1}) - \phi(Z_t))^T\end{pmatrix}\cdot\begin{pmatrix}\mu_t\\ w_t\end{pmatrix} + \beta\begin{pmatrix}r_{t+1}\\ \phi(Z_t)r_{t+1}\end{pmatrix}. \tag{13}$$

So the corresponding ODE can be written as:

$$\begin{pmatrix}\dot{\mu}\\ \dot{w}\end{pmatrix} = \begin{pmatrix}-1 & 0\\ -\Phi^T D^s\mathbf{1} & \Phi^T D^s(P^\pi - I)\Phi\end{pmatrix}\cdot\begin{pmatrix}\mu\\ w\end{pmatrix} + \begin{pmatrix}J_\pi\\ \Phi^T D^s\bar{R}\end{pmatrix}, \tag{14}$$

where P^π is the state transition matrix induced by the policy π, $D^s = \text{diag}(d(s_1), \cdots, d(s_{|\mathcal{S}|}))$ and $\bar{R} := [\bar{R}(s), s \in \mathcal{S}]^T$, where $\bar{R}(s) = \sum_a \pi(a|s)r(s,a)$. Now, using the notation in [24], we have

$$\bar{A} = \tilde{A} = \begin{pmatrix}-1 & 0\\ -\Phi^T D^s\mathbf{1} & \Phi^T D^s(P^\pi - I)\Phi\end{pmatrix}$$

and

$$\tilde{b} = \begin{pmatrix}J_\pi\\ \Phi^T D^s\bar{R}\end{pmatrix}.$$

Next, by centering $\begin{pmatrix}\mu\\ w\end{pmatrix} \leftarrow \begin{pmatrix}\mu\\ w\end{pmatrix} - \begin{pmatrix}J_\pi\\ w^*\end{pmatrix}$ and defining, we have

$$X_k = (Z_k, Z_{k+1})^T,$$
$$A(X_k) = \begin{pmatrix}-1 & 0\\ -\phi(Z_t) & \phi(Z_t)(\phi(Z_{t+1}) - \phi(Z_t))^T\end{pmatrix},$$
$$b(X_k) = \begin{pmatrix}r_{t+1}\\ \phi(Z_t)r_{t+1}\end{pmatrix} - A(X_k)\cdot\begin{pmatrix}J_\pi\\ w^*\end{pmatrix},$$
$$\bar{b} = 0.$$

Note that $\bar{A}\begin{pmatrix}J_\pi\\ w^*\end{pmatrix} = \tilde{b}$. Next, consider the following conditions:

- **Condition 1:** Note that

$$\begin{aligned}&||E[b(X_k|X_0 = (Z_0, Z_1) = (z_0, z_1))]||\\ =&||\sum_i (P(Z_k = i|(Z_0, Z_1) = (z_0, z_1)) - d(i))\\ &\cdot\left(\begin{pmatrix}\bar{r}(i)\\ \phi(i)\bar{r}(i)\end{pmatrix} - \begin{pmatrix}-1 & 0\\ -\phi(i) & \phi(i)(\sum_j p_{ij}^\pi\phi(j) - \phi(i))^T\end{pmatrix}\begin{pmatrix}J_\pi\\ w^*\end{pmatrix}\right)||\\ \leq&||\sum_i (P(Z_k = i|(Z_0, Z_1) = (z_0, z_1)) - d(i))|| \cdot b_{\max},\end{aligned}$$

  where $b_{\max} = 2(r_{\max} + J_\pi) + 2w^*$, where we used Assumption 4 and

$$\begin{aligned}&||\bar{A} - E[A(X_k)|X_0 = (Z_0, Z_1) = (z_0, z_1)]||\\ =&||\sum_i (P(Z_k = i|(Z_0, Z_1) = (z_0, z_1)) - d(i))\\ &\cdot\begin{pmatrix}-1 & 0\\ -\phi(i) & \phi(i)(\sum_j p_{ij}^\pi\phi(j) - \phi(i))^T\end{pmatrix}||\\ \leq& 4||\sum_i (P(Z_k = i|(Z_0, Z_1) = (z_0, z_1)) - d(i))||.\end{aligned}$$

  Since $\{Z_k\}$ is a finite state, aperiodic and irreducible Markov chain, it has a geometric mixing rate, so Assumption 1 holds.
- **Condition 2:** By Assumption 4, $\max_{i\in\mathcal{S}} ||\phi(i)|| \leq 1 < \infty$ and $\max_{i\in\mathcal{S}\times\mathcal{A}} r(i) = r_{\max}$, it implies

$$||A(X_k)|| = ||\begin{pmatrix}-1 & 0\\ -\phi(i) & \phi(i)(\phi(j) - \phi(i))^T\end{pmatrix}|| \leq 4.$$

  Hence it is bounded. To normalize, we can set $A(i) \leftarrow \frac{A(i)}{4}$ and $b(i) \leftarrow \frac{b(i)}{4}$ to ensure the $||\bar{A}|| \leq 1$.

- **Condition 3:** By the standard assumptions on the feature vectors in [28], we have that (1) Φ is full rank; (2) for every $\nu \in R^n$, $\Phi\nu \neq \mathbf{1}$. This ensures that real parts of all eigenvalues of $\bar{A}$ are strictly negative.

As a result, by directly applying Theorem 7 of [24], we have

$$\begin{aligned}&\mathbb{E}[\|w_T - w^*\|^2 + |\mu_T - J_\pi|^2]\\ \leq& c_2(1-c_1\beta)^{T-\tau(\beta)}(\sqrt{\|w_0 - w^*\|^2 + (\mu_0 - J_\pi)^2}\\ &+ \frac{r_{\max}}{3})^2 + c_3\beta\tau(\beta),\end{aligned}$$

from which the result in Eq. (12) follows. □

A.2 Details of the Constants c_1, c_2, and c_3 in Lemma 2

The average parameter is signaled by the average of the rewards, i.e. $\bar{r}_{l,k} = \frac{1}{N}\sum_{i\in\mathcal{N}} r^i_{l,k}$ during both the local TD-update and consensus steps. Therefore, the method updates similar to a centralized TD learning in a single-agent setting. By applying Theorem 2 for the average parameter, we have following results:

