

# Distributed Policy Evaluation Under Multiple Behavior Strategies

Sergio Valcarcel Macua, *Student Member, IEEE*, Jianshu Chen, *Member, IEEE*  
Santiago Zazo, *Member, IEEE*, and Ali H. Sayed, *Fellow, IEEE*

**Abstract**—We apply diffusion strategies to develop a fully-distributed cooperative reinforcement learning algorithm in which agents in a network communicate only with their immediate neighbors to improve predictions about their environment. The algorithm can also be applied to off-policy learning, meaning that the agents can predict the response to a behavior different from the actual policies they are following. The proposed distributed strategy is efficient, with linear complexity in both computation time and memory footprint. We provide a mean-square-error performance analysis and establish convergence under constant step-size updates, which endow the network with continuous learning capabilities. The results show a clear gain from cooperation: when the individual agents can estimate the solution, cooperation increases stability and reduces bias and variance of the prediction error; but, more importantly, the network is able to approach the optimal solution even when none of the individual agents can (e.g., when the individual behavior policies restrict each agent to sample a small portion of the state space).

**Index Terms**—Adaptive networks, Arrow-Hurwicz algorithm, diffusion strategies, distributed processing, gradient temporal difference, mean-square-error, reinforcement learning, saddle-point problem

## I. INTRODUCTION

CONSIDER the problem in which a network of autonomous agents collaborate to predict the response of the environment to their actions. The network forms a connected graph, where there is at least one path between every pair of nodes. The agents learn locally from their individual interactions with the environment and share knowledge with their neighbors. Only direct neighborhood communication is allowed. We assume the environment can be modeled as a Markov decision process. The agents do not have access to the actual state of the environment, but just to feature vectors representing it. The feature representation is convenient in problems with very large state dimensions since it is computationally more efficient to work with features of smaller dimension than the size of the original state-space.

This work was supported in part by the Spanish Ministry of Science and Innovation in the program CONSOLIDER-INGENIO 2010 under the Grant CSD2008-00010 COMONSENS and by the NSF grants CCF-1011918 and ECCS-1407712. A short preliminary version dealing with a special case of this work appears in the conference publication [1].

S. V. Macua and S. Zazo are with the Department of Signals, Systems and Radiocommunications, Escuela Técnica Superior de Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, Madrid 28040, Spain (e-mail: sergio@gaps.ssr.upm.es; santiago@gaps.ssr.upm.es).

J. Chen and A. H. Sayed are with the Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA (e-mail: cjs09@ucla.edu; sayed@ee.ucla.edu).

In the scenario under study in this work, every agent takes actions according to an individual policy, which is possibly different from that of every other agent. The objective of the agents is to assess the response of the environment to a common hypothetical behavior, the *target policy*, which is the same for every agent but different from the actual *behavior policies* they are following. This problem of predicting the response to a target policy different from the behavior policy is commonly referred as *off-policy* learning [2]. Off-policy learning has been claimed to be necessary when the agents need to perform tasks in complex environments because they could perform many different predictions in parallel from a single stream of data [3]–[5].

The predictions by the agents are made in the form of value functions [2], [6], [7]. The gradient-temporal-difference (GTD) algorithm is one useful method for computing approximate value functions. It was originally proposed for the single agent scenario in [8], [9], and derived by means of the stochastic optimization of a suitable cost function. The main advantages of this single-agent GTD are its low complexity and its convergence guarantees (for diminishing step-sizes) under the off-policy setting. In Section III of this work we apply diffusion strategies to develop a distributed GTD algorithm that extends the single-agent GTD to multi-agent networks. There are several distributed strategies that can be used for this purpose, such as consensus [10]–[14] and diffusion strategies [15]–[18]. Consensus strategies have been successfully applied to the solution of static optimization problems, where the objective does not drift with time. They have been studied largely under diminishing step-size conditions to ensure agreement among cooperating agents. Diffusion strategies, on the other hand, have been proved to be particularly apt at endowing networks with continuous adaptation and learning abilities to enable tracking of drifting conditions. There are several forms of diffusion; recent overviews appear in [19]–[21]. It has been shown in [22] that the dynamics of diffusion networks leads to enhanced stability and lower mean-square-error (MSE) than consensus networks. In particular, the analysis in [20]–[22] shows that consensus networks combine local data and in-neighborhood information asymmetrically, which can make the state of consensus networks grow unbounded even when all individual agents are mean stable in isolation. This behavior does not happen in diffusion networks, in which local and external information are symmetrically combined by construction, enhancing the stability of the network. For these reasons, we focus in the remainder of this article on the derivation of a diffusion strategy for GTD over multi-agent networks. As a

byproduct of this derivation, we show that the GTD algorithm, motivated as a two time-scales stochastic approximation in [9], is indeed a stochastic Arrow-Hurwicz algorithm applied to the dual problem of the original formulation.

The convergence analysis of reinforcement learning algorithms is usually challenging even for the single-agent case, and studies are often restricted to the case of diminishing step-sizes [8], [9], [23]. For a distributed algorithm, the analysis becomes more demanding because the estimation process at each node is influenced by the estimates at the other nodes, so the error propagates across the network. Another difficulty in the distributed case is that the agents may follow different behavior policies and, thus, their individual cost functions could have different minimizers. In Section IV, we will analyze the steady-state and transient behavior of the proposed distributed algorithm, deriving closed-form expressions that characterize the network performance for sufficiently small *constant* step-sizes. We employ constant, as opposed to decaying step-sizes, because we are interested in distributed solutions that are able to continually adapt and learn. The performance analysis will reveal that when the agents follow the same behavior policy, they will be able to find an unbiased estimator for the centralized solution. On the other hand, when the agents behave differently, they will approach, up to some bias, the solution of a convex combination of their individual problems. This bias is proportional to the step-size, so it becomes negligible when the step-size is sufficiently small. One important benefit that results when the agents behave differently is that, although the agents do not directly share their samples, the in-network experience becomes richer in a manner that the diffusion strategy is able to exploit. In particular, in the reinforcement learning literature, it is customary to assume that the behavior policy must allow the agents to visit every possible state infinitely often. We will relax this assumption and show that the distributed algorithm is able to perform well even when the individual agents only visit small portions of the state-space, as long as there are other agents that explore the remaining regions. Therefore, even though none of the agents can find the optimal estimate of the value function by itself, they can achieve it through cooperation. This is an interesting capability that emerges from the networked solution.

In this work, we consider a setting in which the agents can communicate with their neighbors, but they operate without influencing each other. This setup is meaningful in many real applications. Consider, for example, a water purification plant controlled and monitored by a wireless actuator-sensor network, in which each device is attached to a different water-tank. The quality of the water (e.g., the amount of bacteria) in one tank will be influenced by the decisions (e.g., delivering some amount of chlorine) made by the device controlling that tank, independently of what other devices do. Still, since all water tanks behave similarly under similar circumstances, the devices in the network can benefit from sharing their individual knowledge.

#### A. Related works

There are several insightful works in the literature that address issues pertaining to distributed learning albeit un-

der different scenarios and conditions than what is studied in this article. For example, the work in [24] proposes a useful algorithm, named QD-learning, which is a distributed implementation of Q-learning using consensus-based stochastic approximation. The diffusion strategy proposed herein is different in several respects. QD-learning asymptotically solves the optimal control problem, learning the policy that maximizes the long-term reward of the agents. Here, we focus on predicting the long-term reward for a given policy, which is an important part of the control problem. However, QD-learning is developed in [24] under the assumption of perfect-knowledge of the state. Here, we study the case in which the agents only know a feature representation of the state, which is used to build a parametric approximation of the value function, allowing us to tackle large problems, for which Q-learning schemes can diverge [25], [26]. Finally, we enforce constant step-sizes in order to enable continuous adaptation and learning. In comparison, the analysis in [24] employs a diminishing step-size that dies out as time progresses and, therefore, turns off adaptation and is not able to track concept drifts in the data.

Another related work [27] analyzes the performance of cooperative distributed asynchronous estimation of linearly approximated value functions using standard temporal difference (TD), but it is well known that TD learning with parametric approximation schemes can diverge when the agents learn off-policy [25], [26]. In addition, although the algorithm in [27] is distributed, in the sense that there is no fusion center, it requires full connectivity (i.e., every node must be able to exchange information with every other node in the network), which is a restrictive assumption that prevents the algorithm from large-scale deployments. In this article, we focus on fully distributed solutions that only require the network of agents to be connected (but not necessarily fully connected). Other related—but more heuristic—approaches include [28], [29].

#### B. Notation

Lower case letters are used to denote both scalar values and vectors. Matrices are denoted by upper case letters. Boldface notation denotes random variables (e.g.,  $s$  is a realization for  $s$ ). The state of the environment and the action taken by an agent are denoted by  $s$  and  $a$ , respectively. With a slight abuse of notation,  $s(i)$  and  $a(i)$  denote the state and action variables at time  $i$ . Moreover, whenever a variable is specific to some agent  $k$  we add a subscript (e.g.,  $s_k(i) = s$  means that the environment seen by agent  $k$  is at state  $s$  at time  $i$ ).

