Multi-Agent Trust Region Policy Optimization

Hepeng Li, *Student Member, IEEE,* and Haibo He, *Fellow, IEEE,*

Abstract—We extend trust region policy optimization (TRPO) [1] to multi-agent reinforcement learning (MARL) problems. We show that the policy update of TRPO can be transformed into a distributed consensus optimization problem for multi-agent cases. By making a series of approximations to the consensus optimization model, we propose a decentralized MARL algorithm, which we call multi-agent TRPO (MATRPO). This algorithm can optimize distributed policies based on local observations and private rewards. The agents do not need to know observations, rewards, policies or value/action-value functions of other agents. The agents only share a likelihood ratio with their neighbors during the training process. The algorithm is fully decentralized and privacy-preserving. Our experiments on two cooperative games demonstrate its robust performance on complicated MARL tasks.

Index Terms—Multi-agent reinforcement learning, trust region policy optimization, alternating direction method of multipliers, privacy protection, decentralized learning.

I. INTRODUCTION

RECENT developments in multi-agent reinforcement learning (MARL) have enabled individual agents to learn cooperative policies to jointly maximize a collective reward for a common goal. This has significantly extended the traditional RL paradigm from single agent domains to multi-agent systems (MAS), which have been found to be useful in a wide range of applications, such as multiplayer games [2], coordinating self-driving vehicles [3], multi-robot systems [4], traffic signal control [5] and smart grids [6].

Most MARL algorithms can be classified into three broad categories: (1) *fully centralized* methods, such as joint action learning [7], which learn a centralized policy by reducing the problem to a single-agent RL over the global observation and action space, (2) *centralized training with decentralized execution* methods, which learn distributed local policies (that select actions based on local observations) in a centralized manner by using global information in the training process, such as aggregated observations, joint rewards, policy parameters or gradients of other agents, etc., and (3) *fully decentralized* methods, such as independent learning [8], which learn local policies in a decentralized manner, and do not require global information in both the training and execution stages.

Of the three categories, *centralized training with decentralized execution* methods are preferred in many studies, because

they only need local information to execute the learned policies while being able to exploit global information to ease the *nonstationarity* issue [9]. *Nonstationarity* issue usually happens in independent learning. Since the policy of each agent changes during the training process, the state transition and reward functions perceived by other agents change as well. The mutual interference can influence the training process and make it difficult to converge towards an equilibrium point. However, in centralized training, local observations, rewards, or policies are shared among all the agents. In consequence, the environment dynamics perceived by the agents becomes stationary. Based on this idea, the multi-agent deep deterministic policy gradient (MADDPG) algorithm is proposed in [10], which trains a centralized action-value function for each agent based on aggregated observations and inferred actions. The local policy of each agent is updated according to the centralized action-value function and estimated policies of other agents. In [11], a value decomposition network is devised to implicitly learn the local action-value function for each individual agent using the aggregated observations and joint rewards. In [12], an actor-attention-critic algorithm is proposed to learn decentralized policies using centrally trained critics. An attention mechanism is incorporated to help select relevant information for each agent to improve the training of the critic.

Nevertheless, the *centralized training with decentralized execution* framework exhibits many limitations. Firstly, the existence of an unavoidable central controller makes the framework vulnerable to single point failure and cyber-attack. Secondly, training policies of all agents in one single controller may consume massive computation and communication resources, which poses a major challenge to the scalability and flexibility of centralized training methods. More importantly, privacy is a major concern in many multi-agent system applications, such as recommendation systems [13], agent-mediated e-commerce [14], semantic web services [15], etc. The requirement of access to other agent's observations, rewards or policies may raise concerns about privacy leakage, which makes centralized training methods questionable for many real-world applications.

TRPO has been successful at solving complex single-agent RL problems [1], and famous for its monotonic improvement guarantee and effectiveness for optimizing large neural network policies [16]. In this work, we extend the TRPO algorithm to MARL problems. We show that the policy update of TRPO can be equivalently transformed into a distributed consensus optimization problem. We approximately solve the consensus optimization, yielding a decentralized MARL algorithm, which we call multi-agent TRPO (MATRPO). In this algorithm, the policy updates are based on local observations and private rewards. The agents do not need to know observations, rewards, policies or value/action-value functions of other agents. During the training process, the agents only communicate with their

neighbors to share a local likelihood ratio. The algorithm is fully decentralized and friendly for privacy. In our experiments, we show that this algorithm can learn high-quality neural network policies for complex MARL problems.

II. RELATED WORK

There exists a series of insightful works in the literature that address the decentralized MARL problem, and a comprehensive overview of decentralized MARL methods can be found in [17], [18]. Like single-agent RL, most decentralized MARL methods can be classified into two categories: (1) policy iteration methods [19], which alternate between policy evaluation and policy improvement; (2) policy gradient methods [20], which update the policy according to an estimated gradient of expected return.

Most policy iteration methods focus on the distributed policy evaluation task, which estimates the global value function under a fixed joint policy. Generally, the task is converted to the mean square projected bellman error (MSPBE) minimization problem, and further reformulated as a distributed saddle-point problem by using linear approximation functions. For example, in [21], a diffusion-based distributed gradient temporal-difference (GTD) algorithm was developed to solve the saddle-point problem, and convergence under sufficiently small step-size updates was established. In [22], a primal-dual distributed GTD algorithm was proposed based on consensus, and asymptotic convergence was established by using ordinary differential equation. In [23], an gossip-based averaging scheme was investigated to incorporate neighbors information in distributed TD updates. In [24], a double averaging scheme combining the dynamic consensus and stochastic average gradient algorithm was proposed to solve saddle-point problem with a convergence guarantee at a global geometric rate. In [25], the finite-time convergence of a consensus-based distributed TD(0) algorithm under constant and time-varying step-sizes was investigated. Different from the above methods attempting to estimate the value function, the $\mathcal{DQ}$-learning algorithm in [26] was to learn the action-value function by distributed Q-factors, which were updated by using a $consensus + innovation$ algorithm. In [27], an multi-agent deep Q-network algorithm was proposed to learn distributed optimal action-value functions by aggregating the feature information from neighboring agents group via an attentive relational encoder.

For policy gradient methods, some works learn a local actor and a consensus critic and some others do the other way around. For example, following the former scheme, two decentralized actor-critic algorithms were proposed in [28] to estimate local policy gradients by using a consensus action-value function and a consensus value function, respectively. Instead of learning a consensus critic, the method in [29] derived a distributed off-policy actor critic algorithm with policy consensus. Through decomposition of the global value and action-value functions, the algorithm avoids learning a centralized critic. Converge guarantee was provided in both methods when using linear approximate functions. However, whether learning a consensus critic or learning a consensus actor, the agents mush know the global system state, which is not the case in a partially observable environment. In addition, privacy issue may arise,

especially in the policy consensus [29], wherein the policy parameters must be shared between agents. To address this issue, a flexible parameter sharing mechanism was introduced in [30] introduced by dividing the policy parameters into shared and non-shared parts. This also allows the agents to train distributed policies that only depend on local observations. To overcome the *nonstationarity* problem, this method assumes that all agents can infer policies of other agents.

Different from the aforementioned two categories of methods, the proposed algorithm is a local policy search method, which iteratively updates the policy by maximizing the expected return over a local neighborhood of the most recent iterate [1]. Compared to prior works, the proposed algorithm features the following major differences: (1) it is effective for optimizing distributed neural network policies; (2) it is privacy-preserving, which means that the agents do not need to share private information, such as local observations, rewards, policy or critic parameters; (3) it is suitable for partially observable MARL problems, wherein the agents have different observation spaces, reward functions and act according to different policies.

III. PRELIMINARIES

A. Markov Game

We consider a partially observable Markov game (POMG) [31], in which N agents communicate with their immediate neighbors through a sparsely connected network to cooperate with each other for a common purpose. The POMG is defined by the tuple $(\mathcal{S}, \{\mathcal{O}^i\}_{i\in\mathcal{N}}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \mathcal{P}, \{r^i\}_{i\in\mathcal{N}}, \rho_0, \gamma, \mathcal{G})$, where $\mathcal{S}$ is a set of environment states, $\mathcal{O}^i$ is a set of observations of the agent i, $\mathcal{A}^i$ is a set of actions that agent i can choose, $\mathcal{P} : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \times \mathcal{S} \mapsto [0,1]$ is the transition probability distribution, $r^i : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \mapsto \mathbb{R}$ is the private reward received by agent i, $\rho_0 : \mathcal{S} \mapsto [0,1]$ is the distribution of the initial state, $\gamma \in [0,1)$ is the discount factor, and $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is an undirected graph representing the communication network, where $\mathcal{N} = \{1, \ldots, N\}$ is the set of N agents and $\mathcal{E} = \{1, \ldots, L\}$ is the set of L communication links.

