Optimization for Reinforcement Learning: From Single Agent to Cooperative Agents

Donghwan Lee[1], Niao He[2], Parameswaran Kamalaruban[3], and Volkan Cevher[3]

[1]Korea Advanced Institute of Science and Technology (KAIST), donghwan@kaist.co.kr.

[2]University of Illinois at Urbana-Champaign (UIUC), niaohe@illinois.edu.

[3]École Polytechnique Fédérale de Lausanne (EPFL), volkan.cevher@epfl.ch.

I. Introduction

Fueled with recent advances in deep neural networks, reinforcement learning (RL) has been in the limelight for many recent breakthroughs in artificial intelligence, including defeating humans in games (e.g., chess, Go, StarCraft), self-driving cars, smart home automation, service robots, among many others. Despite these remarkable achievements, many basic tasks can still elude a single RL agent. Examples abound from multi-player games, multi-robots, cellular antenna tilt control, traffic control systems, smart power grids to network management.

Often, cooperation among multiple RL agents is much more critical: multiple agents must collaborate to complete a common goal, expedite learning, protect privacy, offer resiliency against failures and adversarial attacks, and overcome the physical limitations of a single RL agent behaving alone. These tasks are studied under the umbrella of *cooperative multi-agent RL (MARL)*, where agents seek to learn optimal policies to maximize a shared team reward, while interacting with an unknown stochastic environment and with each other. Cooperative MARL is far more challenging than the single-agent case due to: i) the exponentially growing search space, ii) the non-stationary and unpredictable environment caused by the agents' concurrent yet heterogeneous behaviors, and iii) the lack of *central* coordinators in many applications. These difficulties can be alleviated by appropriate coordination among agents.

The cooperative MARL can be further categorized into subclasses depending on the information structure and types of coordination, such as how much information (e.g., state, action, reward, etc.) is available for each agent, what kinds of information can be shared among the agents, and what kinds of protocols (e.g., communication networks, etc.) are used for coordination. When only local partial state observation is available for each agent, the corresponding multi-agent systems are often described through decentralized partially observable Markov decision processes (MDP), or DEC-POMDP for short,

for which the decision problem is known to be extremely challenging. In fact, even the planning problem of DEC-POMDPs (with known models) is known to be NEXT-complete [1]. Despite some recent empirical successes [2]–[4], finding an exact solution of DEC-POMDPs using RLs with theoretical guarantees remains an open question.

When full state information is available for each agent, we call agents *joint action learners* (JALs) if they also know the joint actions of other agents, and *independent learners* (ILs) if agents only know their own actions. Learning tasks for ILs are still very challenging, since each agent sees other agents as parts of the environment, so without observing the internal states, including other agents actions, the problem essentially becomes non-Markovian [5] and a partially observable MDP (POMDP). It turns out that optimal policy can be found under restricted assumptions such as deterministic MDP [6], and for general stochastic MDPs, several attempts have demonstrated empirical successes [7]–[9]. For a more comprehensive survey on independent MARLs, the reader is referred to the survey [6].

The form of rewards, either centralized or decentralized, also makes a huge difference in multi-agent systems. If every agent receives a common reward, the situation becomes relatively easy to deal with. For instance, JALs can perfectly learn exact optimal policies of the underlying decision problem even without coordination among agents [10]. The more interesting and practical scenario is when rewards are decentralized, i.e., each agent receives its own local reward while the global reward to be maximized is the sum of local rewards. This decentralization is especially important when taking into account the privacy and resiliency of the system.

Clearly, learning without coordination among agents is impossible under decentralized rewards. This article focuses on this important subclass of cooperative MARL with decentralized rewards, assuming the full state and action information is available to each agent. In particular, we consider decentralized coordination through network communications characterized by graphs, where each node in the graph represents each agent and edges connecting nodes represent communication between them.

Distributed optimization rises to the challenge by achieving global consensus on the optimal policy through only local computation and communication with neighboring agents. Recently, several important advances have been made in this direction such as the distributed temporal difference (TD) learning [11], distributed Q-learning [12], distributed actor-critic algorithm [13], and other important results [14]–[17]. These works largely benefit from the synergistic connection between RLs and the core idea of averaging consensus-based distributed optimization [18], which leverages averaging consensus protocols for information propagation over networks and rich theory established in this field during the last decade.

In this survey, we provide an overview of this emerging field with an emphasis on optimization within the decentralized setting (decentralized rewards and decentralized communication protocols). For this

purpose, we highlight the evolution of RL algorithms from single-agent to multi-agent systems, from a distributed optimization perspective, in the hope to catalyze the growing synergy among distributed optimization, signal processing, and RL communities.

In the sequel, we first revisit the basics of single-agent RL in Section II and extend to multi-agent RL in Section III. In Section IV, we provide preliminaries of distributed optimization as well as consensus algorithms. In Section V, we discuss several important consensus-based MARL algorithms with decentralized network communication protocols. Finally, in Section VI, we conclude with future directions and open issues. Note that our review is not exhaustive given the magazine limits; we suggest the interested reader to further read [6], [19], [20].

II. Single-agent RL basics

To understand MARL, it is imperative that we briefly review the basics of single-agent RL setting, where only a single agent interacts with an unknown stochastic environment. Such environments are classically represented by a Markov decision process: $\mathcal{M} := (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where the state-space $\mathcal{S} := \{1, 2, \ldots, |\mathcal{S}|\}$ and action-space $\mathcal{A} := \{1, 2, \ldots, |\mathcal{A}|\}$, upon selecting an action $a \in \mathcal{A}$ with the current state $s \in \mathcal{S}$, the state transits to $s' \in \mathcal{S}$ according to the state transition probability $P(s'|s, a)$, and the transition incurs a random reward $r(s, a)$. For simplicity, we consider the infinite-horizon (discounted) Markov decision problem (MDP), where the agent sequentially takes actions to maximize cumulative discounted rewards. The goal is to find a deterministic policy $\pi^* : \mathcal{S} \to \mathcal{A}$, or a stochastic policy $\pi^* : \mathcal{S} \to \Delta_{\mathcal{A}}$, optimal policy, where $\Delta_{\mathcal{A}}$ is the set of all probability distributions over $\mathcal{A}$, such that the cumulative discounted rewards over infinite time horizons is maximized, i.e.,

$$\pi^* := \arg\max_{\pi \in \Theta} \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r(s_k, \pi(s_k))\right], \tag{1}$$

where $\gamma \in [0, 1)$ is the discount factor, Θ is the set of all admissible deterministic policies, and $(s_0, a_0, s_1, a_1, \ldots)$ is a state-action trajectory generated by the Markov chain under policy π. Solving MDPs involves two key concepts associated with the expected return:

1) $V^{\pi}(s) := \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r(s_k, \pi(s_k))|s_0 = s\right]$ is called the (state) value function for a given policy π, which encodes the expected cumulative reward when starting in the state s, and then, following the policy π thereafter.
2) $Q^{\pi}(s, a) := \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r(s_k, \pi(s_k))|s_0 = s, a_0 = a\right]$ is called the state-action value function or Q-function for a given policy π, which measures the expected cumulative reward when starting from state s, taking the action a, and then, following the policy π.

Their optima over all possible policies are defined by $V^*(s) := \max_{\pi:\mathcal{S}\to\mathcal{A}} V^\pi(s) = \max_a Q^*(s,a)$ and $Q^*(s,a) := \max_{\pi:\mathcal{S}\to\mathcal{A}} Q^\pi(s,a)$, respectively. Given the optimal value functions Q^* or V^*, the optimal policy π^* can be obtained by picking an action a that is greedy with respect to V^* or Q^*, i.e., $\pi^*(s) = \arg\max_a \mathbb{E}_{s'\sim P(\cdot|s,a)}[r(s,a) + \gamma V^*(s')]$ or $\pi^*(s) = \arg\max_a Q^*(s,a)$, respectively. When the MDP instance, $\mathcal{M}$, is known, then it can be solved efficiently via dynamic programming (DP) algorithms. Based on the Markov property, the value function V^π for a given policy π, satisfies the Bellman equation: $V^\pi(s) = \mathbb{E}_{s'\sim P(\cdot|s,\pi(s))}[r(s,\pi(s)) + \gamma V^\pi(s')]$. The similar property holds for Q^π as well. Moreover, the optimal Q-function Q^*, satisfies the *Bellman optimality equation*, $Q^*(s,a) = \mathbb{E}_{s'\sim P(\cdot|s,a)}[r(s,a) + \max_{a'} \gamma Q^*(s',a')]$. Various DP algorithms, such as the policy and value iterations, are obtained by turning the Bellman equations into different update rules.