All vectors are column vectors. Superscript  $\cdot^\top$  denotes transposition. The identity matrix of size  $S$  is denoted by  $I_S$ , the null matrix of size  $M \times L$  is denoted by  $0_{M \times L}$ , and  $\mathbf{1}_M$  and  $\mathbf{0}_M$  stand for vectors of ones and zeros of length  $M$ , respectively. The Kronecker product operation is denoted by  $\otimes$ . The spectrum,  $m$ -th eigenvalue and spectral radius of a matrix are denoted by  $\lambda(\cdot)$ ,  $\lambda_m(\cdot)$  and  $\rho(\cdot)$ , respectively. The operator  $\text{col}\{\cdot\}$  stacks vectors (or matrices) into a long vector (or a tall matrix); while  $\text{vec}[\cdot]$  stacks the columns of a matrix, one beneath the other, into a long vector. The operator  $\text{diag}\{\cdot\}$  creates a diagonal matrix (a block-diagonal

matrix) from a given vector (a set of square matrices). The Euclidean (semi)norm is given by  $\|y\|_D^2 \triangleq y^\top D y$ , where  $D$  is a positive (semi)definite matrix. The expected value operator with respect to probability distribution  $d$  is denoted by  $\mathbb{E}_d[\cdot]$ ; we use multiple sub-indexes (e.g.,  $\mathbb{E}_{d,\phi,\mathcal{P}}[\cdot]$ ) when the expectation is taken with regard to multiple distributions.

## II. BELLMAN EQUATION AND VALUE FUNCTIONS

### A. Markov decision processes (MDP)

We consider Markov decision processes (MDP) [6], [7] that are characterized by a finite set of states  $\mathbb{S}$  of size  $S \triangleq |\mathbb{S}|$ ; a finite set of actions  $\mathbb{A}$ ; the kernel of transition probabilities  $\mathcal{P}(s'|s, a)$ , which gives the probability of going from one state  $s$  to another state  $s'$ , given an action  $a$ ; and the reward function  $r : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \rightarrow \mathbb{R}$  that the agent wants to predict, which is associated with every transition, such that  $r(s, a, s')$  denotes the reward received by a generic agent for the transition from  $s$  to  $s'$  after taking action  $a$ .

The agents want to predict the response of their environment when they follow some stationary policy  $\pi$ , such that  $\pi(a|s)$  stands for the probability of an agent choosing action  $a$  when the environment is at state  $s$ . We assume that the finite-state Markov chain resulting from the MDP is irreducible and aperiodic under any policy of interest. Thus, it has a unique positive stationary probability distribution of visiting each state [6, App. A] [30] denoted by  $d^\pi = [d^\pi(1), \dots, d^\pi(S)]^\top$ , such that  $d^\pi(s) > 0$ , for all  $1 \leq s \leq S$ . The state transition probabilities of the Markov chain, from initial state  $s$  to destination  $s'$  are given by

$$p_{ss'}^\pi \triangleq \mathbb{P}(s' | s) = \sum_{a \in \mathbb{A}} \mathcal{P}(s'|s, a) \pi(a|s) \quad (1)$$

We collect  $p_{ss'}^\pi$  into an  $S \times S$  matrix  $P^\pi$  as its  $(s, s')$ -th entry.

### B. Value function

In order to make predictions of the reward signal, we use state value functions,  $v : \mathbb{S} \rightarrow \mathbb{R}$ , which provide the expected cumulative sum of the reward, weighted by an exponentially-decaying time window [2], [5]–[7]. This time window spans from  $i = 0$  to  $i = \infty$ , but it has an effective length controlled by a constant  $\gamma \in (0, 1)$ , which trades short-sighted ( $\gamma \rightarrow 0$ ) vs. long-term planning ( $\gamma \rightarrow 1$ ). The value function for target policy  $\pi$ , starting from some initial state  $s \in \mathbb{S}$  at time  $i$ , is defined as:

$$v^\pi(s) \triangleq \mathbb{E}_{\pi, \mathcal{P}} \left[ \sum_{t=1}^{\infty} \gamma^{t-1} r(i+t) \mid s(i) = s \right] \quad (2)$$

where  $r(i+1) \triangleq r(s(i), a(i), s(i+1))$ , and the expectation is taken with regard to all possible state-transitions. Note that  $a(i)$  is random because it is drawn from a probability distribution  $\pi$ , which together with the probabilistic transition dictated by  $\mathcal{P}$ , leads to a random future state  $s(i+1)$ . Let  $s'$  denote the destination state after transitioning from  $s$ . Then, some algebra will show that we can write (2) as a fixed point equation, known as the Bellman equation [2], [6], [7]:

$$v^\pi(s) = r^\pi(s) + \gamma \sum_{s' \in \mathbb{S}} p_{ss'}^\pi v^\pi(s') \quad (3)$$

where  $r^\pi(s)$  denotes the expected reward that can be collected over the next transition when the agent is currently at state  $s$ :

$$\begin{aligned} r^\pi(s) &\triangleq \mathbb{E}_{\pi, \mathcal{P}}[r(s, a, s')] \\ &= \sum_{a \in \mathbb{A}} \pi(a|s) \sum_{s' \in \mathbb{S}} \mathcal{P}(s'|s, a) r(s, a, s') \end{aligned} \quad (4)$$

Let  $v^\pi$  and  $r^\pi$  be the vectors of length  $S$  that collect the values  $v^\pi(s)$  and  $r^\pi(s)$  for all  $s \in \mathbb{S}$ , respectively:

$$v^\pi \triangleq \text{col}\{v^\pi(1), \dots, v^\pi(S)\} \quad (5)$$

$$r^\pi \triangleq \text{col}\{r^\pi(1), \dots, r^\pi(S)\} \quad (6)$$

Then, Eq. (3) can be written in vector form as the linear system of equations:

$$(I_S - \gamma P^\pi) v^\pi = r^\pi \quad (7)$$

We shall refer to  $v^\pi$  as the value vector. There are two challenges when we aim to obtain  $v^\pi$  from (7). The first challenge is that the size of the state-space can be very large (e.g., the *chess* game has  $10^{47}$  possible states, making (7) computationally intractable). The second challenge arises when the agents do not know anything about the environment, thus  $P^\pi$  and  $r^\pi$  are unavailable. In the following subsections we review how to address these two issues.

### C. Approximate value function as a saddle-point problem

For the single agent scenario, references [8], [9] introduced efficient algorithms with convergence guarantees under general conditions. The algorithms save on computations by relying on features that span a space of much lower dimensionality than the size of the original state space. More formally, let  $x : \mathbb{S} \rightarrow \mathbb{R}^M$  be some mapping from states to features, such that  $x_s$  is the *feature vector* of length  $M \ll S$  that represents the state  $s$ . Now, it would be efficient to approximate the original value function  $v^\pi(s)$  as a parametric function of  $x_s$ , for some parameter vector  $w \in \mathbb{R}^M$ . When this is done, the problem of making a prediction (i.e., estimating the value vector  $v^\pi$ ) becomes equivalent to seeking a parameter vector  $w^*$  that is optimal in a certain sense. Among many parametrizations, a linear approximation of the form  $v^\pi(s) \approx x_s^\top w$  has been extensively studied in the literature (see, e.g., [26], [31], [32]) and it is promising mainly because it leads to solutions with low computational demands. Moreover, it is expected that if one chooses the mapping of features carefully, then the linear approximation model will generally provide good results (see, e.g., [33]–[38]). Let  $X$  be the matrix of size  $S \times M$  formed by stacking the transposed feature vectors,  $x_s^\top$ , on top of each other:  $X \triangleq \text{col}\{x_1^\top, \dots, x_S^\top\} \in \mathbb{R}^{S \times M}$ . Then, the linear approximation can be expressed in vector form as:

$$v^\pi \approx X w \quad (8)$$

By modeling the value function in the form (8), we solve for  $w$  by using the approximation (8) in (7). Doing so leads to the approximate Bellman equation:

$$X w = r^\pi + \gamma P^\pi X w \quad (9)$$

In this paper, we assume that the features available for the agents constitute a linearly independent set of basis functions,

which effectively represent the states. Thus,  $X$  is *full rank* by construction. However, the fixed point equation (9) may not have a solution  $w$  in general because the right-hand side need not lie in the range space of  $X$ , which we denote by  $\mathbb{X}$ . To address this issue, one approach is to solve instead the *projected Bellman equation* [26]:

$$Xw = \Pi(r^\pi + \gamma P^\pi Xw) \quad (10)$$

where  $\Pi$  is a projection operator onto  $\mathbb{X}$ . Since  $\mathbb{X}$  is a linear space, the projection operator with respect to some metric norm  $\|\cdot\|_D$  is defined as:

$$\Pi x \triangleq \arg \min_{\bar{x} \in \mathbb{X}} \|x - \bar{x}\|_D^2 \quad (11)$$

where  $D$  is a symmetric positive-definite matrix. The matrix  $\Pi$  is given by

$$\Pi = X(X^\top DX)^{-1}X^\top D \quad (12)$$

Therefore, for different choices of  $D$ , we have different projection operators. However, some choices for  $D$  will lead to simpler solutions, as we will reveal in Subsection II-D.