Let $\pi^i : \mathcal{O}^i \times \mathcal{A}^i \mapsto [0,1]$ denote a stochastic policy of agent i, and $\pi(a|s) = \prod_{i=1}^N \pi^i(a|s)$ denote the joint policy of all agents. For each agent $i \in \mathcal{N}$, the goal is to learn an optimal local policy π^i so that the joint policy π maximizes the expected sum of the discounted rewards of all agents, $J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^\infty \gamma^t(r_t^1 + \cdots + r_t^N)\right]$. Here τ denotes a trajectory ($\tau = (s_0, a_0^1 \ldots, a_0^N, s_1, \ldots)$), and $\tau \sim \pi$ indicates that the distribution over the trajectory depends on $\pi : s_0 \sim \rho_0, a_t^i \sim \pi^i(\cdot|s_t), s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t^1, \ldots, a_t^N)$.

Letting $R^i(\tau) = \sum_{t=0}^\infty \gamma^t r_t^i$ denote the discounted return of a trajectory perceived by agent i, we define the local value function as $V_\pi^i(s) = \mathbb{E}_{\tau\sim\pi}\left[R^i(\tau) \mid s_0 = s\right]$, local action-value function as $Q_\pi^i(s,a) = \mathbb{E}_{\tau\sim\pi}\left[R^i(\tau) \mid s_0 = s, a_0 = a\right]$, and local advantage function as $A_\pi^i(s,a) = Q_\pi^i(s,a) - V_\pi^i(s)$ respectively, where $a = (a^1, \ldots, a^N)$ is the joint action of all agents. With these definitions, we express the global value function by $V_\pi(s) = \sum_{i=1}^N V_\pi^i(s)$, the global action-value function by $Q_\pi(s,a) = \sum_{i=1}^N Q_\pi^i(s,a)$, and the global advantage function by $A_\pi(s,a) = \sum_{i=1}^N A_\pi^i(s,a)$.

B. Trust Region Policy Optimization

TRPO is a policy search method with monotonic improvement guarantee [1]. In TRPO, the policy is iteratively updated by maximizing a surrogate objective over a trust-region around the most recent iterate π_{old}:

$$\max_{\pi} \; \mathbb{E}_{s \sim \rho_{\pi_{old}}, a \sim \pi_{old}} \left[\frac{\pi(a|s)}{\pi_{old}(a|s)} A_{\pi_{old}}(s,a) \right] \qquad (1)$$
$$s.t. \; \overline{D}_{KL}^{\rho_{\pi_{old}}}(\pi_{old}, \pi) < \delta.$$

where the objective is an importance sampling estimator of the advantage function $A_{\pi_{old}}(s,a)$, and $\rho_{\pi_{old}}$ is the discounted visitation frequencies according to the policy π_{old}:

$$\rho_{\pi_{old}} = P(s_0 = s) + \gamma P(s_1 = s) + \gamma^2 P(s_1 = s) + \cdots.$$

The trust-region constraint restricts the searching area in the neighborhood of π_{old}, defined by the average KL-Divergence $\overline{D}_{KL}^{\rho_{\pi_{old}}}(\pi_{old}, \pi) = \mathbb{E}_{s \sim \rho_{\pi_{old}}}[D_{KL}(\pi_{old}(\cdot|s) \| \pi(\cdot|s))]$ and the parameter δ.

C. Alternating Direction Method of Multipliers

ADMM solves a separable equality-constrained optimization problem in the form [32]

$$\min_{x,z} \; f(x) + g(z) \qquad (2)$$
$$s.t. \; Ax + Bz = C,$$

where the decision variables $(x, z) \in \mathbb{R}^{n+m}$ decomposes into two sub-vectors, $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^m$, and the objective is a sum of convex functions, $f(x)$ and $g(z)$, with respect to these sub-vectors. The decision vectors x and z are coupled through a linear constraint, where $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{p \times m}$, and $C \in \mathbb{R}^p$.

The ADMM algorithm solves the problem (2) by forming its augmented Lagrange function,

$$\mathcal{L}_{\beta}(x,z,y) = f(x) + g(z) + y^T(Ax + Bz - C)$$
$$+ \frac{\beta}{2} \| Ax + Bz - C \|_2^2, \; \beta > 0, \qquad (3)$$

and approximately minimizing the augmented Lagrange function through sequentially updating the primal variables x and z, and the dual variable y:

$$x^{(k+1)} := \arg\min_{x} \mathcal{L}_{\rho}(x, z^{(k)}, y^{(k)}),$$
$$z^{(k+1)} := \arg\min_{z} \mathcal{L}_{\rho}(x^{(k)}, z, y^{(k)}), \qquad (4)$$
$$y^{(k+1)} := y^{(k)} + \beta(Ax^{(k)} + Bz^{(k)} - C).$$

IV. MULTI-AGENT TRUST REGION POLICY OPTIMIZATION

A. Trust Region Optimization for Multiple Agents

Consider the policy optimization problem (1) for the multi-agent case. We decompose the advantage function $A_{\pi_{old}}(s,a)$ in the objective and rewrite the policy optimization model as follows:

$$\max_{\pi} \mathbb{E}_{s \sim \rho_{\pi_{old}}, a \sim \pi_{old}} \left[\frac{\pi(a|s)}{\pi_{old}(a|s)} \left(A_{\pi_{old}}^1(s,a) + \cdots + A_{\pi_{old}}^N(s,a) \right) \right]$$
$$s.t. \; \overline{D}_{KL}^{\rho_{\pi_{old}}}(\pi_{old}, \pi) < \delta. \qquad (5)$$

Our purpose is to split the objective into N independent sub-objectives so that the optimization problem can be solved distributedly by N agents. However, the sub-objectives are coupled through the joint policy $\pi(a|s)$ and $\pi_{old}(a|s)$.

In our solution, instead of directly optimizing the joint policy $\pi(a|s)$, we train a local policy $\pi^i(a|s)$ for every agent $i \in \mathcal{N}$, where $\pi^i(a|s)$ is a probability distribution of the joint action $a = (a^1, \ldots, a^N)$ conditioned on s. When choosing actions, each agent i independently generate its action a^i according to the local marginal distribution $\pi^i(a^i|s)$. The joint policy can be expressed as $\pi(a|s) = \prod_{i=1}^{N} \pi^i(a^i|s)$.

Our principal theoretical result is that if the local policy has the form, i.e. $\pi^i(a|s) = \pi^i(a^1|s)\pi^i(a^2|s) \cdots \pi^i(a^N|s)$, the policy optimization (5) can be equivalently transformed into the following consensus optimization problem (see Appendix A for proof):

$$\max_{\{\pi^i\}_{i \in \mathcal{N}}} \; \sum_{i=1}^{N} \mathbb{E}_{\substack{s \sim \rho_{\pi_{old}} \\ a \sim \pi_{old}}} \left[\frac{\pi^i(a|s)}{\pi_{old}^i(a|s)} A_{\pi_{old}}^i(s,a) \right]$$
$$s.t. \; \frac{\pi^1(a^i|s)}{\pi_{old}^1(a^i|s)} = \cdots = \frac{\pi^N(a^i|s)}{\pi_{old}^N(a^i|s)}, \forall i \in \mathcal{N} \qquad (6)$$
$$\sum_{i=1}^{N} \mathbb{E}_{s \sim \rho_{\pi_{old}}} \left[D_{KL}(\pi_{old}^i(a^i|s) \| \pi^i(a^i|s)) \right] < \delta.$$

Now, the objective is separable with respect to local policies $\{\pi^i\}_{i \in \mathcal{N}}$. However, the local policy $\pi^i(a|s)$ is conditioned on s, which means in order to learn or act according to $\pi^i(a|s)$, every agent should know the global system state s. Generally, this is not the case in a partially observable environment. Since the agents act according to their local observations in a partially observable environment, the joint policy becomes $\pi(a|o) = \prod_{i=1}^{N} \pi^i(a^i|o^i)$, where $o = (o^1, \ldots, o^N)$ is the aggregated observation of all agents. In this case, we introduce the following approximation for partially observable environments:

$$\max_{\{\pi^i\}_{i \in \mathcal{N}}} \; \sum_{i=1}^{N} \mathbb{E}_{\substack{o^i \sim \rho_{\pi_{old}}^i \\ a \sim \pi_{old}}} \left[\frac{\pi^i(a|o^i)}{\pi_{old}^i(a|o^i)} A_{\pi_{old}}^i(o^i, a) \right]$$
$$s.t. \; \frac{\pi^1(a^i|o^1)}{\pi_{old}^1(a^i|o^1)} = \cdots = \frac{\pi^N(a^i|o^N)}{\pi_{old}^N(a^i|o^N)}, \forall i \in \mathcal{N} \qquad (7)$$
$$\overline{D}_{KL}^{\rho_{\pi_{old}}^i}(\pi_{old}^i, \pi^i) < \delta/N, \forall i \in \mathcal{N},$$

where we substitute the global state s with each agent's local observation $o^i \in \mathcal{O}^i, \forall i \in \mathcal{N}$. In order to meet the trust-region constraint on the KL-Divergence of the joint policy $\pi(a|o)$, we introduce N new constraints on the KL-Divergence of the local policy $\pi^i(a|o^i)$. According to the additive and non-negative properties of KL-Divergence, the following inequality holds:

$$\overline{D}_{KL}^{\rho_{\pi_{old}}}(\pi_{old}, \pi) = \sum_{i=1}^{N} \mathbb{E}_{o^i \sim \rho_{\pi_{old}}^i} \left[D_{KL}(\pi_{old}^i(a^i|o^i) \| \pi^i(a^i|o^i)) \right]$$
$$\leq \sum_{i=1}^{N} \mathbb{E}_{o^i \sim \rho_{\pi_{old}}^i} \left[D_{KL}(\pi_{old}^i(a|o^i) \| \pi^i(a|o^i)) \right]$$
$$= \sum_{i=1}^{N} \overline{D}_{KL}^{\rho_{\pi_{old}}^i}(\pi_{old}^i, \pi^i). \qquad (8)$$