A. Classical RL Algorithms

RLs can be categorized into two main groups, the policy evaluation algorithms and the policy optimization algorithms. The former group addresses the problem of evaluating the value function given a policy π, while the latter group deals with finding an optimal policy. For both groups, many classical RL algorithms can be viewed as stochastic variants of DPs. This insight will be key for scaling MARL in the sequel. In particular, the temporal-difference (TD) learning falls into the policy evaluation group and is one of the most fundamental policy evaluation RL algorithm:

$$V_{k+1}(s_k) = V_k(s_k) + \alpha_k(r(s_k,\pi(s_k)) + \gamma V_k(s_{k+1}) - V_k(s_k)), \tag{2}$$

where $s_k \sim d^\pi$, $s_{k+1} \sim P(\cdot|s_k,\pi(s_k))$, α_k is the learning rate (or step-size), and d^π denotes the stationary state distribution under policy π, namely, $d^\pi(s) = \lim_{k\to\infty} \mathbb{P}[s_k = s|\pi]$. For any fixed policy π, TD update converges to V^π almost surely if the step-size satisfies the so-called *Robbins-Monro rule*, $\sum_{k=0}^\infty \alpha_k = \infty$, $\sum_{k=0}^\infty \alpha_k^2 < \infty$ [21]. Although theoretically sound, the naive TD learning is only applicable to small-scale problems as it needs to store and enumerate values of all states. However, most practical problems we face in the real-world have large state-space. In such cases, enumerating all values in a table is numerically inefficient or even intractable.

Using function approximations resolves this problem by encoding the value function with a parameterized function class, $V(\cdot) \cong V(\cdot;\theta)$. The simplest example is the linear function approximation, $V(\cdot;\theta) = \Phi\theta$, where $\Phi = [\phi(1);\cdots;\phi(|\mathcal{S}|)]^\top \in \mathbb{R}^{|\mathcal{S}|\times n}$ is a feature matrix, and $\phi : \mathcal{S} \to \mathbb{R}$ is a pre-selected feature mapping. With the linear function approximation, TD learning can be written as

$$\theta_{k+1} = \theta_k + \alpha_k(r(s_k,\pi(s_k)) + \gamma\phi(s_{k+1})^T\theta_k - \phi(s_k)^T\theta_k)\phi(s_k). \tag{3}$$

The above update is known to converge to θ^* almost surely [22], where θ^* is the solution to the *projected Bellman equation*, $\Phi\theta = \Pi(R_\pi + \alpha P_\pi \Phi\theta)$, where R_π is the expected reward vector under policy π and P^π is the state transition probability matrix under policy π, and $\Pi := \Phi(\Phi^T D\Phi)^{-1}\Phi^T D$ is the projection onto the range space of Φ, provided that the Markov chain with transition matrix P^π is ergodic and the step-size satisfies the Robbins-Monro rule. Note that the projection corrects the mismatch between the linear value function approximation on the left-hand side and the Bellman operator on the right-hand side which may lie outside of the linear span of columns of the feature matrix Φ.

Finite sample analysis of the TD learning algorithm is only recently established in [23]–[25]. Besides the standard TD, there also exists a wide spectrum of TD variants in the literature [26]–[29]. Note that when a nonlinear function approximation, such as neural networks, is used, these algorithms are not guaranteed to converge.

The policy optimization methods aim to find the optimal policy π^* and broadly fall under two camps, with one focusing on *value-based* updates, and the other focusing on direct *policy-based* updates. There is also a class of algorithms that belong to both camps, called actor-critic algorithms. Q-learning is one of the most representative valued-based algorithms, which obeys the update rule

$$Q_{k+1}(s_k, a_k) = Q_k(s_k, a_k) + \alpha_k(r(s_k, a_k) + \gamma \max_{a\in\mathcal{A}} Q_k(s_{k+1}, a) - Q_k(s_k, a_k)), \tag{4}$$

where $s_k \sim d^{\pi^b}$, d^{π^b} is the stationary distribution vector under π^b, $s_{k+1} \sim P(\cdot|s_k, \pi^b(s_k))$, $a_k \sim \pi^b$, and π^b is called the behavior policy, which refers to the policy used to collect observations for learning. The algorithm converges to Q^* almost surely [30] provided that the step-size satisfies the Robbins-Monro rule, and every state is visited infinitely often. Unlike value-based methods, direct policy search methods optimize a parameterized policy π_θ from trajectories of the state, action, reward, (s, a, r) without any value function evaluation steps, using the following (stochastic) gradient steps:

$$\theta_{k+1} = \theta_k + \alpha_k \hat{\nabla}_\theta J(\theta_k), \text{ where } J(\theta) := \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{\pi_\theta}(s_k)\right], \tag{5}$$

where $\hat{\nabla}_\theta J(\theta_k)$ is a stochastic estimate of the gradient evaluated at θ_k. The gradient of the value function has the simple analytical form $\nabla J(\theta) = \mathbb{E}_{s\sim d_{\pi_\theta}, a\sim\pi_\theta}[\nabla \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a)]$, which, however, needs an estimate of the Q-function, $Q^{\pi_\theta}(s, a)$. The simple policy gradient method replaces $Q^{\pi_\theta}(s, a)$ with a Monte Carlo estimate, which is called REINFORCE [31]. However, the *high variance* of the stochastic gradient estimates due to the Monte Carlo procedure often leads to slow and sometimes unstable convergence. The actor-critic methods combine the advantages of the value-based and direct policy search methods [32]

to reduce the variance. These algorithms parameterize both the policy and the value functions, and simultaneously update both in training

$$\text{Critic update}: w_{k+1} = w_k + \alpha_k(r(s_k, a_k) + \gamma Q(s_{k+1}, a_{k+1}; w_k) - Q(s_k, a_k; w_k))\nabla_w Q(s_k, a_k; w_k) \tag{6}$$

$$\text{Actor update}: \theta_{k+1} = \theta_k + \beta_k Q(s_k, a_k; w_k)\nabla_\theta \log \pi(a_k|s_k; \theta_k), \tag{7}$$

where w_k and θ_k are parameters of the value and policy, respectively, $a_k \sim \pi(\cdot|s_k; \theta_k)$, $a_{k+1} \sim \pi(\cdot|s_{k+1}; \theta_k)$, and the next state s_{k+1} is sampled under the current policy $\pi(\cdot|\cdot; \theta_k)$. Roughly speaking, it consists of two simultaneous and independent iterations, the value evaluation in (6), which tries to evaluate the value of the current policy through the TD steps, and the policy improvement, which tries to find a policy that improves the value through gradient steps. They often exhibit better empirical performance than value-based or direct policy-based methods alone. Nonetheless, when (nonlinear) function approximation is used, the convergence guarantees of all these algorithms remain rather elusive.

B. Modern Optimization-based RL Algorithms

Although the MDP given in (1) is itself a multistage stochastic optimization problem, most classical RLs introduced in the previous subsection, except for the policy gradient methods, are based on solving the Bellman equation and fixed point algorithms, which are rather different from standard gradient-based optimization algorithms. In this paper, we especially focus on the newly developed class of optimization-based RLs in the literature that hinges upon alternative (static and mostly convex) optimization reformulations and stochastic-gradient-type of methods.

Leveraging these optimization reformulations of RLs, recent works (see, e.g., [26], [28], [29], [33]–[35]) generate new principles for solving RL problems as we transition from linear towards nonlinear function approximations as well as establish theoretical guarantees based on rich theory in mathematical optimization literature.

Compared to the classical RL approaches, these optimization-based RLs exhibit several key advantages. First, in many applications such as robot control, the agents' behaviors are required to mediate among multiple different objectives. Sometimes, those objectives can be formulated as constraints, e.g., safety constraints. In this respect, optimization-based approaches are more extensible than the traditional dynamic programming-based approaches when dealing with policy constraints. Second, existing optimization theory provides ample opportunities in developing convergence analysis for RLs with and without function approximations; see, e.g., [33], [34]. More importantly, these methods are highly generalizable to the multi-agent RL setup with decentralized rewards, when integrated with recent fruitful advances made in distributed optimization. This last aspect is our main focus in this survey.

To build up an understanding, we first recall the linear programming (LP) formulation of the *planning* problem [36]

$$\min_{V} \quad \mathbb{E}_s[V(s)] \quad \text{subject to} \quad \mathbb{E}_{r,s'}[r(s,a)+\gamma V(s')|s] \leq V(s), \quad \forall s \in \mathcal{S}, \quad a \in \mathcal{A},$$

or equivalently,

$$\min_{V} \quad \mu^T V \quad \text{subject to} \quad R_a + \gamma P_a V \leq V, \quad \forall a \in \mathcal{A}, \tag{8}$$

where μ is the initial state distribution, $R_a \in \mathbb{R}^{|\mathcal{S}|}$ is the expected reward vector given action a, and $P_a \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ is the state transition probability matrix given action a. It is known that the solution to (8) is the optimal state-value function V^*, while the optimal policy can be recovered from V^* provided that the model is known. Another interesting relation between the optimal value function and policy can be derived from the concept of the duality. From the fundamental theory of convex optimization, the formulation (8) can be equivalently converted to another form, called the (Lagrangian) dual problem. In particular, the optimal value function and optimal policy can be found through solving the min-max problem:

$$\begin{aligned} \min_{V\in\mathcal{V}} \max_{\lambda=(\lambda_a)_{a\in\mathcal{A}}\in\Lambda} L(V,\lambda) :=& \mu^T V + \sum_{a\in\mathcal{A}} \lambda_a^T (R_a + \gamma P_a V - V), \\ =& \mathbb{E}_s[V(s)] + \sum_{s\in\mathcal{S}, a\in\mathcal{A}} \lambda_a(s) \mathbb{E}_{s,r,s'}[r(s,a)+\gamma V(s') - V(s)] \end{aligned} \tag{9}$$

where L is called the Lagrangian function, and sets $\mathcal{V}$ and Λ are properly chosen domains that restrict on the optimal value function and policy. Here, the variable V from the original optimization is called the *primal variable*, while the newly introduced variable, $\lambda := (\lambda_a)_{a\in\mathcal{A}}$, is called the *dual variable* (or Lagrangian multiplier). The optimal solutions, V^* and λ^*, of the min-max problem are called the optimal primal and dual solutions, respectively, and the primal optimal solution V^* of the min-max problem is identical to the original optimization (8). Here, the optimal dual solution is key to the planning problem [36]. In particular, the dual optimal solution yields the optimal policy from the identity

$$\pi^*(a,s) = \frac{\lambda_a^*(s)}{\sum_{a'\in\mathcal{A}} \lambda_{a'}^*(s)},$$

where $\pi^*(a,s)$ is the probability of taking action a at the state s under the optimal stochastic policy.