Equation (10) is now an over-determined consistent linear system of equations. To solve for  $w$ , reference [9] considered the weighted least-squares problem:

$$\underset{w}{\text{minimize}} \quad J_{\text{PB}}(w) \triangleq \|\Pi(r^\pi + \gamma P^\pi Xw) - Xw\|_D^2 \quad (13)$$

where the cost function  $J_{\text{PB}}(w)$  is referred to as the *projected Bellman error*. Since  $Xw$  already lies in  $\mathbb{X}$  and  $D$  is positive definite, it can be verified that

$$J_{\text{PB}}(w) = (X^\top Dr^\pi - Bw)^\top (X^\top DX)^{-1} (X^\top Dr^\pi - Bw) \quad (14)$$

where  $B \triangleq X^\top D(I_S - \gamma P^\pi)X$ . Using (14), it can also be verified that the solution  $w^*$  that minimizes  $J_{\text{PB}}(w)$  satisfies the following normal equations [39]:

$$B^\top (X^\top DX)^{-1} Bw^* = B^\top (X^\top DX)^{-1} X^\top Dr^\pi \quad (15)$$

Since  $\|P^\pi\|_\infty = 1$  and  $\gamma < 1$ , we can bound the spectral radius of  $\gamma P^\pi$  by

$$\rho(\gamma P^\pi) \leq \|\gamma P^\pi\|_\infty = \gamma < 1 \quad (16)$$

Thus, the inverse  $(I_S - \gamma P^\pi)^{-1}$  exists. In addition, since the matrices  $D$  and  $X$  have full-rank by assumption, we conclude that matrix  $B$  is invertible, so the minimizer  $w^*$  is given by

$$w^* = (X^\top D(I_S - \gamma P^\pi)X)^{-1} X^\top Dr^\pi \quad (17)$$

If the quantities  $\{P^\pi, r^\pi\}$  were known, one would proceed to solve (17) and determine the desired vector  $w^*$  and the sought-after value vector  $v^\pi$  from (8). However, we want the agents to learn  $v^\pi$  without any prior knowledge of the environment. In other words, we cannot assume  $P^\pi$  and  $r^\pi$  are known. For this reason, we need to develop an alternative solution method. In the process of doing so, first for single-agents, we shall arrive at the same gradient temporal difference method of [9] albeit by using a fundamentally different approach involving a primal-dual argument. The approach will subsequently enable us to generalize to a fully distributed solution.

So let us continue with the single-agent case for now. Our first step relies on relating Eq. (14) to the saddle-point conditions of a convex optimization problem. Indeed, minimizing  $J_{\text{PB}}(w)$  in (13) is equivalent to the following quadratic programming problem:

$$\begin{aligned} &\underset{\varepsilon, w}{\text{minimize}} \quad \frac{1}{2} \varepsilon^\top (X^\top DX)^{-1} \varepsilon \\ &\text{s.t.} \quad \varepsilon = X^\top Dr^\pi - Bw \end{aligned} \quad (18)$$

where we have introduced the splitting variable  $\varepsilon$ . Since problem (18) is convex and satisfies Slater's condition [40], strong duality holds and the primal and dual optimal values are attained and equal and they form a saddle-point of the Lagrangian. Specifically, the Lagrangian of (18) is

$$L(\varepsilon, w, \theta) = \frac{1}{2} \|\varepsilon\|_{(X^\top DX)^{-1}}^2 + \theta^\top (X^\top Dr^\pi - Bw - \varepsilon) \quad (19)$$

where  $\theta$  is the Lagrange multiplier. By minimizing  $L(\varepsilon, w, \theta)$  over  $\varepsilon$  and  $w$ , we obtain that the dual function is  $g(\theta) = -\infty$  unless  $B^\top \theta = 0_M$ , in which case we have

$$g(\theta) = -\frac{1}{2} \theta^\top X^\top DX \theta + \theta^\top X^\top Dr^\pi \quad (20)$$

Therefore, the dual problem of (18) is given by

$$\begin{aligned} &\underset{\theta}{\text{minimize}} \quad \frac{1}{2} \theta^\top X^\top DX \theta - \theta^\top X^\top Dr^\pi \\ &\text{s.t.} \quad B^\top \theta = 0_M \end{aligned} \quad (21)$$

The main reason to solve (21) instead of the primal problem (18) is that the dual formulation removes the inverse in the weighting matrix,  $X^\top DX$ . This transformation brings two benefits. First, in Sec. II-D, we will see that it is straightforward to optimize (21) from samples. Second, as it is explained in Sections III and IV-B, problem (21) leads to a distributed algorithm in which the agents are able to combine their individual experience to solve the problem.

Had we assumed  $P^\pi$  and  $r^\pi$  to be known, problem (21) would be trivial, with unique solution  $\theta = 0_M$ . However, since we do not assume any prior knowledge, we are going to employ instead a primal-dual algorithm that leads to an iterative stochastic-approximation mechanism to learn from samples. First, we derive the Lagrangian of (21) as

$$L(\theta, w) = \theta^\top X^\top D \left( \frac{1}{2} X \theta + (I_S - \gamma P^\pi) Xw - r^\pi \right) \quad (22)$$

where  $w$  denotes the Lagrange multiplier. We use the same notation  $w$  to denote the dual variable for (22) because it can be verified that by computing the dual of the dual problem (21) we recover the original problem (13), which is equivalent to (18). Thus, the optimal dual variable  $w^*$  of (22) is also the optimal solution to (13). To find a saddle-point  $\{\theta^*, w^*\}$  of the Lagrangian (22) we alternate between applying gradient descent to  $L(\theta, w)$  with respect to  $\theta$  and gradient ascent with respect to  $w$ :

$$\theta_{i+1} = \theta_i - \mu_\theta X^\top D (X \theta_i + (I_S - \gamma P^\pi) Xw_i - r^\pi) \quad (23a)$$

$$w_{i+1} = w_i + \mu_w X^\top (I_S - \gamma P^\pi)^\top DX \theta_i \quad (23b)$$

where  $\mu_\theta$  and  $\mu_w$  are positive step-sizes.

Construction (23a)–(23b) is the well-known Arrow-Hurwicz algorithm (see, e.g., [41], [42, Ch. 9.3.3] and [43, Ch. 10]).

#### D. Primal-dual stochastic optimization

As mentioned before, since the agents do not have prior knowledge of the environment, we need to replace (23a)–(23b) by constructions that do not depend on the quantities  $\{P^\pi, r^\pi\}$ . In order to find the solution directly from samples, we need to convert these gradient iterations into stochastic approximations. The selection of an appropriate weighted norm  $\|\cdot\|_D$  in (13) now becomes relevant. If we choose a weighting matrix  $D$  that represents the probability distribution of visiting each state, then we can express the terms that appear in (23a)–(23b) as expectations that we can substitute with their sample estimates. We proceed to explain the details.

Let us set the weighting matrix in (13) equal to the state-visitation probability induced by the behavior policy (which we emphasize with the corresponding superscript), i.e.,  $D \triangleq D^\phi \triangleq \text{diag}\{d^\phi\}$ . Equations (23a)–(23b) depend on  $P^\pi$  and  $r^\pi$ , meaning that the agent aims to predict the value vector along the expected trajectory that would have been induced by the target policy  $\pi$ . However, the state-visitation distribution of this trajectory,  $d^\pi$ , does not match the distribution of the samples actually gathered by the agent, given by  $d^\phi$ . Importance sampling [44, Ch. 9.7] is a technique for estimating properties of a particular distribution, while only having samples generated from a different distribution. Let us introduce *importance weights*  $\xi(a, s) \triangleq \pi(a|s)/\phi(a|s)$  that measure the dissimilarity between the target ( $\pi$ ) and behavior ( $\phi$ ) policies. By using importance sampling, reference [23] showed that we can write the gradient inside (23a) in terms of moment values of the behavior policy as follows:

$$\begin{aligned} & X^\top D^\phi (X\theta_i + (I_S - \gamma P^\pi)Xw_i - r^\pi) \\ &= \sum_{s \in \mathbb{S}} d^\phi(s) x_s \left( x_s^\top \theta_i + \left( x_s^\top - \gamma \sum_{s' \in \mathbb{S}} p_{ss'}^\pi x_{s'}^\top \right) w_i \right. \\ &\quad \left. - \sum_{a \in \mathbb{A}} \sum_{s' \in \mathbb{S}} \mathcal{P}(s'|s, a) \pi(a|s) r(s, a, s') \right) \\ &= \sum_{s \in \mathbb{S}} \sum_{a \in \mathbb{A}} \sum_{s' \in \mathbb{S}} \mathcal{P}(s'|s, a) \pi(a|s) d^\phi(s) \\ &\quad \cdot x_s (x_s^\top \theta_i + (x_s - \gamma x_{s'})^\top w_i - r(s, a, s')) \\ &= \sum_{s \in \mathbb{S}} \sum_{a \in \mathbb{A}} \sum_{s' \in \mathbb{S}} \mathcal{P}(s'|s, a) \phi(a|s) \xi(a, s) d^\phi(s) \\ &\quad \cdot x_s (x_s^\top \theta_i + (x_s - \gamma x_{s'})^\top w_i - r(s, a, s')) \\ &= \mathbb{E}_{d^\phi, \phi, \mathcal{P}} [x_s (x_s^\top \theta_i + (x_s - \gamma x_{s'})^\top w_i \\ &\quad - r(s, a, s')) \xi(a, s)] \end{aligned} \quad (24)$$

Similarly, we can express the gradient inside (23b) as

$$\begin{aligned} & X^\top (I_S - \gamma P^\pi)^\top D^\phi X \theta_i \\ &= \mathbb{E}_{d^\phi, \phi, \mathcal{P}} [(x_s - \gamma x_{s'})^\top x_s^\top \xi(a, s)] \theta_i \end{aligned} \quad (25)$$

The agent does not know these expected values though. Rather, at every time-step, the agent observes transitions of the form  $\{x_i, a(i), x_{i+1}, r(i+1)\}$ , where  $x_i \triangleq x_{s(i)}$  denotes the feature vector observed at time  $i$ .