Consequently, when the N new KL-Divergence constraints in (7) are satisfied, i.e. $\overline{D}_{KL}^{\rho_{\pi_{old}^i}}(\pi_{old}^i, \pi^i) < \delta/N$, the joint policy π will be constrained within the trust region $\overline{D}_{KL}^{\rho_{\pi_{old}}}(\pi_{old}, \pi) \leq \delta$. It is noted that if the agents have enough observations (e.g. sufficient long histories) to almost surely infer the global state, the approximation (7) could be sufficiently close to the problem (6) for the fully-observable case.

For the policy optimization (7), the joint action a is required to calculate the likelihood ratio $\frac{\pi^i(a|o^i)}{\pi_{old}^i(a|o^i)}$ and the advantage function $A_{\pi_{old}}^i(o^i, a)$. However, the joint action a follows the policy π_{old} rather than π, which means that the agents only need to know the actions performed by the other agents in the last iterate. Knowing the last round actions of other agents is not a restrictive assumption and has been widely adopted by existing decentralized algorithms in the literature. It is also worth mentioning that knowing the actions of other agents does not mean knowing their policies because the local policy $\pi_{old}^i(a|o^i)$ is conditioned on local observation o^i, which is privately owned be the agent i.

B. Distributed Consensus with ADMM

To solve the distributed consensus optimization problem (7), we employ the asynchronous ADMM algorithm proposed in [33]. Specifically, let $\mathcal{N}(e) = \{i, j\}$ denote the agents at the endpoints of the communication link $e \in \mathcal{E}$. For each agent $q = (i, j)$, we introduce an estimator $z_e^q(o^q, a^n), \forall n \in \mathcal{N}$ to estimate the likelihood ratio of its neighbor $\mathcal{N}(e)\backslash\{q\}$. Then, the consensus constraint for the agent i and j can be reformulated as

$$\frac{\pi^i(a^n|o^i)}{\pi_{old}^i(a^n|o^i)} = z_e^i(o^i, a^n), \forall n \in \mathcal{N} \tag{9a}$$

$$-\frac{\pi^j(a^n|o^j)}{\pi_{old}^j(a^n|o^j)} = z_e^j(o^j, a^n), \forall n \in \mathcal{N} \tag{9b}$$

$$z_e^i(o^i, a^n) + z_e^j(o^j, a^n) = 0, \forall n \in \mathcal{N}. \tag{9c}$$

For brevity, we use $\Upsilon^n(\pi^i)$ to represent $\frac{\pi^i(a^n|o^i)}{\pi_{old}^i(a^n|o^i)}$, $\Upsilon(\pi^i)$ to represent $\frac{\pi^i(a|o^i)}{\pi_{old}^i(a|o^i)}$, and z_e^{qn} to represent $z_e^q(o^q, a^n), q \in \mathcal{N}(e)$. By using Equations (9), the problem (7) can be transformed into

$$\min_{\boldsymbol{\pi}, \boldsymbol{z}} \ -\sum_{i=1}^{N} \mathop{\mathbb{E}}_{\substack{o^i \sim \rho_{\pi_{old}}^i \\ a \sim \pi_{old}}} \left[\Upsilon(\pi^i) A_{\pi_{old}}^i(o^i, a) \right]$$

$$s.t. \ C_e^q \Upsilon^n(\pi^q) = z_e^{qn}, \ e \in \mathcal{E}, q \in \mathcal{N}(e), n \in \mathcal{N},$$

$$\sum_q z_e^{qn} = 0, \ e \in \mathcal{E}, q \in \mathcal{N}(e), n \in \mathcal{N} \tag{10}$$

$$\overline{D}_{KL}^{\rho_{\pi_{old}}^i}(\pi_{old}^i, \pi^i) < \delta/N, \ \forall i \in \mathcal{N},$$

where $\boldsymbol{\pi} = \{\pi^i\}_{i\in\mathcal{N}}$, $\boldsymbol{z} = \{z_e^{qn}\}_{e\in\mathcal{E}, q\in\mathcal{N}(e), n\in\mathcal{N}}$, and C_e^q is the weight on the information transmitted between agents $q = (i, j) \in \mathcal{N}(e)$ at the communication link e. The value of C_e^q must satisfy $\sum_{q\in\mathcal{N}(e)} C_e^q = 0$, and is set to either 1 or -1 in our study.

We form the augmented Lagrange function of problem (10):

$$\mathcal{L}_\beta(\boldsymbol{\pi}, \boldsymbol{z}, \boldsymbol{y}) = - \mathop{\mathbb{E}}_{\substack{o^i \sim \rho_{\pi_{old}}^i \\ a \sim \pi_{old}}} \sum_{i\in\mathcal{N}} \Upsilon(\pi^i) A_{\pi_{old}}^i(o^i, a) \tag{11}$$

$$+ \mathop{\mathbb{E}}_{\substack{o^i \sim \rho_{\pi_{old}}^i \\ a \sim \pi_{old}}} \sum_{e\in\mathcal{E}} \sum_{q\in\mathcal{N}(e)} \sum_{n\in\mathcal{N}} y_e^{qn} \left(C_e^q \Upsilon^n(\pi^q) - z_e^{qn} \right)$$

$$+ \mathop{\mathbb{E}}_{\substack{o^i \sim \rho_{\pi_{old}}^i \\ a \sim \pi_{old}}} \sum_{e\in\mathcal{E}} \sum_{q\in\mathcal{N}(e)} \sum_{n\in\mathcal{N}} \frac{\beta}{2} \parallel C_e^q \Upsilon^n(\pi^q) - z_e^{qn} \parallel_2^2,$$

where $\boldsymbol{y} = \{y_e^{qn}\}_{e\in\mathcal{E}, q\in\mathcal{N}(e), n\in\mathcal{N}}$ is the set of the dual variables, $\beta > 0$ is a penalty parameter; the local policy π^i is defined in the feasible set $\Pi^i = \{\pi | \overline{D}_{KL}^{\rho_{\pi_{old}}^i}(\pi_{old}^i, \pi) < \delta/N\}$, and the estimator z_e^{qn} is defined in the feasible set $Z_e = \{z_e^{in}, z_e^{jn} | z_e^{in} + z_e^{jn} = 0, i, j \in \mathcal{N}(e), n \in \mathcal{N}\}$.

To solve problem (10), the asynchronous ADMM minimizes the augmented Lagrange function by sequentially updating the local policies, estimators, and the dual variables. Specifically, let $\boldsymbol{\pi}^{(k)}$, $\boldsymbol{z}^{(k)}$, $\boldsymbol{y}^{(k)}$ be the values of $\boldsymbol{\pi}, \boldsymbol{z}, \boldsymbol{y}$ at iteration k. At iteration $k + 1$, a communication link $e = (i, j)$ is selected according to a certain probability $\psi_e > 0$. Then, the two agents $q \in \mathcal{N}(e)$ at the endpoints of the link e are activated to update their policy π^q, estimator z_e^{qn}, and dual variable y_e^{qn} according to:

$$\pi^{q(k+1)} := \arg\min_{\pi^q \in \Pi^q} \mathcal{L}_\beta(\boldsymbol{\pi}, \boldsymbol{z}^{(k)}, \boldsymbol{y}^{(k)}),$$

$$z_e^{qn(k+1)} := \arg\min_{z_e^{qn} \in Z_e} \mathcal{L}_\beta(\boldsymbol{\pi}^{(k+1)}, \boldsymbol{z}, \boldsymbol{y}^{(k)}), \tag{12}$$

$$y_e^{qn(k+1)} := y_e^{qn(k)} + \beta(C_e^q \Upsilon^n(\pi^{q(k+1)}) - z_e^{qn(k+1)}).$$

Wei and Ozdaglary proved in [33] that if the objective is convex, and Π^i and Z_e are nonempty closed convex sets, the algorithm (12) converges almost surely to the optimal solution at the rate of $\mathcal{O}(1/k)$ as $k \mapsto \infty$.