$$\begin{aligned}\mathbb{E}[||\bar{w}_{L,0} - w^*||^2] \leq& c_2(1-c_1\beta)^{KL-\tau(\beta)}(\sqrt{||\bar{w}_0 - w^*||^2 + (\mu_0 - J_\pi)^2}\\ &+ \frac{r_{\max}}{3})^2 + c_3\beta\tau(\beta),\end{aligned}$$

where $\tau(\beta)$ is a mixing time. Under Assumption 1, $\tau(\beta) = O(\log\frac{1}{\beta})$. To specify the constants c_1, c_2, c_3, recall the definition of Ψ in Eq. (5), which is negative definite [28]. Further, define

$$\tilde{\Psi} := \begin{pmatrix} -1 & 0 \\ -\Phi^T D^s \mathbf{1} & \Psi \end{pmatrix},$$

where $D^s = \text{diag}(d(s_1), \cdots, d(s_{|\mathcal{S}|}))$ and Φ is the feature matrix. It is easy to see the lower diagonal block matrix $\tilde{\Psi}$ is a Hurwitz matrix due to the fact that both diagonal blocks are Hurwitz. Therefore, we have a symmetric matrix $U > 0$ [24] such that

$$\tilde{\Psi}^T U + U\tilde{\Psi} + I = 0,$$

which is referred to as the Lyapunov equation. For symmetric matrix U, there exist the largest and smallest eigenvalues $\lambda_{\max}$ and $\lambda_{\min}$, respectively. In addition, $\lambda_{\max}$ and $\lambda_{\min}$ are both positive. As a result, by [24, Theorem 7], the constants are:

$$\begin{aligned}c_1 &= \frac{0.9}{\lambda_{\max}},\\ c_2 &= 2.25\frac{\lambda_{\max}}{\lambda_{\min}},\\ c_3 &= \frac{2\lambda^2_{\max}(r^2_{\max} + 55(1+r_{\max})^3)}{0.9\lambda_{\min}}.\end{aligned}$$

B COOPERATIVE NAVIGATION TASK

In this section, we provide further experimental details on cooperative navigation task in addition to Section 6.2. Moreover, we use consensus error as another performance metric, which is defined as following

$$\text{CE}\left(\{w^i_k\}_{i=1}^N\right) := \frac{1}{N}\sum_{i\in\mathcal{N}}\left\|w^i_k - \overline{w}_k\right\|^2.$$

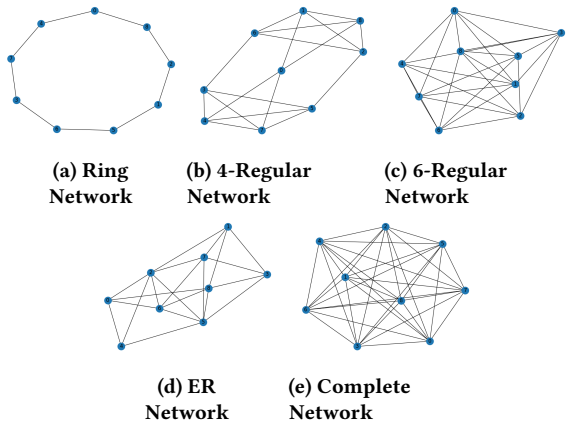


Figure 5: Network Topology

B.1 Network Topology

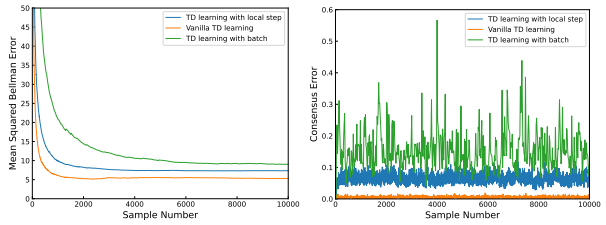
We compare all algorithms under five different network typologies. These are ring network, 4-regular network, 6-regular network, Erdos-Renyi(ER) network with 0.5 connection probability, and fully connected network. These network topologies are illustrated in Figure 5. For simplicity, the local aggregation is the average of neighboring nodes for all networks.

B.2 Convergence Performance

First, we show the empirical convergence performances of all algorithms in terms of MSBEs and CEs. Here we choose the number of local steps to be $K = 20$ for local TD-update algorithm, the batch size to be 20 for batch TD algorithm. For all algorithms, we set the total sample number to be 10000. The comparisons among the algorithms are shown in Figure 6-10 over various network topologies. Left columns of the Figure 6-10 demonstrate the mean squared Bellman error(MSBE), and right columns demonstrate the consensus error(CE).

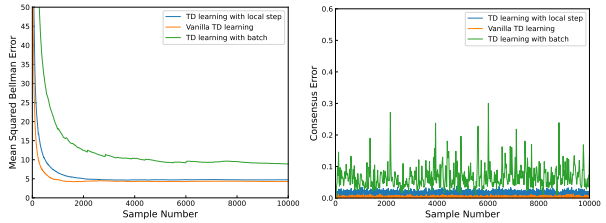
We can see that all algorithms converge. More specifically, in terms of MSBE, the error floor of vanilla TD is the lowest, our proposed local TD-update approach is the second best and batch TD algorithm is the worst with a significant error gap. Similarly for CE, vanilla TD shows the lowest consensus error, our local TD-update approach shows slightly higher error, while batch TD algorithm shows the largest and oscillating consensus error across all network topologies. This verifies our analysis that allowing local steps and performing infrequent communications is feasible and can converge. In this parameter setting, vanilla TD algorithm performs 10000 communication rounds, which is the most, batch TD algorithm performs 1000 communication rounds, while local TD-update algorithm only performs 500 communication rounds. For more details, see discussion on the communication round in Section B.3.

In Figure 11, we present the topology effect on our proposed algorithm. It is, in general, as the network becomes more and more connected the consensus error fluctuates less. Intuitively, with denser network, after local consensus aggregation, the parameter can be closer to the global average of the network.