In addition, the agent knows both its behavior policy  $\phi$  and the target policy  $\pi$  that it wants to evaluate so it can compute the importance weight. Nevertheless, in an actual implementation, the agent need not know the states but just features, hence, the actual policies must be conditioned on the feature vectors. Slightly abusing notation, we introduce the importance weight that the node computes at every time step:

$$\xi(i) \triangleq \frac{\pi(a(i)|x_i)}{\phi(a(i)|x_i)} \approx \frac{\pi(a(i)|s(i))}{\phi(a(i)|s(i))} \triangleq \xi(a(i), s(i)) \quad (26)$$

Since a sample of a random variable is an unbiased estimator of its expected value, we can build a pair of stochastic approximation recursions from (23a)–(23b) and (24)–(25):

$$\theta_{i+1} = \theta_i - \mu_\theta x_i (x_i^\top \theta_i + \delta_{i+1}^\top w_i - r(i+1)) \xi(i) \quad (27a)$$

$$w_{i+1} = w_i + \mu_w \delta_{i+1} x_i^\top \theta_i \xi(i) \quad (27b)$$

where we introduced  $\delta_{i+1} \triangleq x_i - \gamma x_{i+1}$ . Recursions (27a)–(27b) coincide with the single-agent gradient-temporal difference (GTD2) algorithm, which was derived in [9] using a different approach. The above derivation from (18) to (27b) shows that GTD2 is a stochastic Arrow-Hurwicz algorithm applied to the dual problem of (13). More importantly, as we will see in the following sections, the primal-dual approach is convenient for a multi-agent formulation, since it leads to a meaningful in-network state-visitation distribution that combines the individual stationary distributions of the agents, thus overcoming non-exploratory individual behavior policies.

### III. MULTI-AGENT LEARNING

We now consider a network of  $N$  connected agents that operate in similar but independent MDPs. The state-space  $\mathbb{S}$ , action-space  $\mathbb{A}$ , and transition probabilities  $\mathcal{P}$  are the same for every node, but their actions do not influence each other. Thus, the transition probabilities seen by each agent  $k$  are only determined by its own actions,  $a_k(i) \in \mathbb{A}$ , and the previous state of its environment,  $s_k(i) \in \mathbb{S}$ :

$$s_k(i+1) \sim \mathcal{P}(\cdot | s_k(i), a_k(i)), \quad k = 1 \dots N \quad (28)$$

This assumption is convenient because it makes the problem stationary without forcing each agent to know the actions and feature vectors of every other agent in the network. The agents aim to predict the response of their environment to a common *target* policy  $\pi$  while they follow different *behavior* policies, denoted by  $\phi_k(a|s)$  each.

Motivated by recent results on network behavior in [21], [45], we note that, through collaboration, each agent may contribute to the network with its own experience. Let  $D^{\phi_k}$  be the diagonal matrix that represents the stationary state-visitation distribution for agent  $k$ . We then introduce the following global problem in place of (21) with  $D$  substituted by  $D^{\phi_k}$ :

$$\begin{aligned} & \underset{\theta}{\text{minimize}} \quad \sum_{k=1}^N \tau_k \left( \frac{1}{2} \theta^\top X^\top D^{\phi_k} X \theta - \theta^\top X^\top D^{\phi_k} r^\pi \right) \\ & \text{s.t.} \quad \sum_{k=1}^N \tau_k (X^\top D^{\phi_k} (I_S - \gamma P^\pi) X)^\top \theta = 0 \end{aligned} \quad (29)$$

where  $\tau = [\tau_1, \dots, \tau_N]^\top$  is a vector of non-negative parameters whose purpose is to weight the contribution of each agent's local problem to the global problem, such that  $\tau^\top \mathbf{1}_N = 1$ . Since the dual problem (21) removes the inverse of the weighting matrices  $X^\top D^{\phi_k} X$ , we can introduce the in-network stationary distribution

$$D^{\bar{\phi}} \triangleq \sum_{k=1}^N \tau_k D^{\phi_k} \quad (30)$$

Note that solving the aggregated problem (29) is effectively solving the single-agent problem (21) with  $D$  replaced by  $D^{\bar{\phi}}$ . The Lagrangian of (29) is given by

$$L(\theta, w) = \sum_{k=1}^N \tau_k L_k(\theta, w) \quad (31)$$

where the individual Lagrangians are given by

$$L_k(\theta, w) = \theta^\top X^\top D^{\phi_k} \left( \frac{1}{2} X\theta + (I_S - \gamma P^\pi) Xw - r^\pi \right) \quad (32)$$

which are similar to (22) but with stationary distribution  $D^{\phi_k}$ . In order to find the global saddle-point of the aggregate Lagrangian (31) in a cooperative and stochastic manner, we apply diffusion strategies [19]–[21]. We choose the *adapt-then-combine* (ATC) diffusion variant for distributed optimization over networks [17], [18], [45]. The algorithm consists of two-steps: the *adaptation* step, at which every agent updates its own intermediate estimate independently of the other agents; and the *combination* step, at which every agent combines its neighbors' estimates. Similar to the derivation of the single-agent algorithm (27a)–(27b), we can express the gradient of the individual Lagrangians (32) in terms of moment values (i.e., replacing  $D^{\bar{\phi}}$  by  $D^{\phi_k}$  into (24)–(25)). We then follow a primal-dual approach and apply ATC twice: *i*) for minimizing  $L_k(\theta, w)$  in (32) over  $\theta$  through stochastic gradient descent:

$$\hat{\theta}_{k,i+1} = \theta_{k,i} - \mu_\theta x_{k,i} (x_{k,i}^\top \theta_{k,i} + \delta_{k,i+1}^\top w_{k,i} - r_k(i+1)) \xi_k(i) \quad (33a)$$

$$\theta_{k,i+1} = \sum_{l \in \mathcal{N}_k} c_{lk} \hat{\theta}_{l,i+1} \quad (33b)$$

and *ii*) for maximizing  $L_k(\theta, w)$  in (32) over  $w$  through stochastic gradient ascent:

$$\hat{w}_{k,i+1} = w_{k,i} + \mu_w \delta_{k,i+1} x_{k,i}^\top \theta_{k,i} \xi_k(i) \quad (33c)$$

$$w_{k,i+1} = \sum_{l \in \mathcal{N}_k} c_{lk} \hat{w}_{l,i+1} \quad (33d)$$

where  $\mathcal{N}_k$  stands for the neighborhood of agent  $k$  (i.e., the set of agents that are able to communicate with agent  $k$  in a single hop, including  $k$  itself),  $\hat{\theta}$  and  $\hat{w}$  correspond to the locally adapted estimates, and  $\theta$  and  $w$  correspond to the combined estimates for the adapt-then-combine strategy. The combination coefficients  $\{c_{lk}\}$  define the weights on the links in the network and can be chosen freely by the designer, as long as they satisfy:

$$c_{lk} \geq 0, \quad \sum_{l \in \mathcal{N}_k} c_{lk} = 1, \quad c_{lk} = 0 \text{ if } l \notin \mathcal{N}_k \quad (34)$$

$$c_{kk} > 0 \text{ for at least one agent } k \quad (35)$$

Let  $C \triangleq [c_{lk}]$  be the combination matrix. Then, condition (34) implies that  $C$  is left-stochastic. Condition (35) means that there is at least one agent that trusts its local measurements and is able to perform its own adaptation step. We also assume that the topology of the network is connected (i.e., there is at least one path between any pair of nodes) and that the combination matrix  $C$  remains fixed over time. Therefore, conditions (34)–(35) ensure that  $C$  is a primitive matrix (i.e., there exists  $j > 0$  such that all entries of  $C^j$  are strictly positive) [19], [46]. It follows from the Perron-Frobenius Theorem [47] that  $C$  has a unique eigenvalue at one, while all other eigenvalues are strictly inside the unit circle. We normalize the entries of the eigenvector that is associated with the eigenvalue at one to add up to one and refer to it as the Perron eigenvector of  $C$ . All its entries will be strictly positive. We will show in Sec. IV-G and App. B that the values for  $\{\tau_k\}$  turn out to be determined by this Perron eigenvector.