C. Sequential Convexification for Convergence

In the problem (10), we implicitly assume that the policy $\pi^i(a|o^i)$ can be evaluated at every observation and action, and optimize directly over π^i. However, for parameterized policies $\pi^i(a|o^i; \theta^i)$, like neural networks, we would like to optimize over the parameters θ^i. This can lead to convergence issue of the ADMM algorithm due to non-convexity of $\pi^i(a|o^i; \theta^i)$ with respect to θ^i [32], [33].

Since we are considering parameterized policy $\pi^i(a|o^i; \theta^i)$, we will use the parameter vector θ^i to represent the policy π^i, and overload all previous notations, e.g. $\Upsilon^n(\theta^i) := \Upsilon^n(\pi^i)$, $f^i(\pi^i) := f^i(\theta^i)$ and $\overline{D}_{KL}^{\rho_{\theta_{old}}^i}(\theta_{old}^i, \theta^i) := \overline{D}_{KL}^{\rho_{\pi_{old}}^i}(\pi_{old}^i, \pi^i)$.

To address the convergence issue, we borrow an idea from sequential convex programming [34] and approximate the problem (10) with a convex model. Note that, for a small trust region δ, the likelihood ratio $\Upsilon^n(\theta^i)$ and $\Upsilon(\theta^i)$ can be well approximated by first order Taylor expansion round θ_{old}^i:

$$\Upsilon^n(\theta^i) \approx 1 + \nabla_{\theta^i}^T \Upsilon^n(\theta^i)(\theta^i - \theta_{old}^i),$$

$$\Upsilon(\theta^i) \approx 1 + \sum_{n=1}^{N} \nabla_{\theta^i}^T \Upsilon^n(\theta^i)(\theta^i - \theta_{old}^i) \tag{13}$$

and the KL-divergence can be well approximate by second order Taylor expansion round θ_{old}^i:

$$\overline{D}_{KL}^{\rho_{\theta_{old}}^i}(\theta_{old}^i, \theta^i) \approx \frac{1}{2}(\theta^i - \theta_{old}^i)^T H_i(\theta^i - \theta_{old}^i), \quad (14)$$

where $H_i = \nabla_{\theta^i}^2 \overline{D}_{KL}^{\rho_{\theta_{old}}^i}(\theta_{old}^i, \theta^i)$. The idea is also similar to TRPO [1] and constrained policy optimization (CPO) [35].

Substituting (13) - (14) into (10) and (11), we can derive a convex approximation to the primal problem and the augmented Lagrange function, respectively. With the convex approximation, we can use the asynchronous ADMM algorithm (12) to solve for an improved policy θ^i for each agent i.

D. Privacy Protection

During the training process, the agents need to share with neighbors the information on the local likelihood ratio $\Upsilon^n(\theta^i)$ to reach a consensus. The likelihood ratio does not contain private information of the agents, such as the observation, reward, or policy parameters. Therefore, MATRPO is friendly for agents' privacy.

V. PRACTICAL IMPLEMENTATION

A. Sample-Based Approximation of the Consensus Constraint

To meet the consensus constraint, the agents need to reach agreement on the ratio

$$\Upsilon^n(\theta^i) = \frac{\pi^i(a^n|o^i; \theta^i)}{\pi^i(a^n|o^i; \theta_{old}^i)}, \forall n \in \mathcal{N}$$

for every observation and action. This task is challenging when the observation and action spaces are very large. To overcome this challenge, we implement a sample-based approximation to the consensus constraint.

First, we generate a batch of D initial states $\{s_{0,d}\}_{d=1,...,D}$ according to the distribution ρ_0, and each agent has a batch of local observations of the state, i.e. $\{o_{0,d}^i\}_{d=1,...,D}$. Then, we simulate the joint policy $\pi(a|o)$ for T timesteps to generate a batch of D trajectories of the states and actions, and each agent i has a batch of D locally observed trajectories, i.e. $\mathcal{D}^i = \{(o_{0,d}^i, a_{0,d}, o_{1,d}^i, \ldots, a_{T-1,d}, o_{T-1,d}^i)_{d=1,...,D}\}$. After that, the agents calculate the local likelihood ratios $\Upsilon^n(\theta^i)$ based on the $M(M = D \times T)$ samples from these trajectories. Finally, we constrain the agents to reach consensus on the M samples of the likelihood ratio $\Upsilon^n(\theta^i)$.

We also use this sampling procedure to estimate the local advantage function $A_{\pi_{old}}^i(o^i, a)$ [35]:

$$\hat{A}_{\pi_{old}}^i = \delta_t + \lambda\delta_t + \cdots + \lambda^{T-t+1}\delta_{T-1}, \quad \text{where } \delta_t = r_t^i + \gamma V_{\pi_{old}}^i(o_{t+1}^i) - V_{\pi_{old}}^i(o_t^i), \quad (15)$$

and the local value function $V_{\pi_{old}}^i(o^i)$ is approximated by using a neural network based on the local observations o^i and private rewards r^i.

B. Agreement on Logarithmic Ratio

We find that it is better to use the logarithmic likelihood ratio $\log(\Upsilon^n(\theta^i))$ for consensus since it leads to a closed-form solution to the asynchronous ADMM updates (12). As the logarithmic function is monotonic, consensus on $\log(\Upsilon^n(\theta^i))$ is equivalent to consensus on $\Upsilon^n(\theta^i)$. Therefore, the following equations are used as the consensus constraints:

$$C_e^q \log(\Upsilon^n(\theta^q)) = z_e^{qn}, \ e \in \mathcal{E}, q \in \mathcal{N}(e), n \in \mathcal{N}. \quad (16)$$

In addition, we approximate the logarithmic likelihood ratio by its first order Taylor expansion around θ_{old}^i:

$$\begin{aligned} \log(\Upsilon^n(\theta^i)) &\approx \nabla_{\theta^i}^T \log(\Upsilon^n(\theta^i))(\theta^i - \theta_{old}^i) \\ &= \nabla_{\theta^i}^T \Upsilon^n(\theta^i)(\theta^i - \theta_{old}^i). \end{aligned} \quad (17)$$

Then, using the sample-based approximation procedure in Section V-A, we derive the following Lagrange function:

$$\begin{aligned} \mathcal{L}_\beta(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{y}) = &-\frac{1}{M} \sum_{i=1}^N \sum_{n=1}^N \hat{A}_{\pi_{old}}^{i\ T} J^{in}(\theta^i - \theta_{old}^i) \quad (18) \\ &+ \frac{1}{M} \sum_{e=1}^L \sum_{q\in\mathcal{N}(e)} \sum_{n=1}^N y_e^{qnT}[C_e^q J^{qn}(\theta^q - \theta_{old}^q) - z_e^{qn}] \\ &+ \frac{1}{M} \sum_{e=1}^L \sum_{q\in\mathcal{N}(e)} \sum_{n=1}^N \frac{\beta}{2} \parallel C_e^q J^{qn}(\theta^q - \theta_{old}^q) - z_e^{qn} \parallel_2^2, \end{aligned}$$

where $\hat{A}_{\pi_{old}}^i \in \mathbb{R}^{M\times 1}$ is the sample estimate of the advantage function $A_{\pi_{old}}^i(o^i, a)$, $J^{in} \in \mathbb{R}^{M\times D}$ is the Jacobian of the sampled likelihood ratios $\Upsilon^n(\theta^i)$ with respect to $\theta^i \in \mathbb{R}^{D\times 1}$. The policy parameter θ^i is defined in the feasible set $\theta^i \in \Theta^i = \{\theta|\frac{1}{2}(\theta - \theta_{old}^i)^T \overline{H}^i(\theta - \theta_{old}^i) \leq \delta/N\}$, where $\overline{H}^i \in \mathbb{R}^{D\times D}$ is the Hessian of the sample average KL-Divergence $\frac{1}{M}\sum_{m=1}^M D_{KL}(\pi_{old}^i(\cdot|o^i; \theta_{old}^i)||\pi^i(\cdot|o^i; \theta^i))$.