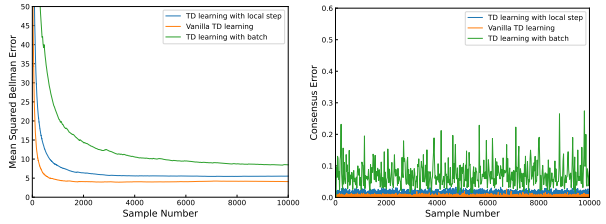
(a) Mean Squared Bellman Error (b) Consensus Error

Figure 6: Comparison among Algorithms in Ring Network



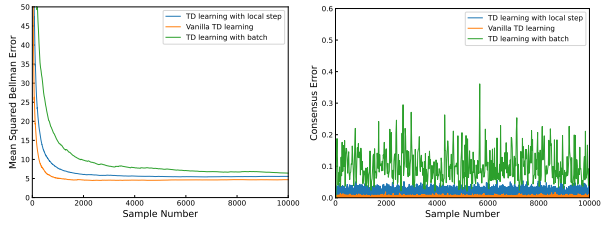
(a) Mean Squared Bellman Error (b) Consensus Error

Figure 7: Comparison among Algorithms in 4-Regular Network



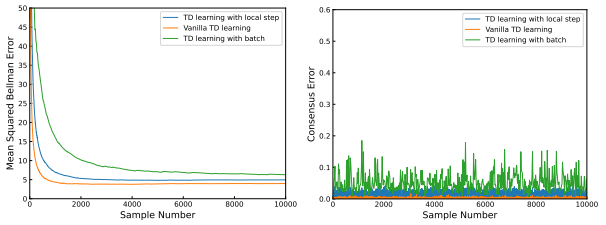
(a) Mean Squared Bellman Error (b) Consensus Error

Figure 8: Comparison among Algorithms in 6-Regular Network



(a) Mean Squared Bellman Error (b) Consensus Error

Figure 9: Comparison among Algorithms in ER Network



(a) Mean Squared Bellman Error (b) Consensus Error

Figure 10: Comparison among Algorithms in Complete Network

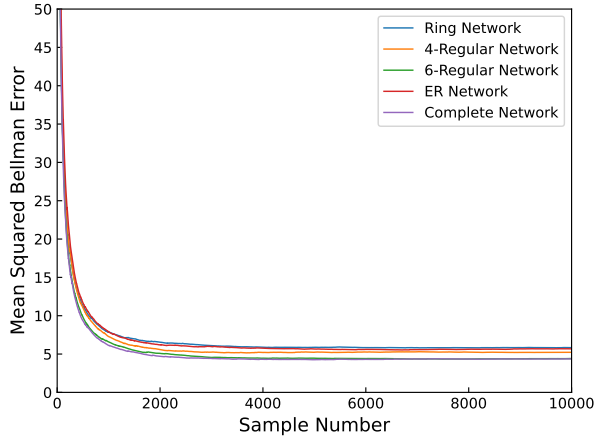


Figure 11: Topology on Local TD Algorithm

In addition, we have compared different pairs of local step and communication round for our proposed algorithm and different pairs of batch size and communication round for batch TD algorithm in Figure 12 for cooperative navigation task. In this setting, all algorithms converge to a similar Bellman error level. However, the convergence for local TD algorithm is faster than batch TD algorithms in all parameter settings. Moreover, with the increase of batch size, batch TD algorithm seems to converge slower while with the increase of local step, local TD algorithm convergence increases in general but not significantly.

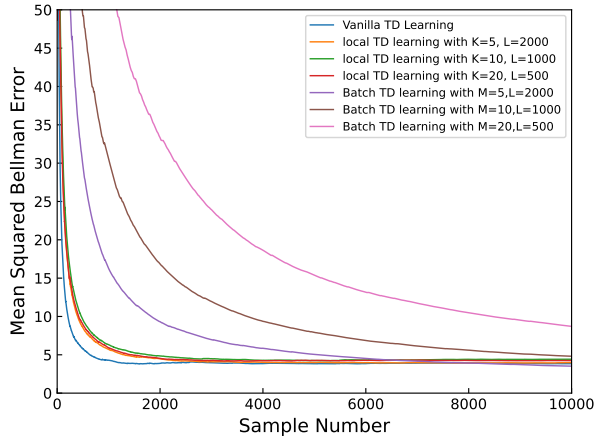


Figure 12: Additional Comparisons

B.3 Convergence Performance With Respect to Communication Rounds

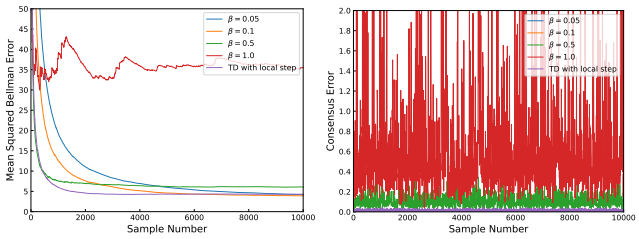
In Figure 13, we provide the convergence results with respect to the communication rounds for all algorithms, where the local step $K = 20$ for local TD-update algorithm and the batch size is 20 for batch TD algorithm. We can see that within 500 communication rounds, local TD-update algorithm converges and requires much less communication round than vanilla TD algorithm, the convergence of which requires more than 1000 communication rounds. On the other hand, local TD-update approach converges to a lower error floor compared to batch TD algorithm. We can observe such empirical results across all network topologies.

B.4 Impacts of the Number of Local Steps on Convergence

Next, we illustrate the effect of the number of local steps K on the convergence for our proposed algorithm. In Figure 14, we vary the number of local steps from $K = 10$ to $K = 200$. The right column is the consensus error of first 2000 samples, which displays a better view. As the number of local steps increases, the mean squared Bellman error converges to a higher error level, on the other hand, the consensus error oscillates more. In summary, larger local steps helps with saving more communication cost while it also result in converging to a higher mean squared Bellman error and a greater fluctuation of the consensus error. This conclusion echoes the results in synthetic experiment in Figure 2b.

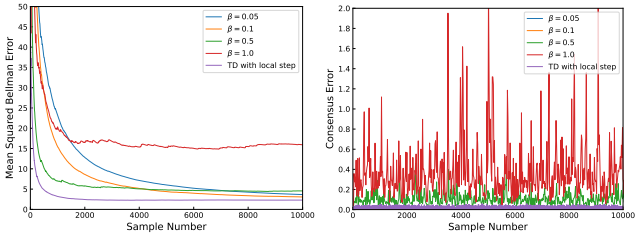
B.5 Impacts of Step Size on Convergence

Here, we illustrate the effect of step size β on the convergence for our proposed algorithm and batch TD algorithm over 4-regular network to shed lights on the choice of step sizes for batch TD algorithm and our proposed local TD algorithm. First of all, Figure 15-17 show the performance of batch TD algorithm over various batch sizes. The purple line shows the performance of our proposed algorithm as a baseline comparison. In general, in terms of mean squared Bellman error, larger step size β leads to faster convergence speed and larger error level. Smaller step size β leads to slower convergence speed, but could eventually converge to a smaller error floor. Also, larger step size β results in a greater consensus error. Thus, we set step size $\beta = 0.1$ and batch size to be 20 for batch TD algorithm.