Building on this min-max formulation, several recent works introduce efficient RL algorithms for finding the optimal policy. For instance, the stochastic primal-dual RL (SPD-RL) in [33] solves the min-max problem (9) with the stochastic primal-dual algorithm

$$V_{k+1} = \Pi_{\mathcal{V}}(V_k - \gamma_k \hat{\nabla}_V L(V_k,\lambda_k)), \quad \lambda_{k+1} = \Pi_{\Lambda}(\lambda_k + \gamma_k \hat{\nabla}_\lambda L(V_k,\lambda_k)),$$

where $\hat{\nabla}_V L$ and $\hat{\nabla}_\lambda L$ are unbiased stochastic estimations of the gradients

$$\nabla_V L(V,\lambda) = \mu + \sum_{a\in\mathcal{A}} (P_a - I)\lambda_a, \quad \nabla_\lambda L(V,\lambda) = \sum_{a\in\mathcal{A}} (R_a + P_a V - V),$$

which are obtained by using samples of (s,a,r,s'), $\Pi_{\mathcal{V}}$ and Π_Λ stand for the projection operators onto the sets $\mathcal{V}$ and Λ. The main idea of the primal-dual algorithm is to take the stochastic gradient descent step with respect to the primal variable V, while taking the stochastic gradient ascent step with respect to the dual variable λ. Under mild conditions such as convexity and concavity, the stochastic primal-dual algorithm for general min-max problems is known to converge to an optimal solution.

Since these gradients are obtained based on the samples, the updates can be executed without the model knowledge. The SPD Q-learning in [35] extends it to the Q-learning framework with off-policy learning, where the sample observations are collected from some time-varying behavior policies. The dual actor-critic in [37] generalizes the setup to continuous state-action MDP and exploits nonlinear function approximations for both value function and the dual policy. The primal-dual algorithm to solve the min-max optimization (9) is closely related to the classical actor-critic algorithm in the sense that both approaches simultaneously update the parameters of the value function and policy. However, these algorithms are apparently different because the classical actor-critic algorithm does not try to solve the min-max problem (9). In particular, the classical actor-critic algorithm consists of two simultaneous and independent iterations, the value evaluation in (6), which tries to evaluate the value of the current policy through the TD steps, and the policy improvement, which tries to find a policy which improves the value through gradient steps.

Apart from the LP formulation, alternative nonlinear optimization frameworks based on the fixed point interpretation of Bellman equations have also been explored, both for policy evaluation and policy optimization. To name a few, Baird's residual gradient algorithm [38], designed for policy evaluation, aims for minimizing the mean-squared Bellman error, i.e.,

$$\min_\theta \ \text{MSBE}(\theta) := \mathbb{E}_s[(\mathbb{E}_{s'}[r(s,\pi(s)) + \gamma\phi^T(s')\theta] - \phi^T(s)\theta)^2] = \min_\theta \|R_\pi + \gamma P_\pi \Phi\theta - \Phi\theta\|_D^2, \quad (10)$$

where R_π and P_π are the expected reward vector and state transition probability matrix under policy π, respectively, Φ is the feature matrix, D is a diagonal matrix with diagonal entries being the stationary state distributions, and $\|x\|_D := \sqrt{x^T D x}$. However, directly minimizing the optimization objective (10) can be challenging due to the double sampling issue. Here, the double sampling issue means the requirement of double samples of the next stats from the current state to obtain an unbiased stochastic estimate of gradients of the objective mainly due to its quadratic nonlinearity.

The gradient TD (GTD) [26] solves the projected Bellman equation by minimizing the mean-square projected Bellman error,

$$\min_{\theta} \text{MSPBE}(\theta) := \|\Pi(R_\pi + \gamma P_\pi \Phi\theta) - \Phi\theta\|_D^2 , \tag{11}$$

where Π is the projection onto the range of the feature matrix Φ. This is driven by the fact that most TD learning algorithms converge to the minimum of MSPBE. To avoid the double sampling issue, GTD uses a stochastic primal-dual algorithm [39] for solving the corresponding min-max problem of the Lagrangian

$$\min_{\lambda}\max_{\theta} L(\lambda,\theta) := \theta^T\Phi^T D\left(\frac{1}{2}\Phi\lambda + (I-\gamma P_\pi)\Phi\theta - R_\pi\right). \tag{12}$$

Alternatively, [28], [40] get around this difficulty by resorting to min-max reformulations of the MSBE and MSBPE and introduce primal-dual type methods for policy evaluation with finite sample analysis. Similar ideas have also been employed for policy optimization based on the (softmax) Bellman optimality equation; see, e.g., [34] (called Smoothed Bellman Error Embedding (SBEED) algorithm).

III. From single-agent to multi-agent RLs

Cooperative MARL extends the single-agent RL to N agents, $\mathcal{V} = \{1, 2, \ldots, N\}$, where the system's behavior is influenced by the whole team of simultaneously and independently acting agents in a common environment. This can be further classified into MARLs with centralized rewards and decentralized rewards.

A. MARL with Centralized Rewards

We start with MARLs with centralized rewards, where all agents have access to a central reward. In this setting, a multi-agent MDP can be characterized by the tuple, $(\mathcal{S}, \{\mathcal{A}^i\}_{i=1}^N, P, r, \gamma)$, where $\mathcal{A}^i$ is a discrete action-space of agent i. Each agent i observes the common state s and executes action $a^i \in \mathcal{A}^i$ inside its own action set $\mathcal{A}^i$ according to its local policy $\pi^i : \mathcal{S} \to \mathcal{A}^i$. The joint action $a := (a^1, a^2, \ldots, a^N) \in \mathcal{A} := \mathcal{A}^1 \times \cdots \times \mathcal{A}^N$ causes the state $s \in \mathcal{S}$ to transit to $s' \in \mathcal{S}$ with probability $P(s'|s,a)$, and the agent receives the common reward $r(s,a)$. The goal for each agent is to learn a local policy $\pi_*^i : \mathcal{S} \to \mathcal{A}^i, i \in \mathcal{V}$ such that $(\pi_*^1, \pi_*^2, \ldots, \pi_*^N) =: \pi^*$ is an optimal central policy.

The MARL in this scenario heavily depends on the degree of coordination, information structure, and various assumptions, such as how much information is available for each agent, what kinds of information can be shared among the agents, and what kinds of protocols are used for coordination.

The main information structure is the availability of the joint action a for each agent. If the joint action is available, the agents are called the *joint action learners (JAL)*, i.e., each agent has access to (s, a, r).

Otherwise, the agents are called the *independent learners (IL)*, i.e., each agent has access to (s, a^i, r). In the JAL case, MA-MDP can be regarded as a variant of the standard MDP under a special action set. The IL case hence is much more challenging because the joint action information is unavailable.

Suppose each agent $i \in \mathcal{V}$ receives the central reward r and knows the joint state and action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$ (i.e., agents are JALs). Cooperative MARL, in this case, is straightforward because all agents have full information to find an optimal solution. As an example, a naive application of the Q-learning [41] to multi-agent settings is

$$Q^i_{k+1}(s_k, a_k) = Q^i_k(s_k, a_k) + \alpha_k \left\{ r(s_k, a_k) + \gamma \max_{a \in \mathcal{A}} Q^i_k(s_{k+1}, a) - Q^i_k(s_k, a_k) \right\},$$

where each agent keeps its local Q-function $Q^i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. In particular, it is equivalent to the single-agent Q-learning executed by each agent in parallel, and $Q^i_k \to Q^*$ as $k \to \infty$ almost surely for all $i \in \mathcal{V}$; thereby $\pi^i_k(\cdot) = \arg\max_a Q^i_k(\cdot, a) \to \pi^i_*(\cdot)$ as $k \to \infty$. Similarly, the policy search methods and actor-critic methods can be easily generalized to MARL with JALs [42]. In such a case, coordination among agents is unnecessary to learn the optimal policy. However, in practice, each agent may not have access to the global rewards due to limitations of communication or privacy issues; as a result, coordination protocols are essential for achieving the optimal policy corresponding to the global reward.

B. Networked MARL with Decentralized Reward

The main focus of this survey is on MARLs with decentralized rewards, where each agent only receives a local reward, and the central reward function is characterized as the average of all local rewards. The goal of each agent is to cooperatively find an optimal policy corresponding to the central reward by sharing local learning parameters over a communication network.