The asynchronous ADMM updates for the Lagrange function (18) have the following closed-forms (see Appendix A-B for proof):

$$\theta^{q(k+1)} := \theta_{old}^q + \sqrt{\frac{2\delta/N}{V^T \overline{H}^{q\ -1} V}} \overline{H}^{q\ -1} V,$$

$$z_e^{qn(k+1)} := \frac{1}{\beta}(y_e^{qn(k)} - \nu_e^n) + C_e^q J^{qn}(\theta^{q(k+1)} - \theta_{old}^q),$$

$$y_e^{qn(k+1)} := \nu_e^n, \quad (19)$$

where

$$V = \frac{1}{M} \sum_{n=1}^N J^{qnT}(\hat{A}_{\pi_{old}}^q - \sum_{e\in\mathcal{E}(q)} C_e^q y_e^{qn(k)} + \beta \sum_{e\in\mathcal{E}(q)} C_e^q z_e^{qn(k)}),$$

$$\nu_e^n = \frac{1}{2} \sum_{q\in\mathcal{N}(e)} \left[y_e^{qn(k)} + \beta C_e^q J^{qn}(\theta^{q(k+1)} - \theta_{old}^q) \right].$$

and $\mathcal{E}(q)$ is the set of all communication links that directly connect to agent q. The pseudocode for our algorithm is given as Algorithm 1.

Algorithm 1 Multi-Agent Trust Region Policy Optimization

Input: Initial local policies $\pi^i(\theta_0^i), \forall i \in \mathcal{N}$; tolerance δ
for $itr = 0, 1, 2, \ldots$ **do**
    Set the joint policy $\pi(\theta_{old}) = \prod_{i=1}^N \pi^i(a^i|o^i; \theta_{itr}^i)$
    for $i = 0, 1, \ldots, N$ **do**
        Sample a set of trajectories $\mathcal{D}^i \sim \pi(\theta_{old})$ for agent i
        Form sample estimates $A^i, J^{in}, \overline{H}^i, \forall i, n \in \mathcal{N}$
    end for
    Initial estimators $\boldsymbol{z}^{(0)}$ and dual variables $\boldsymbol{y}^{(0)}$
    for $k = 0, 1, 2, \ldots$ **do**
        Randomly select a communication link e
        For agents $q = (i, j) \in \mathcal{N}(e)$ at the endpoints of link
        e, update θ^q, z_e^{qn} and y_e^{qn} according to (26)
        Update $\boldsymbol{\theta}^{(k)}, \boldsymbol{z}^{(k)}, \boldsymbol{y}^{(k)} \rightarrow \boldsymbol{\theta}^{(k+1)}, \boldsymbol{z}^{(k+1)}, \boldsymbol{y}^{(k+1)}$
    end for
end for

VI. Experiments

The experiments are designed to investigate the following questions:

- Does MATRPO succeed in learning cooperative policies in a fully decentralized manner when agents have different observation spaces and reward functions?
- How does MATRPO compare with other baseline methods, such as a fully centralized learning method? Does MATRPO perform better than independent learning?
- Can MATRPO be used to solve large-scale MARL problems? Does the performance degrade when the number of agents increases?

To answer these questions, we test MATRPO on two cooperative tasks, in which agents only have partial observations of the environment and receive private rewards. We compare the final performance of MATRPO with two baseline methods: (1) a fully centralized method, which uses TRPO to learn a global policy by reducing the problem to a single-agent RL, (2) an independent learning method, which uses TRPO to learn a local policy by maximizing the local expected rewards. We also examine the scalability of our algorithm by increasing the number of agents in the tasks.

A. Environments

Our experiments are carried out on the multi-agent particle environment proposed by [36], and extended by [10] and [12]. The *Cooperative Navigation* and *Cooperative Treasure Collection* tasks are used to test the proposed algorithm. To make the tasks more challenging in the decentralized training scheme, they are modified, and details on the modified tasks are illustrated in Fig. 1 and explained as follows.

Cooperative Navigation. In this task, N agents must reach a set of N landmarks through coordinated movements. In the original version of this task, the agents are collectively rewarded based on the minimum agent distance to each landmark, and individually punished when colliding with each other. To make it more challenging, we modified the task such that only one of the agents is rewarded. The other agents do not receive reward, bet they are penalized when collision. The agents observe the

(a) Cooperative Navigation   (b) Cooperative Treasure Collection

Fig. 1. (a) *Cooperative Navigation* ($N = 3$). The agents must cover all the landmarks through coordinated movements. While all agents are punished for colliding with each other, only agent 1 is rewarded based on the proximity of any agent to each landmark. The agents communicate through a ring topology network, i.e. $1 - 2 - 3 - 1 - \cdots$. (b) *Cooperative Treasure Collection* ($N_h = 6, N_b = 2$). The hunters need to collect treasures, which are represented by small colored circles, and deposit them into correctly colored banks. The banks are rewarded for the successful collection and depositing of treasure. The hunters do not receive any reward, but are penalized for colliding with each other. The agents communicate through a ring topology network, i.e. Bank 1 - Bank 2 - Hunter 1 - $\cdots$ - Hunter 6 - Bank 1 - $\cdots$.

position and velocity of itself as well as the relative positions of other agents and landmarks.

Cooperative Treasure Collection. In this task, N_h treasure hunters and N_b treasure banks search around the environment to gather treasures. The treasures are generated with different colors and re-spawn randomly upon being collected. The hunters are responsible for collecting treasures and deposit them into correctly colored banks. The banks simply gather as much treasure as possible from the hunters. The agents observe the position and velocity of itself as well as the relative positions and velocities of other agents. The agents also observe the colors and re-spawn positions of the treasures. In our modified version of this task, the banks are rewarded for successful collection and depositing of treasure. They are also negatively rewarded based on the minimum distance of each hunter to the treasures. The hunters do not receive any reward for collecting treasures, but are penalized for colliding with each other. This modification makes the task more challenging than the original version where both the hunters and the banks are rewarded.

For all experiments, we use neural network policies with two hidden layers of 128 SeLU units [37]. The action space is {*left*, *right*, *up*, *down*, *stay*}. For decentralized training, we assume that the agents communicate with neighbors via a ring topology network. We run MATRPO and the baselines algorithms 5 times with different random seeds and compare their average performances on both tasks. Details on the experimental setup and parameters are provided in Table I.

B. Results

Learning curves of MATRPO and the baseline methods for the two cooperative tasks are shown in Fig. 2. MATRPO is successful at learning collaborative policies and achieves consistently high performance for all of the problems. Apart from the small cooperative navigation task ($N = 3$) where Centralized TRPO almost finds the global optimum, MATRPO
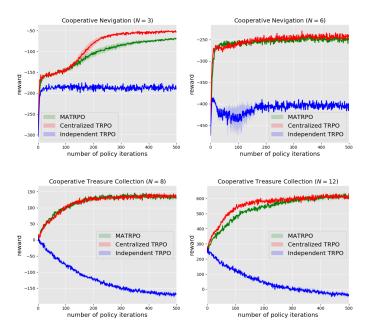
Fig. 2. Learning curves for the *Cooperative Navigation* and *Cooperative Treasure Collection* tasks. Error bars are a 95% confidence interval across 5 runs.

TABLE I
PARAMETERS OF MATRPO FOR THE COOPERATIVE NAVIGATION AND COOPERATIVE TREASURE COLLECTION TASKS.

Task	Cooper. Navigation		Cooper. Treas. Collection	
Number of Agents	$N = 3$	$N = 6$	$N = 8$	$N = 12$
Sim. steps per iter.	10k	10k	10k	10k
Stepsize (δ/N).	0.003	0.003	0.001	0.001
Discount (γ).	0.995	0.995	0.995	0.995
Policy iter.	500	500	500	500
Num. path	100	100	100	100
Path len.	100	100	100	100
ADMM iter.	100	150	200	500
Penalty para. (β).	1.0	1.0	5.0	5.0

performs practically as well as the Centralized TRPO does. Independent TRPO performs poorly on these tasks. Especially, for the cooperative treasure collection task, the performance becomes worse as the training proceeds. Since the hunters are punished for colliding with each other, agents trained by Independent TRPO learn to move far away from each other to avoid collision. As a result, some hunters move outside the window and end up with being distant from the treasures.

In addition, from the experimental results we find that the performance of MARTPO does not deteriorate when the number of agents increase. These results provide empirical evidence that MATRPO can be used to solve large-scale MARL problems.

Note that MATRPO learned all of the collaborative policies in a decentralized way using local observation and privately-owned rewards. The agents does not need to know observations, rewards, or policies of other agents. This is in contrast with most prior methods, which typically rely on aggregated observations and rewards, or policy and action value/value network parameters of other agents.

VII. CONCLUSION

In the paper, we showed that through a particular decomposition and transformation, the TRPO algorithm can be extended to optimization of cooperative policies in multi-agent systems. This enabled the development of our MATRPO algorithm, which updates the local policy through distributed consensus optimization.

In our experiments, we demonstrated that MATRPO can train distributed nonlinear policies based on only local observations and rewards. Furthermore, the privacy of the agents can be well protected during the training process. We justified the superiority and scalability of MATRPO on two cooperative tasks by comparing with centralized TRPO and independent learning. Our work represents a step towards applying MARL in real-world applications, where privacy and scalability are important.