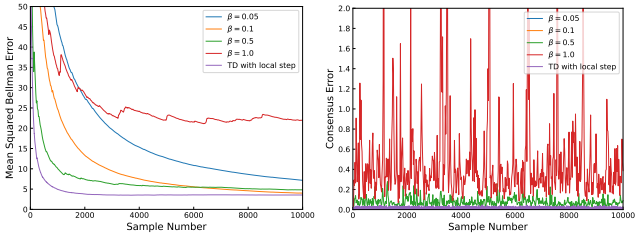
(a) Mean Squared Bellman Error (b) Consensus Error

Figure 15: Comparison among Different step sizes β for TD Learning with Batch Size 5 in 4 Regular Network



(a) Mean Squared Bellman Error (b) Consensus Error

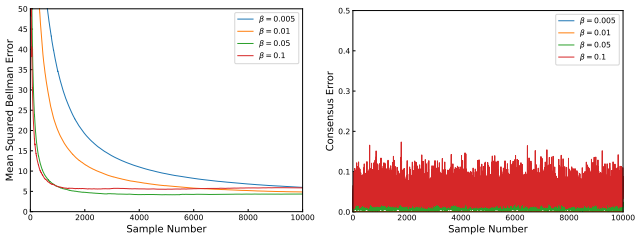
Figure 16: Comparison among Different step sizes β for TD Learning with Batch Size 10 in 4 Regular Network



(a) Mean Squared Bellman Error (b) Consensus Error

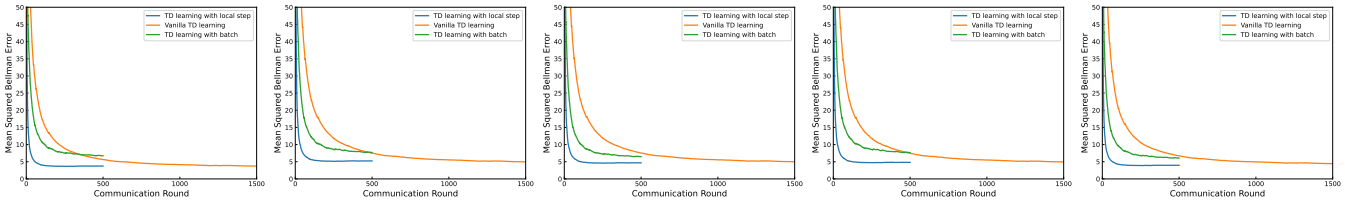
Figure 17: Comparison among Different step size β for TD Learning with Batch Size 20 in 4 Regular Network

In Figure 18-20, we show the effect of step size β on the convergence performance of local TD-update algorithm. Similar to batch TD algorithm, larger step size β leads to faster convergence speed and larger mean squared Bellman and consensus error floor in local TD-update algorithm. However, the error floor differences among different step sizes β are smaller compared to batch TD algorithm. In order to balance among convergence speed, error floor, and communication cost, we decide to use step size $\beta = 0.05$ and local step $K = 20$.



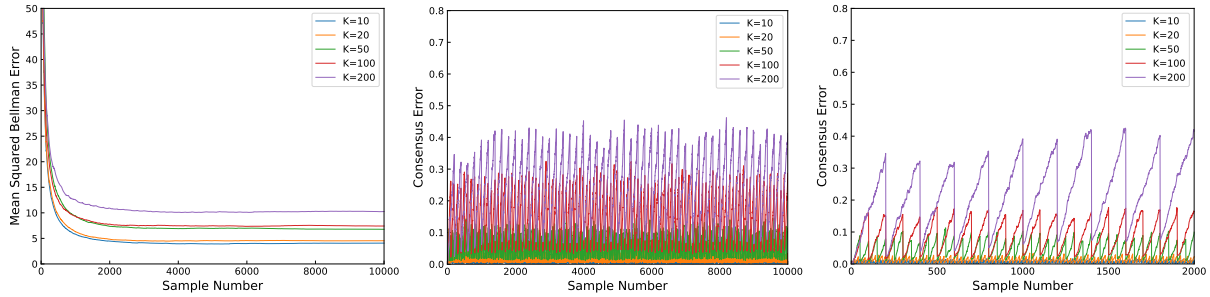
(a) Mean Squared Bellman Error (b) Consensus Error

Figure 18: Comparison among Different step sizes β for TD Learning with Local Step $K = 10$ in 4 Regular Network



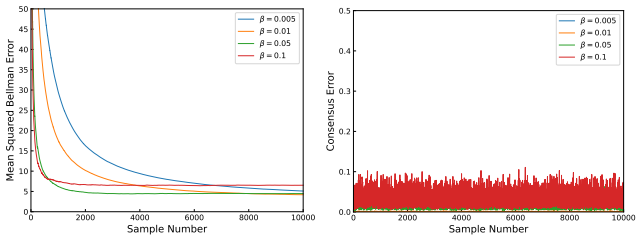
(a) ring network (b) 4-regular network (c) 6-regular network (d) ER network (e) complete network

Figure 13: Convergence Respect to Communication Rounds



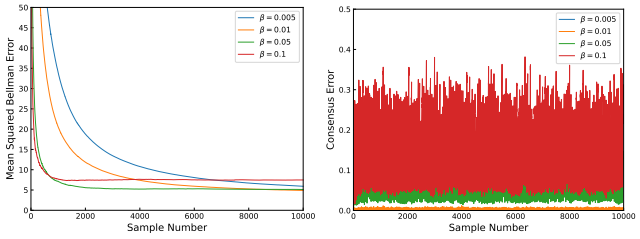
(a) Mean Squared Bellman Error (b) Consensus Error (c) Zoomed Consensus Error

Figure 14: Comparison among Different Local Steps in 4 Regular Network



(a) Mean Squared Bellman Error (b) Consensus Error

Figure 19: Comparison among Different step sizes β for TD Learning with Local Step $K = 20$ in 4 Regular Network



(a) Mean Squared Bellman Error (b) Consensus Error

Figure 20: Comparison among Different step sizes β for TD Learning with Local Step $K = 50$ in 4 Regular Network

C PROOFS OF LEMMA AND THEOREM FOR AVERAGE REWARD SETTING

In this section, we provide the derivation for the consensus error. Then, we prove the Lemma 1 and Theorem 1.