More formally, a coordinated multi-agent MDP with a communication network (i.e., networked MA-MDP) is given as the tuple, $(\mathcal{S}, \{\mathcal{A}^i\}_{i=1}^N, P, \{r^i\}_{i=1}^N, \gamma, \mathcal{G})$, where $r^i(s, a)$ is the random reward of agent i given action a and the current state s, and $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is an undirected graph (possibly time-varying or stochastic) characterizing the communication network. Each agent i observes the common state s, executes its local action $a^i \in \mathcal{A}^i$ according to its local policy $\pi^i : \mathcal{S} \to \mathcal{A}^i$, receives the local reward $r^i(s, a)$, and the joint action $a := (a^1, a^2, \ldots, a^N)$ causes the state $s \in \mathcal{S}$ to transit to $s' \in \mathcal{S}$ with probability $P(s'|s, a)$. The central reward is defined as $r = \frac{1}{N} \sum_{i=1}^N r^i$. In the course of learning, each agent receives learning parameters $\{\theta^j\}_{j \in \mathcal{N}_i}$ from its neighbors of the communication network. The overall model is illustrated as in Figure 1.

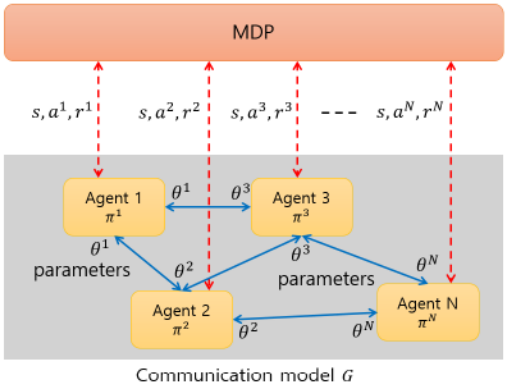

Fig. 1. Coordinated multi-agent MDP with communication network

For an illustrative example, we consider a wireless sensor network (WSN) [43], where data packets are routed to the destination node through multi-hop communications. The WSN is represented by a graph with N nodes (routers), and edges connecting nodes whenever two nodes are within the communication range of each other. The route's QoS performance (quality of service) depends on the decisions of all nodes. Below we formulate the WSN as a networked MA-MDP.

Example 1 (WSN as a networked MA-MDP)**.** *The WSN is a multi-agent system, where sensor nodes are agents. Each agent takes action $a^i \in \mathcal{A}$, which consists of forwarding a packet to one of its neighboring node $j \in \mathcal{N}_i$ and sending the receipt acknowledgement message (ACK) to the predecessor to indicate that it is operating normally, or dropping the data packet and sending the error acknowledgment message (NAK) to the predecessor to indicate an error condition, where $\mathcal{N}_i$ is the set of neighbors of the node i. The global state $s = (s^1, s^2, \ldots, s^N)$ is a tuple of local states s^i, which consists of the set of is neighboring nodes, and the set of packets encapsulated with QoS requirement. A simple example of the reward is $r(s,a) := \sum_{i=1}^N r^i(s^i, a^i)$, where*

$$r^i(s^i, a^i) := \begin{cases} 1 & \text{if ACK received} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

The central reward measures the quality of overall local routing decisions by counting the total number of receipt acknowledgements of messages over the network. Each agent only has access to its own reward, which measures the quality of its own routing decisions based on ACK received from its successor, while the efficiency of overall tasks depends on a sum of local rewards, the total ACKs received by all agents. If each node knows the global state and action (s,a), then the overall system is a networked MA-MDP. Suppose that there exists a central coordinator who knows the full state, joint action, and global reward,

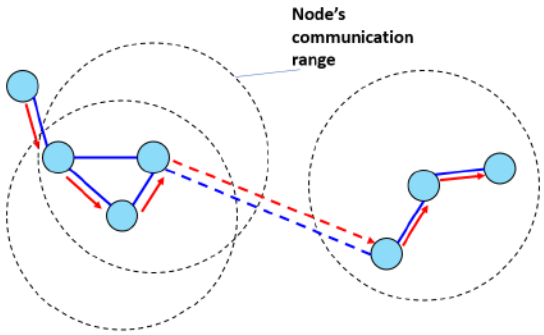

Fig. 2. Routing protocol for wireless sensor networks

(s, a, r). *To operate JALs, the information,* (s, a, r)*, should be broadcasted to all agents during the learning process. In the IL case, the centralized coordinator only needs to broadcast* (s, r) *since each agent* i *knows its own action* a^i*. The underlying setting might be difference, due to the communication constraints and limits imposed by the infrastructure. In general, each agent can take its local action following its local policy, and do not need to optimize over the all set of actions.*

Finding the optimal policy for networked MA-MDPs naturally relates to one of the most fundamental problems in decentralized coordination and control, called the consensus problem. In the sequel, we first review the recent advances in distributed optimization and consensus algorithms, and then march forward to the discussions of recent developments for cooperative MARL based on consensus algorithms.

IV. Distributed Optimization and Consensus Algorithms

In this section, we briefly introduce several fundamental concepts in distributed optimization, which are the backbone of distributed MARL algorithms to be discussed.

A. Consensus

Consider a set of agents, $\mathcal{V} = \{1, 2, \ldots, N\}$, each with some initial values, $x^i(0) \in \mathbb{R}^n$. The agents are interconnected over an underlying communication network characterized by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is a set of undirected edges, and each agent has a local view of the network, i.e., each agent $i \in \mathcal{V}$ is aware of its immediate neighbors, $\mathcal{N}_i$, in the network, and communicates with them only.

The goal of the consensus problem is to design a distributed algorithm that the agents can execute locally to agree on a common value as they refine their estimates. The algorithm must be local in the

sense that each agent performs its own computations and communicates with its immediate neighbors only. Formally speaking, the agents are said to *reach a consensus* if

$$\lim_{k\to\infty} x^i(k) = c, \quad \forall i \in \mathcal{V}, \tag{14}$$

for some $c \in \mathbb{R}^n$ and for every set of initial values $x^i(0) \in \mathbb{R}^n$. For ease of notation, we consider the scalar case, $n = 1$, from now on.

A popular approach to the consensus problem is the *distributed averaging consensus algorithm* [44]

$$x^i(k+1) = \frac{1}{|\mathcal{N}_i|+1} \sum_{j\in\mathcal{N}_i\cup\{i\}} x^j(k), \quad \forall k \geq 0. \tag{15}$$

The averaging update is executed by local agent i, as it only receives values of its neighbors, $x^j(k), j \in \mathcal{N}_i$, and is known to ensure consensus provided that the graph is connected. Note that an undirected graph $\mathcal{G}$ is connected if there is a path connecting every pair of two distinct nodes. Using matrix notations, we can compactly represent (15) as follows

$$x(k+1) = Wx(k), \quad \forall k \geq 0, \tag{16}$$

where $x(k)$ is a column vector with entries, $x^i(k), i = 1, 2, \ldots, N$, and W is the weight matrix associated with (15) such that $[W]_{ij} := \frac{1}{|\mathcal{N}_i|+1}$ if $j \in \mathcal{N}_i \cup \{i\}$ and zero otherwise. Here, $[W]_{ij}$ means the element in the i-th row and j-th column of the matrix W.

The matrix W is a stochastic matrix, i.e., it is nonnegative, and its row sums are one. Hence, W^k converges to a rank one stochastic matrix, i.e., $\lim_{k\to\infty} W^k = \mathbf{1}_n v^T$, where v is the unique (normalized) left-eigenvector of W for eigenvalue 1 with $\|v\|_1 = 1$ and $\mathbf{1}_n$ is an n-dimensional vector with all entries equal to one. Since $x(k) = W^k x(0), \forall k \geq 0$, we have $\lim_{k\to\infty} x(k) = (v^T x(0))\mathbf{1}_n$, implying the consensus.

B. Distributed optimization with averaging consensus

Consider a multi-agent system connected over a network, where each agent i has its own (convex) cost function, $f_i : \mathbb{R}^n \to \mathbb{R}$. Let $F(x) := \sum_{i\in\mathcal{V}} f_i(x)$ be the system objective that the agents want to minimize collectively. The distributed optimization problem is to solve the following optimization problem:

$$\min_{x\in\mathbb{R}^n} F(x) := \sum_{i=1}^{N} f_i(x) \quad \text{subject to} \quad x \in \mathcal{X}, \tag{17}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ represents additional constraints on the variable x. By introducing local copies $x^1, x^2, \ldots, x^N$, it is equivalently expressed as

$$\min_{x^1\in\mathcal{X},\cdots,x^N\in\mathcal{X}} F(x) := \sum_{i=1}^{N} f_i(x^i) \quad \text{subject to} \quad x^1 = x^2 = \cdots = x^N. \tag{18}$$

The distributed averaging consensus algorithm can be generalized to solve the distributed optimization. An example is the *consensus-based distributed subgradient method* [45], where each agent i updates its local variable $x^i(k)$ according to

$$\textbf{Consensus step}: \quad w_{k+1}^i = \frac{1}{|\mathcal{N}_i|+1} \sum_{j \in \mathcal{N}_i \cup \{i\}} x_k^j,$$

$$\textbf{Subgradient descent step}: \quad x_{k+1}^i = \Pi_{\mathcal{X}}[w_{k+1}^i - \alpha_k \partial f_i(w_{k+1}^i)],$$

where ∂f_i is any subgradient of f_i and $\Pi_{\mathcal{X}}$ is the Euclidean projection onto the constraint set $\mathcal{X}$.