APPENDIX A

A. Equivalent Transformation of Policy Optimization in TRPO

Consider a multi-agent system with N agents. Let $\pi^i(a|s)$ be the local policy trained by agent $i \in \mathcal{N} = \{1, 2, \ldots, N\}$. When choosing actions, each agent i acts independently according to the local marginal distribution $\pi^i(a^i|s)$. The joint policy is expressed as $\pi(a|s) = \prod_{i=1}^{N} \pi^i(a^i|s)$.

Theorem 1. *Assume the local policy has the form $\pi^i(a|s) = \pi^i(a^1|s)\pi^i(a^2|s)\cdots\pi^i(a^N|s)$. Then, the TRPO policy update (5) can be equivalently transformed into:*

$$\max_{\{\pi^i\}_{i\in\mathcal{N}}} \quad \sum_{i=1}^{N} \mathop{\mathbb{E}}_{\substack{s\sim\rho_{\pi_{old}} \\ a\sim\pi_{old}}} \left[\frac{\pi^i(a|s)}{\pi^i_{old}(a|s)} A^i_{\pi_{old}}(s,a) \right]$$

$$s.t. \quad \frac{\pi^1(a^i|s)}{\pi^1_{old}(a^i|s)} = \cdots = \frac{\pi^N(a^i|s)}{\pi^N_{old}(a^i|s)}, \forall i \in \mathcal{N} \quad (20)$$

$$\sum_{i=1}^{N} \mathbb{E}_{s\sim\rho_{\pi_{old}}} D_{KL}(\pi^i_{old}(a^i|s) \parallel \pi^i(a^i|s)) < \delta.$$

Proof. We expand the likelihood ratio $\frac{\pi^i(a|s)}{\pi^i_{old}(a|s)}$ of agent i into

$$\frac{\pi^i(a|s)}{\pi^i_{old}(a|s)} = \frac{\pi^i(a^1|s)}{\pi^i_{old}(a^1|s)} \frac{\pi^i(a^2|s)}{\pi^i_{old}(a^2|s)} \cdots \frac{\pi^i(a^N|s)}{\pi^i_{old}(a^N|s)}. \quad (21)$$

When the consensus constraints in (20) are satisfied, we can rewrite the expansion (21) by replacing $\frac{\pi^i(a^j|s)}{\pi^i_{old}(a^j|s)}$ for all $i \neq j$ with $\frac{\pi^j(a^j|s)}{\pi^j_{old}(a^j|s)}$ as follows:

$$\frac{\pi^i(a|s)}{\pi^i_{old}(a|s)} = \frac{\pi^1(a^1|s)}{\pi^1_{old}(a^1|s)} \cdots \frac{\pi^N(a^N|s)}{\pi^N_{old}(a^N|s)} = \frac{\pi(a|s)}{\pi_{old}(a|s)} \quad (22)$$

Substituting (22) into the objective of (20), we have

$$\sum_{i=1}^{N} \mathop{\mathbb{E}}_{\substack{s\sim\rho_{\pi_{old}} \\ a\sim\pi_{old}}} \left[\frac{\pi^i(a|s)}{\pi^i_{old}(a|s)} A^i_{\pi_{old}}(s,a) \right]$$

$$= \mathop{\mathbb{E}}_{\substack{s\sim\rho_{\pi_{old}} \\ a\sim\pi_{old}}} \left[\frac{\pi(a|s)}{\pi_{old}(a|s)} \sum_{i=1}^{N} A^i_{\pi_{old}}(s,a) \right] \quad (23)$$

$$= \mathop{\mathbb{E}}_{\substack{s\sim\rho_{\pi_{old}} \\ a\sim\pi_{old}}} \left[\frac{\pi(a|s)}{\pi_{old}(a|s)} A_{\pi_{old}}(s,a) \right]$$

In addition, since the joint policy is $\pi(a|s) = \prod_{i=1}^{N} \pi^i(a^i|s)$, we have

$$\sum_{i=1}^{N} \mathbb{E}_{s \sim \rho_{\pi_{old}}} D_{KL}(\pi_{old}^i(a^i|s) \| \pi^i(a^i|s))$$

$$= \mathbb{E}_{s \sim \rho_{\pi_{old}}} \sum_{i=1}^{N} D_{KL}(\pi_{old}^i(a^i|s) \| \pi^i(a^i|s)) \quad (24)$$

$$= \mathbb{E}_{s \sim \rho_{\pi_{old}}} D_{KL}(\pi_{old}(\cdot|s) \| \pi(\cdot|s))$$

$$= \overline{D}_{KL}^{\rho_{\pi_{old}}}(\pi_{old}, \pi).$$

$\square$

B. Analytical Solution to ADMM Updates

Consider the augmented Lagrange function $\mathcal{L}_\beta(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{y})$ in (18), and the asynchronous ADMM updates for:

$$\theta^{q(k+1)} := \arg\min_{\theta^q \in \Theta^q} \mathcal{L}_\beta(\boldsymbol{\theta}, \boldsymbol{z}^{(k)}, \boldsymbol{y}^{(k)}),$$
$$z_e^{qn(k+1)} := \arg\min_{z_e^{qn} \in Z_e} \mathcal{L}_\beta(\boldsymbol{\theta}^{(k+1)}, \boldsymbol{z}, \boldsymbol{y}^{(k)}), \quad (25)$$
$$y_e^{qn(k+1)} := y_e^{qn(k)} + \beta(C_e^q J^{qn}(\theta^{q(k+1)} - \theta_{old}^q) - z_e^{qn(k+1)}).$$

where $\Theta^q = \{\theta| \frac{1}{2}(\theta - \theta_{old}^q)^T \overline{H}^q (\theta - \theta_{old}^q) \leq \delta/N\}$ and $Z_e = \{z_e^{in}, z_e^{jn} | z_e^{in} + z_e^{jn} = 0, q = (i,j) \in \mathcal{N}(e), n \in \mathcal{N}\}$. Next, we give the analytical solutions to the ADMM updates.

Theorem 2. *If there exists at least one strictly feasible point in the feasible sets for Θ^i and Z_e for the updates in (25), the optimal updates $\theta^{q(k+1)}$, $z_e^{qn(k+1)}$, and $y_e^{qn(k+1)}$ satisfy:*

$$\theta^{q(k+1)} := \theta_{old}^q + \sqrt{\frac{2\delta/N}{V^T \overline{H}^{q-1} V}} \overline{H}^{q-1} V,$$
$$z_e^{qn(k+1)} := \frac{1}{\beta}(y_e^{qn(k)} - \nu_e^n) + C_e^q J^{qn}(\theta^{q(k+1)} - \theta_{old}^q),$$
$$y_e^{qn(k+1)} := \nu_e^n$$
$$(26)$$

when $M \to \infty$, where

$$V = \frac{1}{M} \sum_{n=1}^{N} J^{qnT}(\hat{A}_{\pi_{old}}^q - \sum_{e \in \mathcal{E}(q)} C_e^q y_e^{qn(k)} + \beta \sum_{e \in \mathcal{E}(q)} C_e^q z_e^{qn(k)}),$$

$$\nu_e^n = \frac{1}{2} \sum_{q \in \mathcal{N}(e)} \left[y_e^{qn(k)} + \beta C_e^q J^{qn}(\theta^{q(k+1)} - \theta_{old}^q) \right].$$

and $\mathcal{E}(q)$ is the set of the communication links that directly connect to agent q.