Iterations (33a)–(33d) constitute the proposed diffusion off-policy GTD algorithm, which we remark is a fully distributed algorithm because the combination step is taken only over  $\mathcal{N}_k$ .

---

**Algorithm 1:** Diffusion off-policy GTD algorithm.  
This procedure runs in parallel at every node  $k$ .

---

**Inputs:** Target  $\pi$  and behavior  $\phi_k$  policies, neighborhood  $\mathcal{N}_k$ , weights  $\{c_{lk}, l = 1, \dots, N\}$ , and step-sizes  $\mu_\theta, \mu_w$   
**Initialize** estimates  $\theta_{k,0}, w_{k,0}$   
**for** every time-step  $i = 1$  to  $T$  **do**  
    Take action  $a_k(i) \sim \phi_k(\cdot | x_{k,i})$   
    Observe feature vector  $x_{k,i+1}$  and reward  $r_k(i+1)$   
    Perform local adaptation steps (33a) and (33c)  
    Combine in-neighborhood estimates into  $\theta_{k,i+1}, w_{k,i+1}$  using (33b) and (33d)  
**end for**  
**Return:**  $w_{k,T+1}$

---

## IV. PERFORMANCE ANALYSIS

In this section we analyze the existence and uniqueness of the optimal solution to the multi-agent learning problem (29). We extend the energy conservation arguments of [16]–[19] to perform a mean-square-error (MSE) analysis of the diffusion GTD algorithm (33a)–(33d) and provide convergence guarantees under sufficiently small step-sizes. We also obtain closed form expressions of the mean-square-deviation (MSD) and analyze the bias of the algorithm. We will rely on some reasonable conditions on the data, as explained next.

### A. Data model

To begin with, we model the quantities appearing in (33a)–(33d) as instantaneous realizations of random variables, which we denote by using boldface notation. We aggregate the variables into vectors of length  $2M$  each:

$$\alpha_{k,i} \triangleq \begin{bmatrix} \theta_{k,i} \\ w_{k,i} \end{bmatrix}, \quad \psi_{k,i} \triangleq \begin{bmatrix} \hat{\theta}_{k,i} \\ \hat{w}_{k,i} \end{bmatrix} \quad (36)$$

$$\mathbf{g}_{k,i+1} \triangleq \begin{bmatrix} -\eta \mathbf{x}_{k,i} \cdot \boldsymbol{\xi}_k(i) \cdot \mathbf{r}_k(i+1) \\ 0_M \end{bmatrix} \quad (37)$$

where we are now writing  $\mu_w \triangleq \mu$  and  $\mu_\theta \triangleq \eta \mu_w$ , such that  $\eta > 0$  is the step-size ratio between the two adaptation steps. We further introduce the following  $2M \times 2M$  coefficient matrix:

$$\mathbf{G}_{k,i+1} \triangleq \begin{bmatrix} \eta \mathbf{x}_{k,i} \mathbf{x}_{k,i}^\top \boldsymbol{\xi}_k(i) & \eta \mathbf{x}_{k,i} \boldsymbol{\delta}_{k,i+1}^\top \boldsymbol{\xi}_k(i) \\ -\boldsymbol{\delta}_{k,i+1} \mathbf{x}_{k,i}^\top \boldsymbol{\xi}_k(i) & 0_{M \times M} \end{bmatrix} \quad (38)$$

Then, the diffusion algorithm (33a)–(33d) with stochastic variables can be expressed as

$$\boldsymbol{\psi}_{k,i+1} = \boldsymbol{\alpha}_{k,i} - \mu (\mathbf{G}_{k,i+1} \boldsymbol{\alpha}_{k,i} + \mathbf{g}_{k,i+1}) \quad (39a)$$

$$\boldsymbol{\alpha}_{k,i+1} = \sum_{l \in \mathcal{N}_k} c_{lk} \boldsymbol{\psi}_{l,i+1} \quad (39b)$$

We assume the following conditions for (39a)–(39b):

**Assumption 1.** *The state transitions  $\{(s_k(i), s_k(i+1))\}$  visited by each agent  $k$  are i.i.d. samples, with initial states  $\{s_k(i)\}$  drawn from the stationary distribution  $d^{\phi_k}$ .*

**Assumption 2.** *There is some positive probability that every state is visited by at least one agent, thus  $D^{\bar{\phi}}$  in (30) is positive-definite.*

**Assumption 3.** *The feature matrix  $X$  and the expected reward signal  $r^\pi$  are bounded from below and from above.*

Given the sequence of states visited by each agent  $\{s_k(1), s_k(2), \dots, s_k(i), \dots\}$ , the segments that start and end at the same state are independent of one another. When the Markov chain that defines these state transitions has short mixing time, these segments tend to be short (see, e.g., [30]). Assumption 1 approximates these independent segments with sequences of just one step. This is a customary approximation (see, e.g., [8], [9], [48]) that simplifies the analysis because the tuples  $\{\mathbf{x}_{k,i}, \mathbf{a}_k(i), \mathbf{x}_{k,i+1}, \mathbf{r}_k(i)\}$  become i.i.d. samples, rendering  $\mathbf{G}_{k,i+1}$  and  $\mathbf{g}_{k,i+1}$  independent of  $\boldsymbol{\alpha}_{k,i}$ .

Assumption 2 refers to a property of the network. For a single-agent algorithm, the agent should visit every state with positive probability; otherwise it may not be able to approach the value function. Here, we impose the milder condition that every state must be visited by *at least* one agent.

Assumption 3 holds for most practical implementations, and will be used in the stability analysis.

### B. Existence and uniqueness of solution

Solving the aggregated dual problem (29) is equivalent to finding the saddle-points  $\{w^o, \theta^o\}$  of the global Lagrangian (31). A saddle-point of the Lagrangian must satisfy [40]:

$$L(\theta^o, w^o) = \min_{\theta} \max_w L(\theta, w) = \max_w \min_{\theta} L(\theta, w) \quad (40)$$

These conditions are equivalent to the following system of linear equations:

$$\nabla_{\theta} L(\theta, w) = X^\top D^{\bar{\phi}} (X\theta - r^\pi + (I_S - \gamma P^\pi)Xw) = 0_M \quad (41)$$

$$\nabla_w L(\theta, w) = X^\top (I_S - \gamma P^\pi)^\top D^{\bar{\phi}} X\theta = 0_M \quad (42)$$

To find the saddle-point  $\{\theta^o, w^o\}$ , we solve for  $\theta$  in (42) first. Since Assumption 2 establishes that  $D^{\bar{\phi}}$  has full-rank, we recall from (15)–(17) that  $X^\top D^{\bar{\phi}} (\gamma P^\pi - I_S)X$  is invertible and, hence,  $\theta^o = 0_M$ . Then, substituting  $\theta^o$  into (41) yields:

$$w^o = \left( X^\top D^{\bar{\phi}} (I_S - \gamma P^\pi) X \right)^{-1} X^\top D^{\bar{\phi}} r^\pi \quad (43)$$

Equation (43) therefore illustrates one clear benefit of cooperation. If the behavior policy of some agent prevents him from exploring the entire state-space, then some of the entries of its corresponding  $d^{\phi_k}$  will be zero and the agent may be unable to estimate the value vector on its own. Nevertheless, as long as any other agent in the network can visit these unexplored states, the matrix  $D^{\bar{\phi}}$  will be positive-definite, guaranteeing the existence and uniqueness of a solution  $w^o$ .

We remark that the off-policy solution  $w^o$  in (43) is in fact an approximation to the *on-policy* solution that the agents wish to predict, which is given by (17) when  $D \triangleq D^\pi$ :

$$w^\pi = \left( X^\top D^\pi (I_S - \gamma P^\pi) X \right)^{-1} X^\top D^\pi r^\pi \quad (44)$$

That is, the obtained solution (43) is still an approximation of (44) because  $D^{\bar{\phi}}$  is not necessarily the same as  $D^\pi$ . However, it is interesting to realize that, by using diffusion strategies, the agents can estimate the exact on-policy solution if the scalars  $\{\tau_k\}$  could be set to satisfy

$$\sum_{k=1}^N \tau_k d^{\phi_k} = d^\pi \quad \Leftrightarrow \quad w^o = w^\pi \quad (45)$$

In the next subsections, we analyze the conditions that allow diffusion GTD to converge to (43).