C.1 The derivation of consensus error

Within each communication round $0 \le l \le L-1$, the parameter update $w_{l,k}^i$ for agent i at local step k can be written as follows

$$\begin{aligned} & w_{l,k+1}^i \\ &= w_{l,k}^i + \beta\delta_{l,k}^i \cdot \phi(s_{l,k}) \\ &= \left(I + \beta\phi(s_{l,k})[\phi(s_{l,k+1}) - \phi(s_{l,k})]^T\right) w_{l,k}^i + \beta(r_{l,k+1}^i - \mu_{l,k}^i)\phi(s_{l,k}) \\ &= B_{l,k} w_{l,k}^i + c_{l,k}^i \end{aligned} \tag{15}$$

where $B_{l,k} := I+\beta\phi(s_{l,k})[\phi(s_{l,k+1})-\phi(s_{l,k})]^T$ and $c_{l,k}^i := \beta(r_{l,k+1}^i - \mu_{l,k}^i)\phi(s_{l,k})$. Then, from local step 0 to $K-1$, we have

$$w_{l,K}^i = \prod_{k=0}^{K-1} B_{l,k} w_{l,0}^i + \sum_{k=0}^{K-1} \prod_{n=k+1}^{K-1} B_{l,n} c_{l,k}^i.$$

After a consensus update, the parameter for agent i will be

$$\begin{aligned} w_{l+1,0}^i &= \sum_{j\in\mathcal{N}_i} A(i,j) \cdot w_{l,K}^j \\ &= \sum_{j\in\mathcal{N}_i} A(i,j) \cdot \left(\prod_{k=0}^{K-1} B_{l,k} w_{l,0}^j + \sum_{k=0}^{K-1} \prod_{n=k+1}^{K-1} B_{l,n} c_{l,k}^j\right) \\ &= \sum_{j\in\mathcal{N}_i} A(i,j) \cdot \prod_{k=0}^{K-1} B_{l,k} w_{l,0}^j + \sum_{j\in\mathcal{N}_i} A(i,j) \cdot \sum_{k=0}^{K-1} \prod_{n=k+1}^{K-1} B_{l,n} c_{l,k}^j. \end{aligned}$$

The equation above shows the parameter update between two consecutive communication rounds. Now we consider the average dynamics of the parameters across all agents. Recall $\bar{w}_{l,k} = \frac{1}{N}\sum_{i\in\mathcal{N}} w_{l,k}^i$, then within each communication round $0 \le l \le L-1$,

using equation 15 we have

$$\begin{aligned}\bar{w}_{l,k+1} &= \frac{1}{N}\sum_{i\in\mathcal{N}} w^i_{l,k+1} \\ &= \frac{1}{N}\sum_{i\in\mathcal{N}} (B_{l,k}w^i_{l,k} + c^i_{l,k}) \\ &= B_{l,k}\bar{w}_{l,k} + \frac{1}{N}\sum_{i\in\mathcal{N}} c^i_{l,k} \\ &= B_{l,k}\bar{w}_{l,k} + \bar{c}_{l,k} \end{aligned} \tag{16}$$

where $\bar{c}_{l,k} \coloneqq \frac{1}{N}\sum_{i\in\mathcal{N}} c^i_{l,k}$. Hence, the average dynamics from local step 0 to $K-1$ will be

$$\bar{w}_{l,K} = \prod_{k=0}^{K-1} B_{l,k}\bar{w}_{l,0} + \sum_{k=0}^{K-1}\prod_{n=k+1}^{K-1} B_{l,n}\bar{c}_{l,k}.$$

After a consensus update, we have

$$\bar{w}_{l+1,0} = \bar{w}_{l,K}. \tag{17}$$

Note that the equation above means that consensus step will not change the average dynamics and average dynamic will only be updated during local steps.

For an agent $i \in \mathcal{N}$, we consider the consensus error at communication round l and local step k, where $0 \le k \le K-1$ and recall $Q^i_{l,k} = w^i_{l,k} - \bar{w}_{l,k}$. Then, we have

$$\begin{aligned} Q^i_{l,k+1} =& w^i_{l,k+1} - \bar{w}_{l,k+1} \\ =& B_{l,k}w^i_{l,k} + c^i_{l,k} - B_{l,k}\bar{w}_{l,k} - \bar{c}_{l,k} \\ =& B_{l,k}(w^i_{l,k} - \bar{w}_{l,k}) + c^i_{l,k} - \bar{c}_{l,k} \\ =& B_{l,k}Q^i_{l,k} + c^i_{l,k} - \bar{c}_{l,k}. \end{aligned}$$

Then, for the matrix form $Q_{l,k} = [Q^1_{l,k}, \cdots, Q^N_{l,k}] \in R^{d\times N}$, we have

$$Q_{l,k+1} = B_{l,k}Q_{l,k} + C_{l,k}(I - \frac{1}{N}\mathbf{1}\mathbf{1}^T)$$

where $C_{l,k} \coloneqq [c^1_{l,k}\cdots c^N_{l,k}]$ and $\mathbf{1}$ denotes the all-1 column vector. Then, for communication round l, we have

$$Q_{l,K} = \prod_{k=0}^{K-1} B_{l,k}Q_{l,0} + \sum_{t=0}^{K-1}\prod_{\tilde{t}>t}^{K-1} B_{l,\tilde{t}}C_{l,k}(I - \frac{1}{N}\mathbf{1}\mathbf{1}^T)$$

After a consensus update, we have

$$\begin{aligned} w^i_{l+1,0} - \bar{w}_{l,k} &= \sum_{j\in\mathcal{N}_i} A(i,j)w^j_{l,K} - \bar{w}_{l,K} \\ &= \sum_{j\in\mathcal{N}_i} A(i,j)(w^j_{l,K} - \bar{w}_{l,K}) = \sum_{j\in\mathcal{N}_i} A(i,j)Q^j_{l,K}. \end{aligned}$$