The algorithm is a simple combination of the averaging consensus and the classical subgradient method. As in the averaging consensus, the update is executed by local agent i, and it only receives the values of its neighbors, $x_k^j, j \in \mathcal{N}_i$. When all cost functions are convex, it is known that local variables, x_k^i, reach a consensus and converge to a solution to (18), $x^* \in \mathcal{X}$, under properly chosen step-sizes.

Other distributed optimization algorithms include the EXTRA [46] (exact first-order algorithm for decentralized consensus optimization), push-sum algorithm [47] for directed graph models, gossip-based algorithm [48], and etc. A comprehensive and detailed summary of the distributed optimization can be found in the monograph [18].

C. Distributed min-max optimization with averaging consensus

To put it one step further, distributed averaging consensus algorithm can also be generalized to solve the min-max problem in a distributed fashion. The distributed min-max optimization problem deals with the zero-sum game:

$$\min_{x \in \mathcal{X}} \max_{\lambda \in \Lambda} L(x, \lambda) := \sum_{i=1}^N L^i(x, \lambda), \tag{19}$$

where $L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is a convex-concave function and L is separable, i.e., $L = \sum_{i=1}^N L^i$. Introducing local copies $x^1, \ldots, x^N$ and $\lambda^1, \cdots, \lambda^N$, the min-max problem is equivalent to

$$\min_{x^1,\ldots,x^N \in \mathcal{X}} \max_{\lambda^1,\ldots,\lambda^N \in \Lambda} \sum_{i=1}^N L^i(x^i, \lambda^i) \quad \text{s.t.} \quad x^1 = x^2 = \cdots = x^N, \quad \lambda^1 = \lambda^2 = \cdots = \lambda^N. \tag{20}$$

Similar to the distributed subgradient method, the distributed primal-dual algorithm works by performing averaging consensus and sugradient descent for the local variable $x^i(k)$ and $\lambda^i(k)$ of each agent:

$$\textbf{Consensus step}: \quad x_{k+1/2}^i = \frac{1}{|\mathcal{N}_i|+1} \sum_{j \in \mathcal{N}_i \cup \{i\}} x_k^j, \quad \lambda_{k+1/2}^i = \frac{1}{|\mathcal{N}_i|+1} \sum_{j \in \mathcal{N}_i \cup \{i\}} \lambda_k^j,$$

$$\textbf{Primal-dual step}: \quad x_{k+1}^i = \Pi_{\mathcal{X}}[x_{k+1/2}^i - \alpha_k \partial_x L_i(x_{k+1/2}^i, \lambda_{k+1/2}^i)],$$

$$\lambda_{k+1}^i = \Pi_{\Lambda}[\lambda_{k+1/2}^i - \beta_k \partial_\lambda L_i(x_{k+1/2}^i, \lambda_{k+1/2}^i)],$$

where α_k and β_k are step-sizes, $\partial_x L_i$ and $\partial_\lambda L_i$ are any subgradients of $L_i(x,\lambda)$ with respect to x and λ, respectively, and $\Pi_{\mathcal{X}}$ and Π_Λ are the Euclidean projection onto the constraint sets $\mathcal{X}$ and Λ, respectively. The distributed primal-dual algorithm and other variants have been well studied in [49]–[51].

V. Networked MARL with decentralized rewards

In this section, we focus on networked MARL with decentralized rewards, where the networked MA-MDP is described by the tuple, $(\mathcal{S}, \{\mathcal{A}^i\}_{i=1}^N, P, \{r^i\}_{i=1}^N, \gamma, \mathcal{G})$. The goal of each agent is to cooperatively find an optimal policy corresponding to the central (or global) reward, $r = (r^1 + r^2 + \cdots + r^N)/N$, by sharing local learning parameters over a communication network characterized by graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

Decentralized rewards are common in practice when multiple agents cooperate to learn under sensing and physical limitations. They are also particularly useful when MARL agents cooperate to learn an optimal policy securely due to privacy considerations. For instance, if we do not want to reveal full information about the policy design criterion to an RL agent to protect privacy, a plausible approach is to operate multiple RL agents, and provide each agent with only partial information about the reward function. In this case, no single agent alone can learn the optimal policy corresponding to the whole environment, without information exchange among other agents.

Most recent algorithms to be discussed in this section, including [11]–[17], [39], [52], apply the distributed averaging consensus algorithm introduced in Section IV in one way or another. We now discuss these algorithms in details below, with a brief summary provided in Table I.

A. Distributed Policy Evaluation

The goal of distributed policy evaluation is to evaluate the central value function

$$V^\pi(s) = \mathbb{E}\left[\sum_{k=0}^{\infty}\gamma^k \frac{1}{N}\sum_{i=1}^{N} r_\pi^i(s_k)\,\middle|\, s_0 = s\right] \tag{21}$$

in a distributed manner. The information available to each agent is $(s, r^i, \{\theta^j\}_{j\in\mathcal{N}_i})$, where $\{\theta^j\}_{j\in\mathcal{N}_i}$ represents the set of learning parameters agent i receives from its neighbors over the communication network, and $\mathcal{N}_i$ is the set of all neighbors of node i over the graph $\mathcal{G}$. Note that for policy evaluation with state value function V, the information a or a^i is not necessary, thereby it is not indicated in the information set $(s, r^i, \{\theta^j\}_{j\in\mathcal{N}_i})$.

The distributed TD-learning [11] executes the following local updates of agent i:

$$\theta^i \leftarrow \underbrace{\frac{1}{|\mathcal{N}_i|+1}\sum_{j\in\mathcal{N}_i\cup\{i\}}\theta^j}_{\text{Mixing term}} + \alpha_k \underbrace{(r^i(s,\pi(s)) + \gamma\phi(s')^T\theta^i - \phi(s)^T\theta^i)\phi(s)}_{\text{TD update}},$$

TABLE I

COOPERATIVE MARL WITH DECENTRALIZED REWARDS AND COMMUNICATION NETWORKS (LFA: LINEAR FUNCTION APPROXIMATION; NFA: NONLINEAR FUNCTION APPROXIMATION; N/A: NOT APPLICABLE

	Papers	Availability of actions	Reward	Function Approx.	Convergence
Policy Evaluation	Doan et al. [11]	N/A	Decentralized	LFA	Yes
	Wai et al. [16]			LFA	Yes
	Lee [17]			LFA	Yes
	Macua et al. [39]	N/A	Centralized	LFA	Yes
	Stanković et al. [52]			LFA	Yes
Policy Optimization	Kar et al. [12]	JAL	Decentralized	Tabular	Yes
	Zhang et al. [13]	JAL		LFA, NFA	Yes
	Zhang et al. [14]	JAL		LFA, NFA	Local
	Qu et al. [15]	JAL		NFA	Local

where each agent i keeps its local parameter θ^i and α_k is the step-size. The algorithm resembles the consensus-based distributed subgradient method in Section IV-B. The first term, dubbed as the mixing term, is an average of local copies of the learning parameter of neighbors, $\mathcal{N}_i$, received from communication over networks, and controls local parameters to reach a consensus. The second term, referred to as the TD update, follows the standard TD updates. Under suitable conditions such as the graph connectivity, each local copy, θ^i, converges to θ^* in expectation and almost surely [11], where θ^* is the optimal solution found by the single-agent TD learning acting on the central reward.

Example 2. *Consider the WSN in Example 1. Suppose each agent has its own fixed policy (routing protocol), π_i, according to which the agent forwards a packet encoded in the state $\mathbf{s}$ to one of its neighboring node $j \in \mathcal{N}_i$, sends ACKs to the predecessor when it receives a packet without error conditions or drops the data packet and sends NAKs to the predecessor otherwise. Each agent receives a local reward according to (13) based on ACK received from its successors, knows the global state $\mathbf{s}$, and keeps its local parameter θ^i. During the learning, each agent i receives the local parameters θ^j_k, $j \in \mathcal{N}_i$ from its neighbors, and its parameter θ^i_k converges to θ^* which best approximate the optimal value function. By doing so, each agent can learn the optimal solution, θ^*, for the value function estimation without knowing the global reward.*

B. Distributed Policy Optimization

The goal of distributed policy optimization is to cooperatively find an optimal central policy corresponding to the central reward, r. Note that the distributed TD-learning in the previous section only finds the state value function under a given policy. The averaging consensus idea can also be extended to Q-learning and actor-critic algorithms for finding the optimal policy for networked MARL.