### C. Error recursion

We introduce the following error measures, which measure the difference between the estimates  $\{\boldsymbol{\alpha}_{k,i}, \boldsymbol{\psi}_{k,i}\}$  at time  $i$  and the optimal solution  $\alpha^o = \text{col}\{\theta^o, w^o\}$  for each agent  $k$ :

$$\tilde{\boldsymbol{\psi}}_{k,i} \triangleq \alpha^o - \boldsymbol{\psi}_{k,i} \quad (46)$$

$$\tilde{\boldsymbol{\alpha}}_{k,i} \triangleq \alpha^o - \boldsymbol{\alpha}_{k,i} \quad (47)$$

Then, subtracting both sides of (39a)–(39b) from  $\alpha^o$ , we obtain

$$\tilde{\boldsymbol{\psi}}_{k,i+1} = (I_{2M} - \mu \mathbf{G}_{k,i+1}) \tilde{\boldsymbol{\alpha}}_{k,i} + \mu (\mathbf{G}_{k,i+1} \alpha^o + \mathbf{g}_{k,i+1}) \quad (48)$$

Using the fact that  $c_{lk} = 0$  if  $l \notin \mathcal{N}_k$ , the error recursion for the combination step becomes

$$\tilde{\boldsymbol{\alpha}}_{k,i} = \sum_{l \in \mathcal{N}_k} c_{lk} \tilde{\boldsymbol{\psi}}_{l,i} = \sum_{l=1}^N c_{lk} \tilde{\boldsymbol{\psi}}_{l,i} \quad (49)$$

We collect the error variables from across the network into block vectors of size  $2MN$ :

$$\tilde{\boldsymbol{\psi}}_i \triangleq \text{col}\{\tilde{\boldsymbol{\psi}}_{1,i}, \dots, \tilde{\boldsymbol{\psi}}_{N,i}\} \quad (50)$$

$$\tilde{\boldsymbol{\alpha}}_i \triangleq \text{col}\{\tilde{\boldsymbol{\alpha}}_{1,i}, \dots, \tilde{\boldsymbol{\alpha}}_{N,i}\} \quad (51)$$

Let  $\mathcal{C}$  and  $\mathcal{R}_i$  be matrices of size  $2MN \times 2MN$  defined by

$$\mathcal{C} \triangleq \mathcal{C} \otimes I_{2M} \quad (52)$$

$$\mathcal{R}_i \triangleq \text{diag}\{\mathbf{G}_{1,i}, \dots, \mathbf{G}_{N,i}\} \quad (53)$$

and let  $\mathcal{G}_i$  be the matrix of size  $2MN \times 2M$  defined by

$$\mathcal{G}_i \triangleq \text{col}\{\mathbf{G}_{1,i}, \dots, \mathbf{G}_{N,i}\} \quad (54)$$

We also introduce the vectors of length  $2MN$ :

$$\mathbf{g}_i \triangleq \text{col}\{\mathbf{g}_{1,i}, \dots, \mathbf{g}_{N,i}\} \quad (55)$$

$$\mathbf{n}_i \triangleq \mathcal{G}_i \alpha^o + \mathbf{g}_i \quad (56)$$

Then, the individual error recursions in (46)–(47) lead to the following network recursion:

$$\tilde{\alpha}_{i+1} = \mathcal{C}^\top (I_{2MN} - \mu \mathcal{R}_{i+1}) \tilde{\alpha}_i + \mu \mathcal{C}^\top \mathbf{n}_{i+1} \quad (57)$$

This recursion shows how the error dynamics evolves over the network over time.

#### D. Convergence in the mean

Introduce the following expected values for each agent:

$$\begin{aligned} G_k &\triangleq \mathbb{E} \mathbf{G}_{k,i} \\ &= \begin{bmatrix} \eta X^\top D^{\phi_k} X & \eta X^\top D^{\phi_k} (I_S - \gamma P^\pi) X \\ -X^\top (I_S - \gamma P^\pi)^\top D^{\phi_k} X & 0_{M \times M} \end{bmatrix} \end{aligned} \quad (58)$$

$$\mathbf{g}_k \triangleq \mathbb{E} \mathbf{g}_{k,i} = \begin{bmatrix} -\eta X^\top D^{\phi_k} \mathbf{r}^\pi \\ 0_M \end{bmatrix} \quad (59)$$

Since Assumption 1 implies that the variables  $\mathcal{R}_{i+1}$  and  $\tilde{\alpha}_i$  are independent of each other, then by taking expectations of both sides of (57) we obtain

$$\mathbb{E} \tilde{\alpha}_{i+1} = \mathcal{C}^\top (I_{2MN} - \mu \mathcal{R}) \mathbb{E} \tilde{\alpha}_i + \mu \mathcal{C}^\top (\mathcal{G} \alpha^o + \mathbf{g}) \quad (60)$$

where

$$\mathcal{R} \triangleq \mathbb{E} \mathcal{R}_i = \text{diag}\{G_1, \dots, G_N\} \quad (61)$$

$$\mathcal{G} = \mathbb{E} \mathcal{G}_i = \text{col}\{G_1, \dots, G_N\} \quad (62)$$

$$\mathbf{g} = \mathbb{E} \mathbf{g}_i = \text{col}\{\mathbf{g}_1, \dots, \mathbf{g}_N\} \quad (63)$$

Therefore, the convergence of (60) is guaranteed when the matrix  $\mathcal{C}^\top (I_{2MN} - \mu \mathcal{R})$  is stable.

**Theorem 1** (Mean convergence). *For the data model of Section IV-A, there exists small enough step-sizes, say  $0 < \mu < \mu^o$  (for some  $\mu^o > 0$  given by (121) in Appendix A), such that the matrix  $\mathcal{C}^\top (I_{2MN} - \mu \mathcal{R})$  is stable and, therefore, the mean-error recursion (60) is stable for every agent  $k = 1, \dots, N$  and converges to the bias value given by*

$$\begin{aligned} \tilde{\alpha}_\infty &\triangleq \lim_{i \rightarrow \infty} \mathbb{E} \tilde{\alpha}_i \\ &= (I_{2MN} - \mathcal{C}^\top (I_{2MN} - \mu \mathcal{R}))^{-1} \mu \mathcal{C}^\top (\mathcal{G} \alpha^o + \mathbf{g}) \end{aligned} \quad (64)$$

*Proof:* See Appendix A. ■

As it is explained in Appendix A, the value  $\mu^o$  only depends on the inputs of the algorithm, namely, data-samples (state-features and transition rewards), the weighted-topology matrix  $\mathcal{C}$ , the cost-weights  $\{\tau_k\}$ , and the step-size ratio parameter  $\eta$ .

#### E. Mean-square stability

Although the error vector converges in the mean, we still need to ensure that it has bounded fluctuations around its fixed point value. To do so, we study the evolution and steady-state

value of the variance  $\mathbb{E} \|\tilde{\alpha}_i\|^2$ . By computing the weighted squared Euclidean (semi)norm of both sides of (57)—using an arbitrary positive (semi)definite weighting matrix  $\Sigma$  that we are free to choose—and applying the expectation operator, we obtain the following variance relation:

$$\mathbb{E} \|\tilde{\alpha}_{i+1}\|_\Sigma^2 = \mathbb{E} \|\tilde{\alpha}_i\|_{\Sigma'}^2 + 2b_\Sigma^\top \mathbb{E} \tilde{\alpha}_i + \text{Tr}(\mu^2 \Sigma \mathcal{C}^\top \mathcal{R}_n \mathcal{C}) \quad (65)$$

where

$$\begin{aligned} \Sigma' &\triangleq (I_{2MN} - \mu \mathcal{R}^\top) \mathcal{C} \Sigma \mathcal{C}^\top (I_{2MN} - \mu \mathcal{R}) \\ &\quad + \mu^2 \mathbb{E} [(\mathcal{R}_{i+1} - \mathcal{R})^\top \mathcal{C} \Sigma \mathcal{C}^\top (\mathcal{R}_{i+1} - \mathcal{R})] \end{aligned} \quad (66)$$

$$b_\Sigma \triangleq \mu \mathbb{E} \left[ \left( I_{2MN} - \mu \mathcal{R}_{i+1}^\top \right) \mathcal{C} \Sigma \mathcal{C}^\top \mathbf{n}_{i+1} \right] \quad (67)$$

$$\mathcal{R}_n \triangleq \mathbb{E} [\mathbf{n}_i \mathbf{n}_i^\top] = \mathbb{E} [(\mathcal{G}_i \alpha^o + \mathbf{g}_i)(\mathcal{G}_i \alpha^o + \mathbf{g}_i)^\top] \quad (68)$$

Let  $\sigma = \text{vec}(\Sigma)$ . Using the Kronecker product property  $\text{vec}(Y \Sigma Z) = (Z^\top \otimes Y) \text{vec}(\Sigma)$  [39], we can vectorize  $\Sigma'$  in (66) and find that its vector form is related to  $\Sigma$  via the following linear relation:  $\sigma' \triangleq \text{vec}(\Sigma') = \mathcal{F} \sigma$ , where the matrix  $\mathcal{F}$  is given by

$$\begin{aligned} \mathcal{F} &\triangleq ((I_{2MN} - \mu \mathcal{R}^\top) \mathcal{C}) \otimes ((I_{2MN} - \mu \mathcal{R}^\top) \mathcal{C}) \\ &\quad + \mu^2 \mathbb{E} \left[ ((\mathcal{R}_{i+1}^\top - \mathcal{R}^\top) \mathcal{C}) \otimes ((\mathcal{R}_{i+1}^\top - \mathcal{R}^\top) \mathcal{C}) \right] \end{aligned} \quad (69)$$