As a result, we have

$$\begin{aligned} Q_{l+1,0} &= Q_{l,K}A^T \\ &= \prod_{k=0}^{K-1} B_{l,k}Q_{l,0}A^T + \sum_{t=0}^{K-1}\prod_{\tilde{t}>t}^{K-1} B_{l,\tilde{t}}C_{l,t}(I - \frac{1}{N}\mathbf{1}\mathbf{1}^T)A^T. \end{aligned}$$

After L communication rounds, we have

$$\begin{aligned} Q_{L,0} =& \prod_{l=0}^{L-1}\prod_{k=0}^{K-1} B_{l,k}Q_{0,0}(A^T)^L \\ &+ \sum_{l=0}^{L-1}\prod_{j=1}^{L-1-l}\prod_{k=0}^{K-1} B_{l+j,k}\sum_{t=0}^{K-1}\prod_{\tilde{t}>t}^{K-1} B_{l,\tilde{t}}C_{l,t}(I - \frac{1}{N}\mathbf{1}\mathbf{1}^T)(A^T)^{L-l} \end{aligned}$$

Note that for the second term when $l = L-1$, inside the summation, the summand becomes $\sum_{t=0}^{K-1}\prod_{\tilde{t}>t}^{K-1} B_{L-1,\tilde{t}}C_{L-1,t}(I - \frac{1}{N}\mathbf{1}\mathbf{1}^T)A^T$. In other words, the matrix multiplier in front becomes an identity matrix.

C.2 Proof of Lemma 1

The norm of the consensus error is following

$$\begin{aligned} &||Q_{L,0}|| \\ =&|| \prod_{l=0}^{L-1}\prod_{k=0}^{K-1} B_{l,k}Q_{0,0}(A^T)^L \\ &+ \sum_{l=0}^{L-1}\prod_{j=1}^{L-1-l}\prod_{k=0}^{K-1} B_{l+j,k}\sum_{t=0}^{K-1}\prod_{\tilde{t}>t}^{K-1} B_{l,\tilde{t}}C_{l,t}(I - \frac{1}{N}\mathbf{1}\mathbf{1}^T)(A^T)^{L-l}|| \\ \le&|| \prod_{l=0}^{L-1}\prod_{k=0}^{K-1} B_{l,k}Q_{0,0}(A^T)^L|| \\ &+ || \sum_{l=0}^{L-1}\prod_{j=1}^{L-1-l}\prod_{k=0}^{K-1} B_{l+j,k}\sum_{t=0}^{K-1}\prod_{\tilde{t}>t}^{K-1} B_{l,\tilde{t}}C_{l,t}(I - \frac{1}{N}\mathbf{1}\mathbf{1}^T)(A^T)^{L-l}||. \end{aligned} \tag{18}$$

Before obtaining bounds on the terms of the consensus error in equation 18, we first provide some useful bounds on $B_{l,k}$ and $C_{l,k}$. First, we have

$$\begin{aligned} ||B_{l,k}|| &= ||I + \beta\phi(s_{l,k})[\phi(s_{l,k+1}) - \phi(s_{l,k})]^T|| \\ &\le 1 + \beta||\phi(s_{l,k})||(||\phi(s_{l,k+1})|| + ||\phi(s_{l,k})||) \\ &\le 1 + 2\beta \end{aligned}$$

where the second inequality is due to Assumption 4. Then, we have $||C_{k,l}|| \le 2\beta\sqrt{N}r_{\max}$, where $r_{\max} = \sup_{i,s,a} r^i(s,a)$ by Assumption 2. This is because

$$\begin{aligned} ||C_{k,l}|| &= ||\beta\phi(s_{k,l})\left((r^1_{k,l+1},\cdots,r^N_{l,k+1}) - (\mu^1_{k,l},\cdots,\mu^N_{l,k})\right)|| \\ &\le \beta||\phi(s_{k,l})||\cdot(||(r^1_{k,l+1},\cdots,r^N_{l,k+1})|| + ||(\mu^1_{k,l},\cdots,\mu^N_{l,k})||) \\ &= 2\beta\sqrt{N}r_{\max}. \end{aligned}$$

Next, inspired by [24], we want to use the following bound

$$(1+x)^K \le 1 + 2xK$$

for small x. Note that

$$(1+x)^K|_{x=0} = 1 + 2xK|_{x=0}$$

and when $x \le \frac{\log 2}{K-1}$,

$$\frac{\partial}{\partial x}(1+x)^K = K(1+x)^{K-1} \le Ke^{x(K-1)} \le 2K = \frac{\partial}{\partial x}(1+2xK)$$

where the first inequality is due to the fact $\log(1+x) \le x$ for $x \ge 0$ and the second inequality is due to the fact $x \le \frac{\log 2}{K-1}$. Let $2\beta = x$ and $\beta \le \frac{1}{2K} \le \frac{\log 2}{2(K-1)}$.

For the first term in equation 18, when $\beta \le \frac{1}{2K}$, we have that

$$\begin{aligned}
||\prod_{l=0}^{L-1}\prod_{k=0}^{K-1} B_{l,k} Q_{0,0}(A^T)^L|| &\le ||\prod_{l=0}^{L-1}\prod_{k=0}^{K-1} B_{l,k}|| \cdot ||Q_{0,0}(A^T)^L|| \\
&\le \kappa(1+2\beta)^{KL}(1-\eta^{N-1})^L \\
&\le \kappa(1+4\beta K)^L(1-\eta^{N-1})^L \\
&= \kappa\rho^L
\end{aligned}$$

where we define $\rho := (1+4\beta K)(1-\eta^{N-1})$. When $0 < \beta K < \min\{\frac{1}{2}, \frac{\eta^{N-1}}{4(1-\eta^{N-1})}\}$, we have $0 < \rho < 1$. The second inequality comes from the following two results.