The distributed Q-learning in [12] locally updates the Q-function according to

$$Q^i(s,a) \leftarrow Q^i(s,a) - \eta(s,a) \underbrace{\sum_{j\in\mathcal{N}_i\cup\{i\}} (Q^i(s,a) - Q^j(s,a))}_{\text{Mixing term}}$$
$$+ \underbrace{\alpha(s,a)\,(r^i(s,a) + \gamma \max_{a'\in\mathcal{A}} Q^i(s',a') - Q^i(s,a))}_{\text{Q-learning update}},$$

where i is the agent index, $\eta(s,a)$ and $\alpha(s,a)$ are learning rates (or step-sizes) depending on the number of instances when (s,a) is encountered. The information available to each agent is $(s,a,r^i,\{Q^j\}_{j\in\mathcal{N}_i\cup\{i\}})$. The overall diagram of the distributed Q-learning algorithm is given in Figure 3. Each agent i keeps the local Q-function, Q^i, and the mixing term consists of Q-functions of neighbors received from communication networks. It has been shown that each local Q^i reaches a consensus and converges to Q^* almost surely [12] with suitable step-size rules and under assumptions such as the connectivity of the graph and an infinite number of state-action visits.

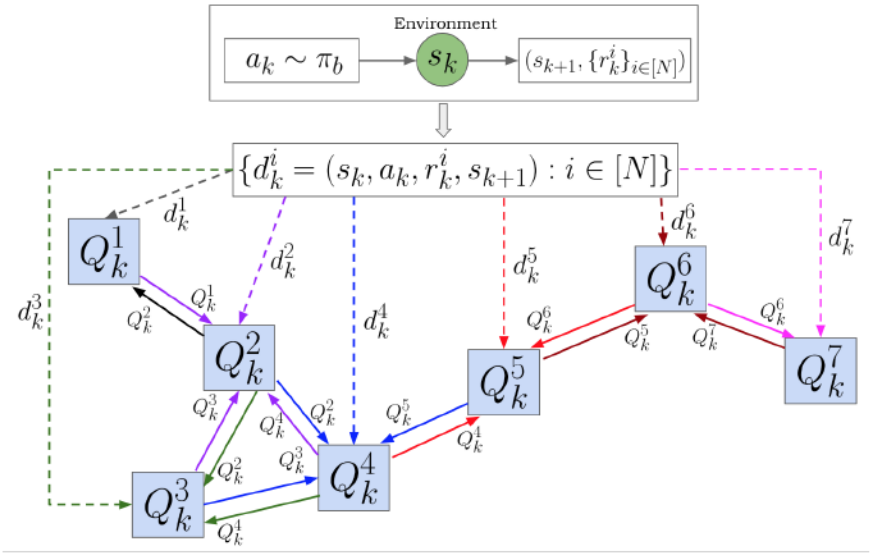

Fig. 3. Diagram of distributed Q-learning algorithm in [12]. Here the joint-action a_k is chosen by a behavior policy π_b.

The distributed actor-critic algorithm in [13] generalizes the single-agent actor-critic to networked MA-MDP settings where the averaging consensus steps are taken for the value function parameters

$$\text{Critic update}: \theta^i_{k+1/2} = \theta^i_k + \alpha_k(r^i(s_k,a_k) + \gamma Q(s_{k+1},a_{k+1};\theta^i_k) - Q(s_k,a_k;\theta^i_k))\nabla_\theta Q(s_k,a_k;\theta^i_k)$$

$$\text{Actor update}: w^i_{k+1} = w^i_k + \beta_k A(s_k,a_k;\theta^i_k)\nabla_{w^i}\log\pi^i(a^i_k|s_k;w^i_k)$$

$$\text{Consensus step}: \theta^i_{k+1} = \frac{1}{|\mathcal{N}_i|+1}\sum_{j\in\mathcal{N}_i\cup\{i\}}\theta^j_{k+1/2}$$

where w^i and θ^i are parameters of nonlinear function approximations for the local actor and local critic, respectively, and $a^i_k \sim \pi^i(\cdot|s_k;w^i_k)$, $a^i_{k+1} \sim \pi^i(\cdot|s_{k+1};w^i_k)$. Here, $A(s_k,a_k;\theta^i_k) := Q(s_k,a_k;\theta^i_k) - \sum_{a^i\in\mathcal{A}^i}\pi^i(a^i|s_k;w^i_k)Q(s_k,(a^1_k,\ldots,a^i,\ldots,a^N_k);\theta^i_k)$ is the advantage function evaluated at (s_k,a_k). The overall diagram of the distributed actor-critic is given in Figure 4. Each agent i keeps its local parameters $\{\theta^i, w^i\}$, and in the mixing step, it only receives local parameters of the critic from neighbors. The actor and critic updates are similar to those of typical actor-critic algorithms with local parameters. The information available to each agent is $(s,a,r^i,w^i,\{\theta^j\}_{j\in\mathcal{N}_i\cup\{i\}})$.

The results in [14] study a MARL generalization of the fitted Q-learning with the information structure $(s,a,r^i,\{\theta^j\}_{j\in\mathcal{N}_i\cup\{i\}})$. Compared to the tabular distributed Q-learning in [12], the distributed actor-critic and fitted Q-learning may not converge to an exact optimal solution mainly due to the use of function approximations.

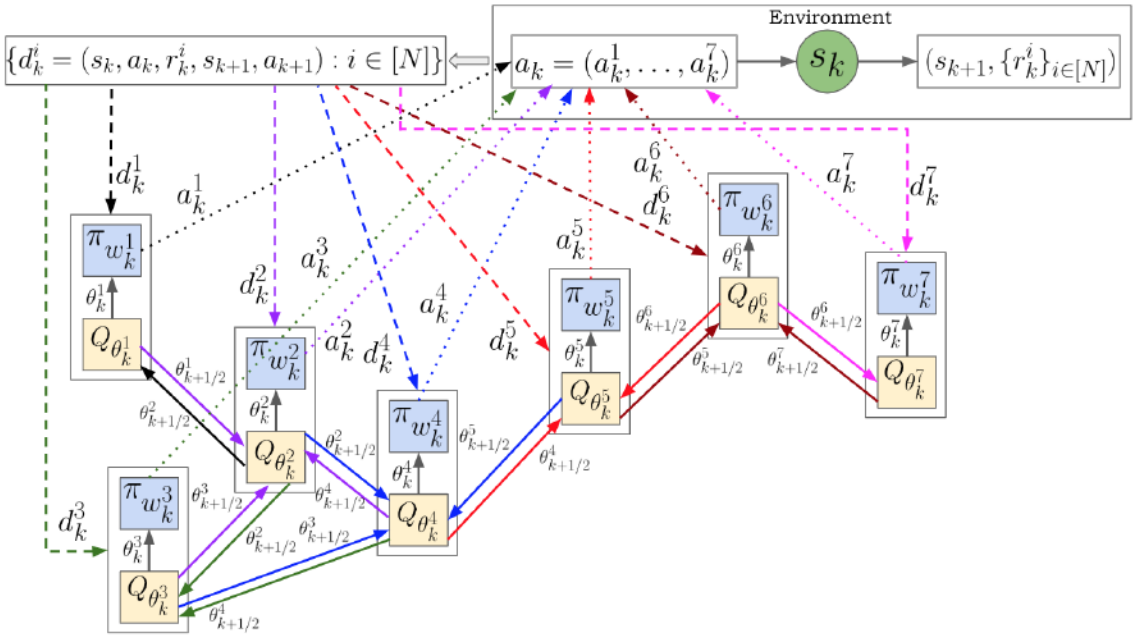

Fig. 4. Diagram of distributed actor-critic algorithm in [13]. Here the joint-action a_k is taken in on-policy manner.

C. Optimization Frameworks for Networked MA-MDP

Recall that in Section II-B, we discussed optimization frameworks of single-agent RL problem. By integrating them with consensus-based distributed optimization, they can be naturally adapted to solve networked MA-MDPs. In this subsection, we introduce some recent work in this direction, such as the value propagation [15], primal-dual distributed incremental aggregated gradient [16], distributed GTD [17]. The main idea of these algorithms is essentially rooted in formulating the overall MDP into a min-max optimization problem, $\min_{x\in\mathcal{X}}\max_{\lambda\in\Lambda} L(x,\lambda)$, with separable function $L(x,\lambda)=\sum_{i=1}^{N} L^i(x,\lambda)$, and solving the distributed min-max optimization problem (20). For MARL tasks, the distributed min-max problem can be solved using stochastic variants of the distributed saddle-point algorithms in Section IV-C.