Furthermore, using the property  $\text{Tr}(\Sigma Y) = (\text{vec}[Y^\top])^\top \sigma$ , we can rewrite (65) as:

$$\mathbb{E} \|\tilde{\alpha}_{i+1}\|_\sigma^2 = \mathbb{E} \|\tilde{\alpha}_i\|_{\mathcal{F}\sigma}^2 + 2\sigma^\top \mathcal{U} \cdot \mathbb{E} \tilde{\alpha}_i + h^\top \sigma \quad (70)$$

where

$$\mathcal{U} \triangleq \mu \mathbb{E} [(\mathcal{C}^\top \mathbf{n}_{i+1}) \otimes (\mathcal{C}^\top (I_{2MN} - \mu \mathcal{R}_{i+1}))] \quad (71)$$

$$h \triangleq \mu^2 \text{vec} [\mathcal{C}^\top \mathcal{R}_n \mathcal{C}] \quad (72)$$

In (70) we are using the notation  $\|x\|_\sigma^2$  to represent  $\|x\|_\Sigma^2$ . Note that (70) is not a true recursion because the weighting matrices corresponding to  $\sigma$  and  $\mathcal{F}\sigma$  are different. Moreover, recursion (70) is coupled with the mean-error recursion (60). To study the convergence of (70) we will expand it into a state-space model following [39], [49]. Let  $L \triangleq 2MN$  and let  $p(x)$  denote the characteristic polynomial of the  $L^2 \times L^2$  matrix  $\mathcal{F}$ , given by

$$p(x) \triangleq \det(xI - \mathcal{F}) = x^{L^2} + p_{L^2-1} x^{L^2-1} + \dots + p_0 \quad (73)$$

By the Cayley-Hamilton Theorem [39], we know that every matrix satisfies its characteristic equation (i.e.,  $p(\mathcal{F}) = 0$ ), so that

$$\mathcal{F}^{L^2} = -p_0 I_{L^2} - p_1 \mathcal{F} - \dots - p_{L^2-1} \mathcal{F}^{L^2-1} \quad (74)$$

Replacing  $\sigma$  in (70) by  $\mathcal{F}^j \sigma$ ,  $j = 0, \dots, L^2 - 1$ , we can derive the following state-space model:

$$\underbrace{\begin{bmatrix} \mathbb{E} \|\tilde{\alpha}_{i+1}\|_\sigma^2 \\ \mathbb{E} \|\tilde{\alpha}_{i+1}\|_{\mathcal{F}\sigma}^2 \\ \vdots \\ \mathbb{E} \|\tilde{\alpha}_{i+1}\|_{\mathcal{F}^{L^2-1}\sigma}^2 \end{bmatrix}}_{\mathcal{W}_{i+1}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & & & \ddots & 0 \\ 0 & 0 & 0 & \dots & 1 \\ -p_0 & -p_1 & -p_2 & \dots & -p_{L^2-1} \end{bmatrix}}_{\mathcal{T}} \underbrace{\begin{bmatrix} \mathbb{E} \|\tilde{\alpha}_i\|_\sigma^2 \\ \mathbb{E} \|\tilde{\alpha}_i\|_{\mathcal{F}\sigma}^2 \\ \vdots \\ \mathbb{E} \|\tilde{\alpha}_i\|_{\mathcal{F}^{L^2-1}\sigma}^2 \end{bmatrix}}_{\mathcal{W}_i}$$



$$+ 2 \underbrace{\begin{bmatrix} \sigma^\top \mathcal{U} \\ \sigma^\top \mathcal{F} \mathcal{U} \\ \vdots \\ \sigma^\top \mathcal{F}^{L^2-1} \mathcal{U} \end{bmatrix}}_{\mathcal{Q}} \mathbb{E} \tilde{\alpha}_i + \underbrace{\begin{bmatrix} h^\top \sigma \\ h^\top \mathcal{F} \sigma \\ \vdots \\ h^\top \mathcal{F}^{L^2-1} \sigma \end{bmatrix}}_{\mathcal{Y}} \quad (75)$$

Let  $n \triangleq \mathbb{E} n_i = \mathcal{G} \alpha^o + g$ . We combine (75) with the mean-recursion (60) and rewrite them more compactly as:

$$\begin{bmatrix} \mathcal{W}_{i+1} \\ \mathbb{E} \tilde{\alpha}_{i+1} \end{bmatrix} = \begin{bmatrix} \mathcal{T} & 2\mathcal{Q} \\ 0 & \mathcal{C}^\top (I_{2MN} - \mu \mathcal{R}) \end{bmatrix} \begin{bmatrix} \mathcal{W}_i \\ \mathbb{E} \tilde{\alpha}_i \end{bmatrix} + \begin{bmatrix} \mathcal{Y} \\ \mathcal{C}^\top n \end{bmatrix} \quad (76)$$

**Theorem 2** (Mean-square stability). *Assume the step-size parameter  $\mu$  is sufficiently small so that terms that depend on higher-order powers of  $\mu$  can be ignored. Then, for the data model of Section IV-A, there exists  $0 < \mu_{\text{MS}}^o \leq \mu^o$  (for  $\mu^o$  used in Theorem 1 and given by (121) in Appendix A), such that when  $0 < \mu < \mu_{\text{MS}}^o$ , the variance recursion (76) is mean-square stable.*

*Proof:* Observe that the stability of the joint recursion (76) is equivalent to the stability of the matrices  $\mathcal{T}$  and  $\mathcal{C}^\top (I_{2MN} - \mu \mathcal{R})$ , which is further equivalent to the following conditions on their spectral radii:

$$\rho(\mathcal{C}^\top (I_{2MN} - \mu \mathcal{R})) < 1, \quad \rho(\mathcal{T}) < 1 \quad (77)$$

The first condition is the same mean-stability condition that was discussed in Theorem 1. For the second condition, we note from (75) that  $\mathcal{T}$  is in companion form, and it is known that its eigenvalues are the roots of  $p(x)$ , which are also the eigenvalues of  $\mathcal{F}$ . Therefore, a necessary and sufficient condition for the stability of  $\mathcal{T}$  is the stability of the matrix  $\mathcal{F}$ . When the step-sizes are small enough, the last term in (69) can be ignored since it depends on  $\mu^2$  and we can write

$$\mathcal{F} \approx (\mathcal{C}^\top (I_{2MN} - \mu \mathcal{R}))^\top \otimes (\mathcal{C}^\top (I_{2MN} - \mu \mathcal{R}))^\top \quad (78)$$

which is stable if  $\mathcal{C}^\top (I_{2MN} - \mu \mathcal{R})$  is stable. ■

We remark that  $0 < \mu_{\text{MS}}^o \leq \mu^o$  is chosen to dismiss higher-order powers of  $\mu$ , and that  $\mu^o$  (see Appendix A) only depends on the inputs of the algorithm (i.e., data-samples, the weighted-topology matrix  $C$ , the weights  $\{\tau_k\}$  and the parameter  $\eta$ ).

#### F. Mean-square performance

Taking the limit of both sides of (70) we obtain:

$$\begin{aligned} \lim_{i \rightarrow \infty} \mathbb{E} \|\tilde{\alpha}_{i+1}\|_\sigma^2 &= \lim_{i \rightarrow \infty} \mathbb{E} \|\tilde{\alpha}_i\|_{\mathcal{F}\sigma}^2 \\ &\quad + 2\sigma^\top \mathcal{U} \lim_{i \rightarrow \infty} \mathbb{E} \tilde{\alpha}_i + h^\top \sigma \end{aligned} \quad (79)$$

Theorem 1 guarantees that  $\lim_{i \rightarrow \infty} \mathbb{E} \tilde{\alpha}_i = \tilde{\alpha}_\infty$ , so the steady-state variance recursion in (79) leads to

$$\lim_{i \rightarrow \infty} \mathbb{E} \|\tilde{\alpha}_i\|_\sigma^2 = q^\top (I - \mathcal{F})^{-1} \sigma \quad (80)$$

where  $q \triangleq h + 2\mathcal{U} \tilde{\alpha}_\infty$ . Result (80) is useful because it allows us to derive several performance metrics through the proper selection of the free weighting parameter vector  $\sigma$  (or, equivalently, the parameter matrix  $\Sigma$ ). For example, the network mean-square-deviation (MSD) is defined as the average of the

MSD of all the agents in the network:

$$\begin{aligned} \text{MSD}^{\text{network}} &\triangleq \lim_{i \rightarrow \infty} \frac{1}{N} \sum_{k=1}^N \mathbb{E} \|\tilde{\alpha}_{k,i}\|^2 \\ &= \lim_{i \rightarrow \infty} \mathbb{E} \|\tilde{\alpha}_i\|_{\frac{1}{N} I_{2MN}}^2 \end{aligned} \quad (81)$$