Proof. Define $x^q = \theta^q - \theta_{old}^q$ and $x^q \in X^q = \{x| \frac{1}{2} x^T \overline{H}^q x \leq \delta/N\}$. The Lagrange function (18) is simplified as

$$\mathcal{L}_\beta(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}) = -\frac{1}{M} \sum_{i=1}^{N} \sum_{n=1}^{N} \hat{A}_{\pi_{old}}^{iT} J^{in} x^i \quad (27)$$

$$+ \frac{1}{M} \sum_{e=1}^{L} \sum_{q \in \mathcal{N}(e)} \sum_{n=1}^{N} y_e^{qnT}(C_e^q J^{qn} x^q - z_e^{qn})$$

$$+ \frac{1}{M} \sum_{e=1}^{L} \sum_{q \in \mathcal{N}(e)} \sum_{n=1}^{N} \frac{\beta}{2} \| C_e^q J^q x^q - z_e^{qn} \|_2^2,$$

(1) We first consider the update of θ^q, which is transformed into the following problem

$$\min_{x^q \in X^q} \mathcal{L}_\beta(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}), \quad q \in \mathcal{N}(e). \quad (28)$$

This is a convex optimization problem with quadratic constraint. According to the strong duality theory, the optimal value p^* satisfies

$$p^* = \min_{x^q} \max_{\lambda^q > 0} \mathcal{L}_\beta(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}) + \lambda^q(\frac{1}{2} x^T \overline{H}^q x - \delta/N)$$

$$= \max_{\lambda^q > 0} \min_{x^q} \mathcal{L}_\beta(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}) + \lambda^q(\frac{1}{2} x^T \overline{H}^q x - \delta/N) \quad (29)$$

$$= \max_{\lambda^q > 0} \min_{x^q} f(x^q, \lambda^q)$$

Based on the first order condition with respect to x^q, the optimal solution x^{q*} should satisfy

$$\frac{\partial f(x^q, \lambda^q)}{\partial x^q}$$

$$= -\frac{1}{M} \sum_{n=1}^{N} J^{qnT} \hat{A}_{\pi_{old}}^q + \frac{1}{M} \sum_{e \in \mathcal{E}(q)} \sum_{n=1}^{N} C_e^q J^{qnT} y_e^{qn}$$

$$+ \frac{\beta}{M} \sum_{e \in \mathcal{E}(q)} \sum_{n=1}^{N} C_e^q J^{qnT}(C_e^q J^{qn} x^q - z_e^{qn}) + \lambda^q \overline{H}^q x^q$$

$$= \left(\lambda^q \overline{H}^q + \frac{|\mathcal{E}(q)|\beta}{M} \sum_{n=1}^{N} J^{qnT} J^{qn} \right) x^q -$$

$$\frac{1}{M} \sum_{n=1}^{N} J^{qnT} \left(\hat{A}_{\pi_{old}}^q - \sum_{e \in \mathcal{E}(q)} C_e^q y_e^{qn} + \beta \sum_{e \in \mathcal{E}(q)} C_e^q z_e^{qn(k)} \right)$$

$$= 0$$
$$(30)$$

where $|\mathcal{E}(q)|$ is the cardinality of $\mathcal{E}(q)$. Solving Eq. (30), we get

$$x^{q*} = \left(\lambda^q \overline{H}^q + \frac{|\mathcal{E}(q)|\beta}{M} \sum_{n=1}^{N} J^{qnT} J^{qn} \right)^{-1} V \quad (31)$$

where

$$V = \frac{1}{M} \sum_{n=1}^{N} J^{qnT}(\hat{A}_{\pi_{old}}^q - \sum_{e \in \mathcal{E}(q)} C_e^q y_e^{qn(k)} + \beta \sum_{e \in \mathcal{E}(q)} C_e^q z_e^{qn(k)}).$$

When $M \to \infty$, we have $\overline{H}^q = \frac{1}{M} \sum_{n=1}^{N} J^{qnT} J^{qn}$ because

$$\nabla_{\theta^i}^2 D_{KL}(\pi_{old}^i(\cdot|o^i; \theta_{old}^i) \| \pi^i(\cdot|o^i; \theta^i)) |_{\theta^i = \theta_{old}^i}$$

$$= \sum_{n=1}^{N} \nabla_{\theta^i}^2 D_{KL}(\pi_{old}^i(a^n|o^i; \theta_{old}^i) \| \pi^i(a^n|o^i; \theta^i)) |_{\theta^i = \theta_{old}^i}$$

$$= \sum_{n=1}^{N} \mathbb{E}_{a \sim \pi_{old}^i} \left[\nabla_{\theta^i} \log(\Upsilon^n(\theta^i)) \nabla_{\theta^i} \log(\Upsilon^n(\theta^i)) \right] |_{\theta^i = \theta_{old}^i}.$$
$$(32)$$

Therefore, the optimal solution x^{q*} can be simplified as

$$x^{q*} = \frac{1}{\lambda^q + |\mathcal{E}(q)|\beta} \overline{H}^{q-1} V \quad (33)$$

Substituting (33) into (29), we derive

$$
\begin{aligned}
p^* &= \max_{\lambda^q > 0} \ f(x^{q*}, \lambda^q) \\
&= \max_{\lambda^q > 0} \ -\frac{1}{M(\lambda^q + |\mathcal{E}(q)|\beta)} \sum_{n=1}^{N} \hat{A}^{q\,T}_{\pi_{old}} J^{qn} \overline{H}^{q-1} V \\
&+ \frac{1}{M} \sum_{e \in \mathcal{E}(q)} \sum_{n=1}^{N} y_e^{qnT} \left(\frac{1}{\lambda^q + |\mathcal{E}(q)|\beta} C_e^q J^{qn} \overline{H}^{q-1} V - z_e^{qn} \right) \\
&+ \frac{1}{M} \sum_{e \in \mathcal{E}(q)} \sum_{n=1}^{N} \frac{\beta}{2} \| \frac{1}{\lambda^q + |\mathcal{E}(q)|\beta} C_e^q J^{qn} \overline{H}^{q-1} V - z_e^{qn} \|_2^2 \\
&+ \lambda^q \left(\frac{1}{2(\lambda^q + |\mathcal{E}(q)|\beta)^2} V^{\mathrm{T}} \overline{H}^{q-1} V - \delta/N \right)
\end{aligned}
$$
(34)

Based on the first order condition, the optimal λ^q should satisfy

$$
\begin{aligned}
&\frac{\partial f(x^{q*}, \lambda^q)}{\partial \lambda^q} \\
&= \frac{1}{M(\lambda^q + |\mathcal{E}(q)|\beta)^2} \sum_{n=1}^{N} \hat{A}^{q\,T}_{\pi_{old}} J^{qn} \overline{H}^{q-1} V - \\
&\quad \frac{1}{M(\lambda^q + |\mathcal{E}(q)|\beta)^2} \sum_{e \in \mathcal{E}(q)} \sum_{n=1}^{N} C_e^q y_e^{qnT} J^{qn} \overline{H}^{q-1} V - \\
&\quad \frac{n(\mathcal{E}(q))\beta}{(\lambda^q + |\mathcal{E}(q)|\beta)^3} V^{\mathrm{T}} \overline{H}^{q-1} V + (\text{Use } \overline{H}^q = \frac{1}{M} \sum_{n=1}^{N} J^{qnT} J^{qn}) \\
&\quad \frac{\beta}{M(\lambda^q + |\mathcal{E}(q)|\beta)^2} \sum_{e \in \mathcal{E}(q)} \sum_{n=1}^{N} C_e^q V^{\mathrm{T}} \overline{H}^{q-1} J^{qnT} z_e^{qn} + \\
&\quad \frac{|\mathcal{E}(q)|\beta - \lambda^q}{2(\lambda^q + |\mathcal{E}(q)|\beta)^3} V^{\mathrm{T}} \overline{H}^{q-1} V - \delta/N \\
&= \left(\frac{1}{(\lambda^q + |\mathcal{E}(q)|\beta)^2} - \frac{1}{2(\lambda^q + |\mathcal{E}(q)|\beta)^2} \right) V^{\mathrm{T}} \overline{H}^{q-1} V - \delta/N \\
&\quad (\text{Use } a^{\mathrm{T}} \overline{H}^{q-1} b = b^{\mathrm{T}} \overline{H}^{q-1} a \text{ since } \overline{H}^{q-1} \text{ is symmetric}) \\
&= \frac{1}{2(\lambda^q + |\mathcal{E}(q)|\beta)^2} V^{\mathrm{T}} \overline{H}^{q-1} V - \delta/N \\
&= 0
\end{aligned}
$$
(35)

Solving Eq. (35), we get

$$
\lambda^{q*} = \sqrt{\frac{V^T \overline{H}^{q\,-1} V}{2\delta/N}} - |\mathcal{E}(q)|\beta.
$$
(36)

Substituting (36) into (33), the optimal solution x^{q*} is

$$
x^{q*} = \sqrt{\frac{2\delta/N}{V^T \overline{H}^{q\,-1} V}} \overline{H}^{q-1} V.
$$
(37)

Since $x^q = \theta^q - \theta^q_{old}$, the optimal update $\theta^{q(k+1)} := \arg\min_{\theta^q \in \Theta^q} \mathcal{L}_\beta(\theta, z^{(k)}, y^{(k)})$ satisfies

$$
\theta^{q(k+1)} = \theta^q_{old} + \sqrt{\frac{2\delta/N}{V^T \overline{H}^{q\,-1} V}} \overline{H}^{q-1} V.
$$
(38)

where

$$
V = \frac{1}{M} \sum_{n=1}^{N} J^{qn\mathrm{T}} (\hat{A}^q_{\pi_{old}} - \sum_{e \in \mathcal{E}(q)} C_e^q y_e^{qn(k)} + \beta \sum_{e \in \mathcal{E}(q)} C_e^q z_e^{qn(k)}).
$$