The multi-agent policy evaluation algorithms in [16] and [17] are multi-agent variants of the GTD [26] based on the consensus-based distributed saddle-point framework for solving the mean-squared projected Bellman error in (11), which can be equivalently converted into an optimization problem with separable objectives:

$$\min_{\theta}\frac{1}{2}\sum_{i=1}^{N}\|\Pi(R_\pi^i+\alpha P^\pi\Phi\theta)-\Phi\theta\|_D^2. \tag{22}$$

To alleviate the double sampling issues in GTD, the approach in [16] applies the Fenchel duality with an additional proximal term to each objective, arriving at the reformulation:

$$\min_{\{\theta^i\}_{i=1}^N}\sum_{i=1}^{N} d^i(\theta^i)\quad \text{s.t.}\quad \theta^1=\theta^2=\cdots=\theta^N, \tag{23}$$

where the local objectives are expressed as max-forms

$$d^i(\theta):=\max_{w^i}\{J_i(\theta,w_i):=w_i^T(\Phi^T D((1/N)R_\pi^i+\alpha P^\pi\Phi\theta)-\Phi\theta)-(1/2)w_i^T\Phi^T D\Phi w_i+(\rho/2)\|\theta^i\|_2^2\}.$$

The resulting problem can be solved by using stochastic variants of the consensus-based distributed subgradient method akin to [53]. In particular, the algorithm introduces gradient surrogates of the objective function with respect to the local primal and dual variables, and the mixing steps for consensus are applied to both the local parameters and local gradient surrogates. The main idea of the primal-dual algorithm used in [53] is briefly (with some simplifications) written by

$$\text{Primal update}: \theta_{k+1}^i=\underbrace{\frac{1}{|\mathcal{N}_i|+1}\sum_{j\in\mathcal{N}_i\cup\{i\}}\theta_k^j}_{\text{mixing term}}-\alpha\hat{g}_k^i$$

$$\text{Dual update}: w_{k+1}^i=w_k^i+\beta\hat{h}_k^i$$

where α and β are step-sizes, $\hat{g}_k^i$ and $\hat{h}_k^i$ are surrogates of the gradients, $\nabla_{\theta^i}J_i(\theta_k^i,w_k^i)$ and $\nabla_{w^i}J_i(\theta_k^i,w_k^i)$, respectively, from through some basic gradient tracking steps.

The multi-agent policy evaluation in [17] approaches in a different way to solve (22). Assuming each parameter θ^i is scalar for simplicity, the distributed optimization (22) can be converted into

$$\min_{\{\theta^i\}_{i=1}^N} \frac{1}{2}\sum_{i=1}^N \|\Pi(R_\pi^i + \alpha P^\pi \Phi\theta^i) - \Phi\theta^i\|_D^2 + \bar{\theta}^T L^T L\bar{\theta} \quad \text{s.t.} \quad L\bar{\theta} = 0, \tag{24}$$

where $\bar{\theta}$ is the vector enumerating the local parameters, $\{\theta^i\}_{i=1}^N$, and $L = L^T \in \mathbb{R}^N$ is the graph Laplacian matrix. Note that if the underlying graph is connected, then $L\bar{\theta} = 0$ if and only if $\theta^1 = \theta^2 = \cdots = \theta^N$. By constructing the Lagrangian dual of the above constrained optimization, we obtain the corresponding single min-max problem. Thanks to the Laplacian matrix, the corresponding stochastic primal-dual algorithm is automatically decentralized. Compared to [53], it only needs to share local parameters with neighbors rather than the gradient surrogates.

The MARL in [15] combines the averaging consensus and SBEED [34] (Smoothed Bellman Error Embedding), which is called distributed SBEED here. In particular, the distributed SBEED aims to solve the so-called smoothed Bellman equation

$$V_\theta(s) = \frac{1}{N}\sum_{i=1}^N R_a^i(s) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)}[V_\theta(s')] - \lambda \sum_{i=1}^N \ln(\pi_{w^i}^i(s,a^i)),$$

by minimizing the corresponding mean squared smoothed Bellman error:

$$\min_{\theta,\{w^i\}_{i=1}^N} \mathbb{E}_{s,a}\left[\left(\frac{1}{N}\sum_{i=1}^N R_a^i(s) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)}[V_\theta(s')] - \lambda\sum_{i=1}^N \ln(\pi_{w^i}^i(s,a^i)) - V_\theta(s)\right)^2\right], \tag{25}$$

where λ is a positive real number capturing the smoothness level, θ and w are deep neural network parameters for the value and policy, respectively. Directly applying the stochastic gradient to the above objective using samples leads to biases due to the nonlinearity of the objective (or double sampling issue). To alleviate this difficulty, the distributed SBEED introduces the primal-dual form as in [34], which results in a distributed saddle-point problem similar to (20) and is processed with a stochastic variants of the distributed proximal primal-dual algorithm in [50].

D. Special Case: Networked MARL with Centralized Rewards

Lastly, we remark that the algorithms in this section can be directly applied to MA-MDPs with central rewards. As in Section III, we consider an MDP, $(\mathcal{S}, \{\mathcal{A}^i\}_{i=1}^N, P, r, \gamma)$, with an additional network communication model $\mathcal{G}$, while each agent i receives the common reward $r(s,a)$ instead of the local reward $r^i(s,a)$. One may imagine reinforcement learning algorithms running in N identical and independent

simulated environments. Under this assumption, a distributed policy evaluation was studied in [52]. It combines GTD [26] with the distributed averaging consensus algorithm as follows:

$$\text{GTD update}: \begin{cases} \theta^i_{k+1/2} = \theta^i_k + \alpha_k(\phi(s) - \gamma\phi(s'))(\phi(s)^T w^i_k) \\ w^i_{k+1/2} = w^i_k + \alpha_k(\delta^i_k - \phi(s)^T w^i_k)\phi(s) \end{cases}$$

$$\text{Consensus step}: \begin{cases} \theta^i_{k+1} = \frac{1}{|\mathcal{N}_i|+1} \sum_{j\in\mathcal{N}_i\cup\{i\}} \theta^j_{k+1/2} \\ w^i_{k+1} = \frac{1}{|\mathcal{N}_i|+1} \sum_{j\in\mathcal{N}_i\cup\{i\}} w^j_{k+1/2} \end{cases}$$

where $\delta^i_k = r(s,\pi(s)) + \gamma\phi(s')^T\theta^i_k - \phi(s)^T\theta^i_k$ is the local TD-error. Each agent has access to the information $(s, a, r, \{\theta^j\}_{j\in\mathcal{N}_i})$, while the action a is not used in the updates. The first update is equivalent to the GTD in [26] with a local parameter (θ^i, w^i) and the second term is equivalent to the distributed averaging consensus update in (15). Since the GTD update rule is equivalent to a stochastic primal-dual algorithm, the above update rule is equivalent to a distributed algorithm for solving the distributed saddle-point problem in (20). In the same vein, the multi-agent policy evaluation [39] generalizes the GQ learning to distributed settings, which is more general than GTD in the sense that it incorporates an importance weight of agent i that measures the dissimilarity between the target and behavior policy for the off-policy learning.

VI. Future Directions

Until now, we mainly focused on networked MARL and recent advances which combine tools in consensus-based distributed optimization with MARL under decentralized rewards. There remain much more challenging agendas to be studied. By bridging two domains in a synergistic way, these research topics are expected to generate new results and enrich both fields.

a) Robustness of networked MARL: Communication networks in real world, oftentimes, suffer from communication delays, noises, link failures, or packet drops. Moreover, network topologies may vary as time goes by, and the information exchange over the networks may not be bidirectional in general. Extensive results on distributed optimization algorithms over time-varying, directed graphs, w/o communication delays have been actively studied in the distributed optimization community, yet mostly in deterministic and convex settings. The study of networked MARLs under aforementioned communication limitations is an open and challenging topic.

b) Resilience of networked MARL: Building resilient networked MARL under adversarial attacks is another important topic. A resilient consensus-based distributed optimization algorithm under adversarial attacks has been studied in [54], which considers scenarios where adversarial agents exist among networked agents and send arbitrary parameters to their neighboring agents to disrupt the solution search.

In such cases, analysis of fundamental limitations on distributed optimization algorithms and protocols resilient against such adversarial behaviors are available. For networked MARL, such issues remain largely unexplored.

c) Development of deep networked MARL algorithms: Another interesting direction is the application of consensus-based distributed optimizations to recent deep RL algorithms, such as deep Q-learning [55], trust region policy optimization (TRPO) [56], proximal policy optimization (PPO) [57], deep deterministic policy gradient (DDPG) [58], twin delayed DDPG (TD3) [59]. Most of these algorithms are variants of policy search algorithm and involve optimization procedures in certain stages. Ideas of distributed optimizations can potentially be applied to these deep RL algorithms as well.

d) Theoretical understanding of networked MARL with deep neural nets: Fundamental analysis of networked MARL with nonlinear function approximation is still an open question. For the optimization-based MARLs, when the value function or policy are parameterized by deep neural networks, the resulting distributed min-max problems discussed eventually become nonconvex-nonconcave. Solving this class of distributed optimization problems in a principled manner remains an intriguing research topic.

e) MARL for parallel computing: Lastly, networked MARLs can be used to reduce memory and computational cost, and accelerate the training by exploiting parallel computation. Most RL algorithms require enormous experiences to find a reasonably good policy, which may not be easily collected by a single agent. Instead, a large number of cooperative RL agents over networks can more effectively collect experiences using their own sensors such as crowd sources. Moreover, these agents can learn different parts of learning parameters and features with lower dimensions compared to the state-space, which could greatly reduce the memory and computational cost. There exist several works in this direction, such as the distributed gossiping TD learning in [60], the distributed policy search algorithm [42], etc. In this case, the design of network topology and infrastructures becomes quite critical in improving the learning efficiency and balancing the tradeoff between communication and computation cost.