First, consider the case where A is a symmetric matrix for simplicity, then we have

$$\begin{aligned}
||Q_{0,0}A^L_{1,:}|| &= ||Q_{0,0}A^L_{1,:} - Q_{0,0}\frac{1}{N}\mathbf{1}|| \\
&= ||\sum_{i\in\mathcal{N}}(A^L_{1,i} - \frac{1}{N})Q^i_{0,0}|| \\
&\le \sum_{i\in\mathcal{N}} |A^L_{1,i} - \frac{1}{N}| \cdot ||Q^i_{0,0}|| \\
&\le N \cdot 2\frac{1+\eta^{-(N-1)}}{1-\eta^{N-1}}(1-\eta^{N-1})^L \cdot \max_{i\in\mathcal{N}} ||Q^i_{0,0}|| \\
&\le 2N\frac{1+\eta^{-(N-1)}}{1-\eta^{N-1}}(1-\eta^{N-1})^L \cdot ||Q_{0,0}||,
\end{aligned}$$

where the second inequality is from [17] (Proposition 1). Hence, $||Q_{0,0}A^L|| \le 2N^2\frac{1+\eta^{-(N-1)}}{1-\eta^{N-1}}(1-\eta^{N-1})^L \cdot ||Q_{0,0}|| = \kappa_1(1-\eta^{N-1})^L||Q_{0,0}||$, where $\kappa_1 = 2N^2\frac{1+\eta^{-(N-1)}}{1-\eta^{N-1}}$. Second, we have

$$\begin{aligned}
||\prod_{l=0}^{L-1}\prod_{k=0}^{K-1} B_{l,k}|| &\le \prod_{l=0}^{L-1}\prod_{k=0}^{K-1} ||B_{l,k}|| \\
&\le \prod_{l=0}^{L-1}\prod_{k=0}^{K-1}(1+2\beta) = (1+2\beta)^{KL}.
\end{aligned}$$

To bound the second term of equation 18, we have

$$\begin{aligned}
||(I - \frac{1}{N}\mathbf{1}\mathbf{1}^T)A^{L-l}|| &= ||A^{L-l} - \frac{1}{N}\mathbf{1}\mathbf{1}^T|| \\
&\le 2N^2(1+\eta^{-(N-1)})(1-\eta^{N-1})^{L-l-1}.
\end{aligned}$$

where the inequality is also from [17] (Proposition 1). Then, we also have

$$\begin{aligned}
&\sum_{t=0}^{K-1}\prod_{\tilde{t}>t}^{K-1} ||B_{l,\tilde{t}}|| \cdot ||C_{l,t}|| \\
\le &\sum_{t=0}^{K-1}(1+2\beta)^{K-1-t} \cdot 2\beta\sqrt{N}r_{\max} \\
= &2\beta\sqrt{N}r_{\max}\sum_{t=0}^{K-1}(1+2\beta)^{K-1-t} \\
\le &4\beta K\sqrt{N}r_{\max}.
\end{aligned}$$

Then, for the multipliers, we have

$$||\prod_{j=1}^{L-1-l}\prod_{k=0}^{K-1} B_{l+j,k}|| \le (1+2\beta)^{(L-l-1)K} \le (1+4\beta K)^{L-l-1}.$$

Finally, for the second term in consensus error equation 18, we have

$$\begin{aligned}
&||\sum_{l=0}^{L-1}\prod_{j=1}^{L-1-l}\prod_{k=0}^{K-1} B_{l+j,k}\sum_{t=0}^{K-1}\prod_{\tilde{t}>t}^{K-1} B_{l,\tilde{t}}C_{l,t}(I - \frac{1}{N}\mathbf{1}\mathbf{1}^T)(A^T)^{L-l}|| \\
\le &\sum_{l=0}^{L-1}||\prod_{j=1}^{L-1-l}\prod_{k=0}^{K-1} B_{l+j,k}|| \cdot ||\sum_{t=0}^{K-1}\prod_{\tilde{t}>t}^{K-1} B_{l,\tilde{t}}C_{l,t}|| \cdot ||(I - \frac{1}{N}\mathbf{1}\mathbf{1}^T)(A^T)^{L-l}|| \\
\le &\sum_{l=0}^{L-1}(1+4\beta K)^{L-l-1} \cdot 4\beta K\sqrt{N}r_{\max} \cdot 2N^2(1+\eta^{-(N-1)})(1-\eta^{N-1})^{L-l-1} \\
\le &\kappa_2\beta K\sum_{l=0}^{L-1}\rho^{L-l-1} \\
\le &\frac{\kappa_2\beta K}{1-\rho}
\end{aligned}$$

where $\kappa_2 = 8(1+\eta^{-(N-1)})N^{\frac{5}{2}}r_{\max}$.

As a result, we have the results consensus bound of equation 9 in Lemma 1.

C.3 Proof of the Theorem 1

For the mean square error, we have

$$\begin{aligned}
&\mathbb{E}[\sum_{i=1}^{N}\|w^i_{L,0} - w^*\|^2] \\
= &\mathbb{E}[\sum_{i=1}^{N}\|w^i_{L,0} - \bar{w}_{L,0} + \bar{w}_{L,0} - w^*\|^2] \\
\le &2\mathbb{E}[\sum_{i=1}^{N}\|w^i_{L,0} - \bar{w}_{L,0}\|^2] + 2\mathbb{E}[\sum_{i=1}^{N}\|\bar{w}_{L,0} - w^*\|^2] \\
\le &2d\mathbb{E}[\|Q_{L,0}\|^2] + 2N\mathbb{E}[\|\bar{w}_{L,0} - w^*\|^2] \qquad (19)
\end{aligned}$$

where the first inequality is due to $\|x+y\|^2 \le 2\|x\|^2 + 2\|y\|^2$ and the second inequality $\|X\|_F \le \sqrt{d}\|X\|$ for $X \in \mathbb{R}^{d\times N}$. Then, the stated result in equation 11 follows from Lemmas 1 and 2, and equation 19.

This concludes the proof.

REFERENCES

[1] Jalaj Bhandari, Daniel Russo, and Raghav Singal. 2018. A finite time analysis of temporal difference learning with linear function approximation. In *Conference on learning theory*. PMLR, 1691–1692.

[2] Tianyi Chen, Kaiqing Zhang, Georgios B Giannakis, and Tamer Basar. 2018. Communication-efficient distributed reinforcement learning. *arXiv preprint arXiv:1812.03239* (2018).

[3] Xin Chen, Guannan Qu, Yujie Tang, Steven Low, and Na Li. 2022. Reinforcement learning for selective key applications in power systems: Recent advances and future challenges. *IEEE Transactions on Smart Grid* (2022).

[4] Ziyi Chen, Yi Zhou, Rongrong Chen, and Shaofeng Zou. 2021. Sample and Communication-Efficient Decentralized Actor-Critic Algorithms with Finite-Time Analysis. *arXiv preprint arXiv:2109.03699* (2021).