(2) Next, we consider the update of z_e^{qn}:

$$
\min_{z_e^{qn} \in Z_e} \mathcal{L}_\beta(x, z, y), \ q \in \mathcal{N}(e).
$$
(39)

where $Z_e = \{z_e^{in}, z_e^{jn} | z_e^{in} + z_e^{jn} = 0, q = (i, j) \in \mathcal{N}(e), n \in \mathcal{N}\}$. This is a convex optimization problem. The optimal value p^* satisfies

$$
p^* = \min_{z_e^{in}, z_e^{jn}} \max_{\nu_e^n > 0} \mathcal{L}_\beta(x, z, y) + \nu_e^{nT}(z_e^{in} + z_e^{jn}).
$$
(40)

Based on the first order optimality conditions, the optimal z_e^{in}, z_e^{jn} should satisfy:

$$
-\frac{1}{M} y_e^{in} - \frac{1}{M}\beta \left(C_e^i J^{in} x^i - z_e^{in} \right) + \nu_e^n = 0
$$
(41)

$$
-\frac{1}{M} y_e^{jn} - \frac{1}{M}\beta \left(C_e^j J^{jn} x^j - z_e^{jn} \right) + \nu_e^n = 0
$$
(42)

Solving Eqs. (41) and (42), we get

$$
z_e^{in*} = \frac{1}{\beta}(y_e^{in} - M\nu_e^n) + C_e^i J^{in} x^i,
$$
(43)

$$
z_e^{jn*} = \frac{1}{\beta}(y_e^{jn} - M\nu_e^n) + C_e^j J^{jn} x^j.
$$
(44)

Since z_e^{in*}, z_e^{jn*} should satisfy $z_e^{in*} + z_e^{jn*} = 0$, we have

$$
\frac{1}{\beta}(y_e^{in} + y_e^{jn}) - \frac{2M}{\beta}\nu_e^n + (C_e^i J^{in} x^i + C_e^j J^{jn} x^j) = 0.
$$
(45)

Therefore, the optimal dual variable ν_e^{n*} is

$$
\begin{aligned}
\nu_e^{n*} &= \frac{1}{2M}(y_e^{in} + y_e^{jn}) + \frac{\beta}{2M}(C_e^i J^{in} x^i + C_e^j J^{jn} x^j) \\
&= \frac{1}{2M} \sum_{q \in \mathcal{N}(e)} [y_e^{qn} + \beta C_e^q J^{qn} x^q].
\end{aligned}
$$
(46)

Since $x^q = \theta^q - \theta^q_{old}$, the optimal update $z_e^{qn(k+1)} := \arg\min_{z_e^{qn} \in Z_e} \mathcal{L}_\beta(\theta^{(k+1)}, z, y^{(k)})$ satisfies

$$
z_e^{qn(k+1)} := \frac{1}{\beta}(y_e^{qn(k)} - \nu_e^n) + C_e^q J^{qn}(\theta^{q(k+1)} - \theta^q_{old}),
$$
(47)

where

$$
\nu_e^n = \frac{1}{2} \sum_{q \in \mathcal{N}(e)} \left[y_e^{qn(k)} + \beta C_e^q J^{qn}(\theta^{q(k+1)} - \theta^q_{old}) \right].
$$
(48)

(3) Substituting $z_e^{qn(k+1)}$ into the update

$$
y_e^{qn(k+1)} := y_e^{qn(k)} + \beta(C_e^q J^{qn}(\theta^{q(k+1)} - \theta^q_{old}) - z_e^{qn(k+1)}),
$$
(49)

we get

$$
y_e^{qn(k+1)} = \nu_e^n.
$$
(50)

$\square$

REFERENCES

[1] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1889–1897. [Online]. Available: http://proceedings.mlr.press/v37/schulman15.html

[2] P. Peng, Y. Wen, Y. Yang, Q. Yuan, Z. Tang, H. Long, and J. Wang, "Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games," *ArXiv pre-prints*, 2017, arXiv:1703.10069.

[3] D. Li, D. Zhao, Q. Zhang, and Y. Chen, "Reinforcement learning and deep learning based lateral control for autonomous driving," *ArXiv pre-prints*, 2018, arXiv:1810.12778.

[4] L. Matignon, L. Jeanpierre, and A.-I. Mouaddib, "Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes," in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, ser. AAAI'12. AAAI Press, 2012, p. 2017–2023.

[5] X. Wang, L. Ke, Z. Qiao, and X. Chai, "Large-scale traffic signal control using a novel multiagent reinforcement learning," *IEEE Transactions on Cybernetics*, pp. 1–14, 2020.

[6] M. Pipattanasomporn, H. Feroze, and S. Rahman, "Multi-agent systems in a distributed smart grid: Design and implementation," in *2009 IEEE/PES Power Systems Conference and Exposition*, 2009, pp. 1–8.

[7] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," in *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, ser. AAAI '98/IAAI '98. USA: American Association for Artificial Intelligence, 1998, p. 746–752.

[8] M. Tan, *Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, p. 487–494.

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

[10] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6382–6393.

[11] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '18. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2018, p. 2085–2087.

[12] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 2961–2970. [Online]. Available: http://proceedings.mlr.press/v97/iqbal19a.html

[13] R. Cissée and S. Albayrak, "An agent-based approach for privacy-preserving recommender systems," in *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '07. New York, NY, USA: Association for Computing Machinery, 2007. [Online]. Available: https://doi.org/10.1145/1329125.1329345

[14] Minghua He, N. R. Jennings, and Ho-Fung Leung, "On agent-mediated electronic commerce," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 4, pp. 985–1003, 2003.

[15] J. M. Such, A. Espinosa, and A. García-Fornes, "A survey of privacy in multi-agent systems," *The Knowledge Engineering Review*, vol. 29, no. 3, p. 314–344, 2014.

[16] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," *ArXiv pre-prints*, 2019, arXiv:1709.06560.

[17] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *ArXiv pre-prints*, 2019, arXiv:1911.10635.

[18] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3826–3839, 2020.

[19] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 1st ed. Athena Scientific, 1995.

[20] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, ser. NIPS'99. Cambridge, MA, USA: MIT Press, 1999, p. 1057–1063.

[21] S. Valcarcel Macua, J. Chen, S. Zazo, and A. H. Sayed, "Distributed policy evaluation under multiple behavior strategies," *IEEE Transactions on Automatic Control*, vol. 60, no. 5, pp. 1260–1274, 2015.

[22] D. Lee, H. Yoon, and N. Hovakimyan, "Primal-dual algorithm for distributed reinforcement learning: Distributed gtd," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 1967–1972.

[23] A. Mathkar and V. S. Borkar, "Distributed reinforcement learning via gossip," *IEEE Transactions on Automatic Control*, vol. 62, no. 3, pp. 1465–1470, 2017.

[24] H.-T. Wai, Z. Yang, Z. Wang, and M. Hong, "Multi-agent reinforcement learning via double averaging primal-dual optimization," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 9672–9683.

[25] T. T. Doan, S. T. Maguluri, and J. Romberg, "Finite-time analysis of distributed td (0) with linear function approximation on multi-agent reinforcement learning," in *ICML*, 2019.

[26] S. Kar, J. M. F. Moura, and H. V. Poor, "$\mathcal{QD}$-learning: A collaborative distributed strategy for multi-agent reinforcement learning through consensus $+$ innovations," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1848–1862, 2013.

[27] X. Liu and Y. Tan, "Attentive relational state representation in decentralized multiagent reinforcement learning," *IEEE Transactions on Cybernetics*, pp. 1–13, 2020.

[28] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 5872–5881. [Online]. Available: http://proceedings.mlr.press/v80/zhang18n.html

[29] Y. Zhang and M. M. Zavlanos, "Distributed off-policy actor-critic reinforcement learning with policy consensus," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 4674–4679.

[30] W. Li, B. Jin, X. Wang, J. Yan, and H. Zha, "F2a2: Flexible fully-decentralized approximate actor-critic for cooperative multi-agent reinforcement learning," *ArXiv pre-prints*, 2020, arXiv:2004.11145.

[31] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, ser. ICML'94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, p. 157–163.

[32] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, p. 1–122, Jan. 2011. [Online]. Available: https://doi.org/10.1561/2200000016

[33] E. Wei and A. Ozdaglar, "On the o(1=k) convergence of asynchronous distributed alternating direction method of multipliers," in *2013 IEEE Global Conference on Signal and Information Processing*, 2013, pp. 551–554.

[34] S. Magnússon, P. C. Weeraddana, and C. Fischione, "A distributed approach for the optimal power-flow problem based on admm and sequential convex approximations," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 3, pp. 238–253, 2015.

[35] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, p. 22–31.

[36] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," *ArXiv pre-prints*, 2018, arXiv:1703.04908.

[37] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 971–980. [Online]. Available: http://papers.nips.cc/paper/6698-self-normalizing-neural-networks.pdf