VII. Biographies

a) Donghwan Lee: received the B.S. degree in Electronic Engineering from Konkuk University, Seoul, South Korea, in 2008, the M.S. degree in Electrical and Electronic Engineering, Yonsei University, Seoul, South Korea, in 2010, the M.S. degree in Mathematics and the Ph.D. degree in Electrical and Computer Engineering from Purdue University, West Lafayette, IN, USA, in 2017. He was a postdoctoral research associate with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, IL, USA, from 2017 to 2019. He is currently an assistant professor with the School of Electrical Engineering, KAIST, Daejeon, South Korea. His current research interests include stochastic programming,

multi-agent systems, and reinforcement learning. He was an associate editor of IEEE Transactions on Fuzzy Systems in 2015.

b) Niao He: received B.S. in Mathematics from University of Science and Technology of China in 2010 and Ph.D. in Operations Research from Georgia Institute of Technology in 2015. Currently, she is an assistant professor in the Department of Industrial and Enterprise Systems Engineering and Coordinated Science Laboratory at the University of Illinois at Urbana-Champaign. Her research interests are in optimization and machine learning. She is also a recipient of the AISTATS Best Paper Award in 2016, NSF CISE CRII Award in 2018 and the NCSA Faculty Fellowship in 2018.

c) Parameswaran Kamalaruban: received B.S. in Electronics and Telecommunication Engineering at the University of Moratuwa and Ph.D. in Computer Science at the Australian National University and is currently a postdoctoral researcher at EPFL. His research interests include statistical learning theory, information theory, and reinforcement learning.

d) Volkan Cevher: received the B.Sc. in electrical engineering from Bilkent University in 1999 and the Ph.D. in electrical and computer engineering from the Georgia Institute of Technology in 2005. He was a Research Scientist with the University of Maryland from 2006-2007 and also with Rice University from 2008-2009. Currently, he is an Associate Professor at the Swiss Federal Institute of Technology Lausanne and a Faculty Fellow in the Electrical and Computer Engineering Department at Rice University. His research interests include signal processing theory, machine learning, convex optimization, and information theory. Dr. Cevher was the recipient of the Google Faculty Research Award in 2018, the IEEE Signal Processing Society Best Paper Award in 2016, a Best Paper Award at CAMSAP in 2015, a Best Paper Award at SPARS in 2009, and an ERC CG in 2016 as well as an ERC StG in 2011.

REFERENCES

[1] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of markov decision processes," *Mathematics of Operations Research*, vol. 27, no. 4, pp. 819–840, 2002.

[2] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel, "Multi-agent reinforcement learning in sequential social dilemmas," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, 2017, pp. 464–473.

[3] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017, pp. 1146–1155.

[4] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate to solve riddles with deep distributed recurrent q-networks," *arXiv preprint arXiv:1602.02672*, 2016.

[5] G. J. Laurent, L. Matignon, and L. Fort-Piat, "The world of independent learners is not markovian," *International Journal of Knowledge-based and Intelligent Engineering Systems*, vol. 15, no. 1, pp. 55–64, 2011.

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

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

[8] M. Lauer and M. Riedmiller, "An algorithm for distributed reinforcement learning in cooperative multi-agent systems," in *In Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[9] ——, "Reinforcement learning for stochastic cooperative multi-agent systems," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*, 2004, pp. 1516–1517.

[10] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998.

[11] T. T. Doan, S. T. Maguluri, and J. Romberg, "Convergence rates of distributed TD(0) with linear function approximation for multi-agent reinforcement learning," *arXiv preprint arXiv:1902.07393*, 2019.

[12] S. Kar, J. M. Moura, and H. V. Poor, "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.

[13] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, "Fully decentralized multi-agent reinforcement learning with networked agents," *arXiv preprint arXiv:1802.08757*, 2018.

[14] ——, "Finite-sample analyses for fully decentralized multi-agent reinforcement learning," *arXiv preprint arXiv:1812.02783*, 2018.

[15] C. Qu, S. Mannor, H. Xu, Y. Qi, L. Song, and J. Xiong, "Value propagation for decentralized networked deep multi-agent reinforcement learning," *arXiv preprint arXiv:1901.09326*, 2019.

[16] H.-T. Wai, Z. Yang, P. Z. Wang, and M. Hong, "Multi-agent reinforcement learning via double averaging primal-dual optimization," in *Advances in Neural Information Processing Systems*, 2018, pp. 9672–9683.

[17] D. Lee, "Stochastic primal-dual algorithm for distributed gradient temporal difference learning," *arXiv preprint arXiv:1805.07918*, 2018.

[18] A. Nedich, "Convergence rate of distributed averaging dynamics and optimization in networks," *Foundations and Trends® in Systems and Control*, vol. 2, no. 1, pp. 1–100, 2015.

[19] L. Bu, R. Babu, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 38, no. 2, pp. 156–172, 2008.

[20] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multi-agent systems: A review of challenges, solutions and applications," *arXiv preprint arXiv:1812.11794*, 2018.

[21] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, 1998.

[22] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Transactions on Automatic Control*, vol. 42, no. 5, pp. 674–690, 1997.

[23] J. Bhandari, D. Russo, and R. Singal, "A finite time analysis of temporal difference learning with linear function approximation," *arXiv preprint arXiv:1806.02450*, 2018.

[24] G. Dalal, B. Szörényi, G. Thoppe, and S. Mannor, "Finite sample analyses for TD(0) with function approximation," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[25] R. Srikant and L. Ying., "Finite-time error bounds for linear stochastic approximation and TD learning," *Proceedings of the Thirty-Second Conference on Learning Theory, PMLR 99:2803-2830*, 2019.

[26] R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora, "Fast gradient-descent

methods for temporal-difference learning with linear function approximation," in *International Conference on Machine Learning (ICML)*, 2009, pp. 993–1000.

[27] R. S. Sutton, H. R. Maei, and C. Szepesvári, "A convergent $O(n)$ temporal-difference algorithm for off-policy learning with linear function approximation," in *Advances in neural information processing systems*, 2009, pp. 1609–1616.

[28] B. Dai, N. He, Y. Pan, B. Boots, and L. Song, "Learning from conditional distributions via dual embeddings," in *Artificial Intelligence and Statistics*, 2017, pp. 1458–1467.

[29] D. Lee and N. He, "Target-based gradient TD-learning," in *International Conference on Machine Learning*, 2019.

[30] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic programming*. Athena Scientific Belmont, MA, 1996.

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

[32] V. R. Konda and J. N. Tsitsiklis, "On actor-critic algorithms," *SIAM journal on Control and Optimization*, vol. 42, no. 4, pp. 1143–1166, 2003.

[33] Y. Chen and M. Wang, "Stochastic primal-dual methods and sample complexity of reinforcement learning," *arXiv preprint arXiv:1612.02516*, 2016.

[34] B. Dai, A. Shaw, L. Li, L. Xiao, N. He, Z. Liu, J. Chen, and L. Song, "SBEED: convergent reinforcement learning with nonlinear function approximation," in *International Conference on Machine Learning*, 2018, pp. 1133–1142.

[35] D. Lee and N. He, "Stochastic primal-dual Q-learning," *arXiv preprint arXiv:1810.08298*, 2018.

[36] M. L. Puterman, *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[37] B. Dai, A. Shaw, N. He, L. Li, and L. Song, "Boosting the actor with dual critic," in *International Conference on Learning Representations*, 2018.

[38] L. Baird, "Residual algorithms: reinforcement learning with function approximation," in *Proceedings of the 12th International Conference on Machine Learning*, 1995, pp. 30–37.

[39] S. V. 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.

[40] S. Mahadevan, B. Liu, P. Thomas, W. Dabney, S. Giguere, N. Jacek, I. Gemp, and J. Liu, "Proximal reinforcement learning: A new theory of sequential decision making in primal-dual spaces," *arXiv preprint arXiv:1405.6757*, 2014.

[41] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," *AAAI/IAAI*, vol. 1998, pp. 746–752, 1998.

[42] L. Peshkin, K.-E. Kim, N. Meuleau, and L. P. Kaelbling, "Learning to cooperate via policy search," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, 2000, pp. 489–496.

[43] X. Liang, I. Balasingham, and S.-S. Byun, "A multi-agent reinforcement learning based routing protocol for wireless sensor networks," in *2008 IEEE International Symposium on Wireless Communication Systems*, 2008, pp. 552–557.

[44] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[45] A. Nedic, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[46] W. Shi, Q. Ling, G. Wu, and W. Yin, "EXTRA: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

[47] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2014.

[48] A. Nedic, "Asynchronous broadcast-based convex optimization over a network," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1337–1351, 2010.

[49] M. Zhu and S. Martínez, "On distributed convex optimization under inequality and equality constraints," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 151–164, 2011.

[50] M. Hong, D. Hajinezhad, and M.-M. Zhao, "Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks," in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 1529–1538.

[51] D. Yuan, S. Xu, and H. Zhao, "Distributed primal–dual subgradient method for multiagent optimization via consensus algorithms," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 6, pp. 1715–1724, 2011.

[52] M. S. Stanković and S. S. Stanković, "Multi-agent temporal-difference learning with linear function approximation: weak convergence under time-varying network topologies," in *American Control Conference (ACC)*, 2016, pp. 167–172.

[53] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 3, pp. 1245–1260, 2017.

[54] S. Sundaram and B. Gharesifard, "Distributed optimization under adversarial nodes," *IEEE Transactions on Automatic Control*, vol. 64, no. 3, pp. 1063–1076, 2018.

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

[56] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, 2015, pp. 1889–1897.

[57] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[58] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[59] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *arXiv preprint arXiv:1802.09477*, 2018.

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