[5] Thinh Doan, Siva Maguluri, and Justin Romberg. 2019. Finite-time analysis of distributed TD (0) with linear function approximation on multi-agent reinforcement learning. In *International Conference on Machine Learning*. PMLR, 1626–1635.

[6] Thinh T Doan, Siva Theja Maguluri, and Justin Romberg. 2021. Finite-time performance of distributed temporal-difference learning with linear function approximation. *SIAM Journal on Mathematics of Data Science* 3, 1 (2021), 298–320.

[7] FNU Hairi, Jia Liu, and Songtao Lu. 2022. Finite-Time Convergence and Sample Complexity of Multi-Agent Actor-Critic Reinforcement Learning with Average Reward. In *International Conference on Learning Representations*.

[8] FNU Hairi, Zifan Zhang, and Jia Liu. 2023. Local TD-Update is More Sample-Efficient Than Batching for MARL Policy Evaluation with Average Reward. *https://kevinliu-osu.github.io/publications/MARL-PE-LocalTD-TR.pdf* (2023).

[9] Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. 2019. Learning to schedule communication in multi-agent reinforcement learning. *arXiv preprint arXiv:1902.01554* (2019).

[10] Chandrashekar Lakshminarayanan and Csaba Szepesvari. 2018. Linear stochastic approximation: How far does constant step-size and iterate averaging go?. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1347–1355.

[11] Donghwan Lee, Hyungjin Yoon, and Naira Hovakimyan. 2018. Primal-dual algorithm for distributed reinforcement learning: distributed GTD. In *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 1967–1972.

[12] David A Levin and Yuval Peres. 2017. *Markov chains and mixing times*. Vol. 107. American Mathematical Soc.

[13] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. 2017. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in Neural Information Processing Systems* 30 (2017).

[14] Yixuan Lin, Kaiqing Zhang, Zhuoran Yang, Zhaoran Wang, Tamer Başar, Romeil Sandhu, and Ji Liu. 2019. A communication-efficient multi-agent actor-critic algorithm for distributed reinforcement learning. In *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 5562–5567.

[15] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems* 30 (2017).

[16] Sergio Valcarcel Macua, Jianshu Chen, Santiago Zazo, and Ali H Sayed. 2014. Distributed policy evaluation under multiple behavior strategies. *IEEE Trans. Automat. Control* 60, 5 (2014), 1260–1274.

[17] Angelia Nedic and Asuman Ozdaglar. 2009. Distributed subgradient methods for multi-agent optimization. *IEEE Trans. Automat. Control* 54, 1 (2009), 48–61.

[18] Shi Pu and Angelia Nedić. 2021. Distributed stochastic gradient tracking methods. *Mathematical Programming* 187, 1 (2021), 409–457.

[19] Shuang Qiu, Zhuoran Yang, Jieping Ye, and Zhaoran Wang. 2021. On Finite-Time Convergence of Actor-Critic Algorithm. *IEEE Journal on Selected Areas in Information Theory* 2, 2 (2021), 652–664.

[20] Guannan Qu, Yiheng Lin, Adam Wierman, and Na Li. 2020. Scalable multi-agent reinforcement learning for networked systems with average reward. *Advances in Neural Information Processing Systems* 33 (2020), 2074–2086.

[21] Jineng Ren and Jarvis Haupt. 2019. A communication efficient hierarchical distributed optimization algorithm for multi-agent reinforcement learning. In *Real-world sequential decision making workshop at international conference on machine learning*.

[22] Martin Riedmiller, Andrew Moore, and Jeff Schneider. 2000. Reinforcement learning for cooperating and communicating reactive agents in electrical power grids. In *Workshop on Balancing Reactivity and Social Deliberation in Multi-Agent Systems*. Springer, 137–149.

[23] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2016. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295* (2016).

[24] Rayadurgam Srikant and Lei Ying. 2019. Finite-time error bounds for linear stochastic approximation andtd learning. In *Conference on Learning Theory*. PMLR, 2803–2830.

[25] Richard S Sutton. 1988. Learning to predict by the methods of temporal differences. *Machine learning* 3, 1 (1988), 9–44.

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

[27] J.N. Tsitsiklis and B. Van Roy. 1997. An analysis of temporal-difference learning with function approximation. *IEEE Trans. Automat. Control* 42, 5 (1997), 674–690. https://doi.org/10.1109/9.580874

[28] John N Tsitsiklis and Benjamin Van Roy. 1999. Average cost temporal-difference learning. *Automatica* 35, 11 (1999), 1799–1808.

[29] John N Tsitsiklis and Benjamin Van Roy. 2002. On average versus discounted reward temporal-difference learning. *Machine Learning* 49, 2 (2002), 179–191.

[30] Hoi-To Wai, Zhuoran Yang, Zhaoran Wang, and Mingyi Hong. 2018. Multi-agent reinforcement learning via double averaging primal-dual optimization. *Advances in Neural Information Processing Systems* 31 (2018).

[31] Hanyu Wei, Chan Wang, Rongpeng Li, and Minjian Zhao. 2022. Mean-field MARL-based Priority-Aware CSMA/CA Strategy in Large-Scale MANETs. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 311–316.

[32] Tengyu Xu, Zhe Wang, and Yingbin Liang. 2020. Improving sample complexity bounds for (natural) actor-critic algorithms. *arXiv preprint arXiv:2004.12956* (2020).

[33] Chao Yu, Xin Wang, Xin Xu, Minjie Zhang, Hongwei Ge, Jiankang Ren, Liang Sun, Bingcai Chen, and Guozhen Tan. 2019. Distributed multiagent coordinated learning for autonomous driving in highways based on dynamic coordination graphs. *IEEE Transactions on Intelligent Transportation Systems* 21, 2 (2019), 735–748.

[34] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2021. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control* (2021), 321–384.

[35] Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Basar. 2018. Fully decentralized multi-agent reinforcement learning with networked agents. In *International Conference on Machine Learning*. PMLR, 5872–5881.

[36] Xin Zhang, Zhuqing Liu, Jia Liu, Zhengyuan Zhu, and Songtao Lu. 2021. Taming Communication and Sample Complexities in Decentralized Policy Evaluation for Cooperative Multi-Agent Reinforcement Learning. Virtual Event.