Choosing the weighting matrix in (80) as  $\Sigma = I_{2MN}/N$ , we get:

$$\boxed{\text{MSD}^{\text{network}} = \frac{1}{N} q^\top (I - \mathcal{F})^{-1} \text{vec}(I_{2MN})} \quad (82)$$

We can also obtain the MSD of any particular node  $k$ , as

$$\text{MSD}_k \triangleq \lim_{i \rightarrow \infty} \mathbb{E} \|\tilde{\alpha}_i\|_{\mathcal{J}_k}^2 \quad (83)$$

where  $\mathcal{J}_k$  is a block-diagonal matrix of  $N$  blocks of size  $2M \times 2M$ , such that all blocks in the diagonal are zero except for block  $k$  which is the identity matrix. Following the same procedure as with the network MSD we obtain

$$\boxed{\text{MSD}_k = q^\top (I - \mathcal{F})^{-1} \text{vec}(\mathcal{J}_k)} \quad (84)$$

#### G. Bias analysis

We showed in (43) that, under Assumption 2, there exists a unique solution  $\alpha^o$  for the global optimization problem (29). On the other hand, the error recursion (60) converges in the mean-square sense to some bias value  $\tilde{\alpha}_\infty$ . Now, we examine under which conditions  $\tilde{\alpha}_\infty$  is small when the step-size is small. The analysis of the bias value  $\tilde{\alpha}_\infty$  in (64) is similar to the examination developed in [18, Theorem 3] for multi-objective optimization. The main difference lies in the fact that we do not assume the matrix  $\mathcal{R}$  in (61) to be symmetric.

**Theorem 3** (Bias at small step-size). *Consider the data model of Section IV-A, where the combination matrix  $C$  is primitive left-stochastic (it satisfies (34)–(35)). Suppose the Perron eigenvector that corresponds to the eigenvalue of  $C$  at one is equal to the vector of weights  $\{\tau_k\}$  in the global problem (29) (i.e.,  $\tau^\top C = \tau^\top$  and  $\tau^\top \mathbb{1}_N = 1$ ). Assume further that the step-size  $\mu$  is sufficiently small to ensure mean-square stability. Then, it holds that*

$$\boxed{\tilde{\alpha}_\infty = O(\mu)} \quad (85)$$

*Proof:* See Appendix B. ■

We remark that the bias in (85) comes from agents following different behavior policies, which means that they are solving different optimization problems, with different minimizer each. When they use diffusion strategies, the combination step pulls them toward the global solution. Nevertheless, the adaptation step pushes each agent towards the minimizer of its individual cost function. Note, however, that if all agents followed the same behavior policy, their individual optimization problems would be identical, therefore, both the adaptation and the combination steps would pull them toward the global solution and their fixed-point estimates would be unbiased with respect to the solution of the global optimization problem (29), as stated in [1]. More formally,  $G_k \triangleq G$  and  $g_k \triangleq \bar{g}$  would be the same for every agent  $k$ , and the saddle-

point conditions of the Lagrangian of the global problem (31) would not depend on the combination weights  $\{\tau_k\}$ :

$$\sum_{k=1}^N \tau_k (\bar{G}\alpha^o + \bar{g}) = \bar{G}\alpha^o + \bar{g} = 0_{2M}$$

Therefore, if all the agents followed the same behavioral policy, then  $\bar{G}\alpha^o + \bar{g} = 0_{2MN}$  and, from (64), we would conclude that  $\tilde{\alpha}_\infty = 0$ .

## V. SIMULATIONS

Consider a group of animals foraging in a 2D-world (see Figure 1). The group forms a network of  $N = 15$  agents with arbitrarily connected topology and neighborhood size  $|\mathcal{N}_k|$  varying between 2 and 9. The weights of the links (i.e., the  $c_{lk}$  elements of  $C$ ) are obtained independently by each node following an *averaging rule* [19], [50], such that equal weight is given to any member of the neighborhood, including itself (i.e.,  $c_{lk} = 1/|\mathcal{N}_k|, l \in \mathcal{N}_k$ ). Note this rule leads to a left (rather than doubly) stochastic combination matrix  $C$  that satisfies (34)–(35). We assume that the combination matrix  $C$ —and, hence, the network topology—remains fixed.

The world is a discrete, bounded square with 20 rows and 20 columns, which amounts to  $S = 400$  states. Each agent self-localizes itself in the grid by sensing a Gaussian radial basis function of its distance to  $M = 64$  fixed markers (i.e., each of these values is a feature). The agents move in four possible directions, namely,  $\mathbb{A} = \{\text{north, south, east, west}\}$ . At every time step, the agents move and consume some energy (i.e., they receive some negative reward). In the north-east corner of the world, there is food, which the agents understand as positive reward. However, there is a large area below the food with a predator that is harmful to go through, so agents receive large negative reward if they visit these states (see caption of Figure 1 to see the exact numerical values).

Since the agents are getting negative reward at every time step (because of energy consumption) they want to know how to reach the food, while losing as less energy as possible. A natural policy, denoted  $\pi_1$ , could be to go straight to the food with high (0.8) probability and low (0.2) probability of going in another direction, but then the agents would face the harmful predator and the total expected reward may be low. Thus, we say that  $\pi_1$  is a myopic policy. Another more insightful policy, denoted  $\pi_2$ , could be to take a detour and avoid the predator's area with very high (0.95) probability. Nevertheless, if the detour takes too long, then the agents would consume too much energy and it may not be worth trying. In order to evaluate which policy is better (myopic  $\pi_1$  or detour  $\pi_2$ ), the agents have to learn the value vector of each candidate-policy from samples. If the agents were learning *on-policy*, they would have to follow one candidate policy for long enough so they could apply stochastic optimization over the samples, then they would have to start again but following the other candidate policy. In other words, on-policy learning does not allow to reuse samples while evaluating different policies. The benefit of the *off-policy* formulation is that the agents can evaluate several target policies in parallel from a single data stream.

We consider the case in which the agents are territorial and tend to settle in different regions each. In other words, the behavior policies of the agents  $\{\phi_k\}$  are all different and constrain exploration to some regions of the state-space. At every time-step, each agent is attracted to the state at the center of its territory with 0.8 probability, and it moves in a different direction with 0.2 probability. Since the agents only have samples of state-transitions in their respective territories, it is difficult for them to predict the value vector for each of the target policies ( $\pi_1$  and  $\pi_2$ ). However, since they sample complementary regions of the state-space, they can collaborate, applying diffusion strategies, to learn the value vector of the two target policies and evaluate which one is better.

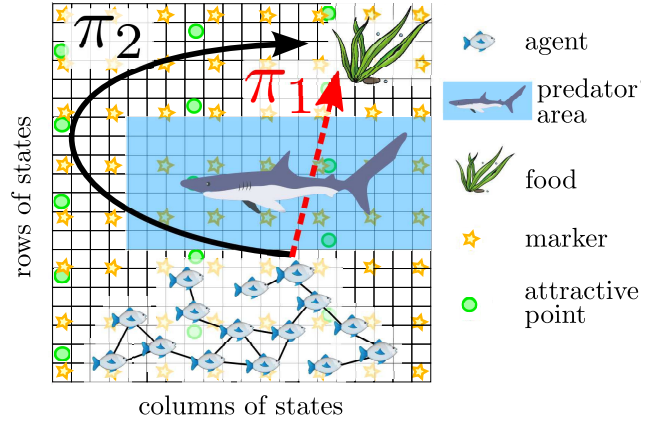


Fig. 1. Problem representation. Network of 15 randomly connected agents. States are numbered from left to right and from down to top, so that the bottom-left corner corresponds with  $s = 1$ , and the top-right corner with  $s = S = 400$ . The agents are attracted towards the center of a region each (green circle) at states  $\{395, 368, \dots, 21\}$ . The predator's area is the rectangle  $5 \leq \text{column} \leq 20, 8 \leq \text{row} \leq 15$ . The reward at every time step is  $-1$ , except when crossing the predator's area where it becomes  $-15$ , and when eating the food when it rises to  $+20$ . The 64 features are obtained with 8 Gaussian basis functions per row and per column and with standard deviation 0.005, equidistantly located at positions (yellow stars) that are computed by taking the grid as a continuous unit-square.

Figure 2 shows the learning curve of the algorithm. Since the agents have only samples from small portions of the state-space, the non-cooperative algorithm may diverge. On the other hand, when the agents cooperate (i.e., communicate their estimates to their neighbors), the diffusion algorithm allows them to benefit from the experience from other agents in the network, and they approach the same solution as a centralized architecture (i.e., with a fusion center that gathers all the samples from every node) would achieve but more efficiently, by communicating only within neighborhoods.

## VI. CONCLUSION

Diffusion GTD maintains the efficiency of the single-agent GTD2 [9], with linear complexity in both computation time and memory footprint. With diffusion GTD, the agents learn directly from samples (without any apriori knowledge of the environment) and cooperate to improve the stability and accuracy of their prediction. We remark that cooperation is fully distributed with communications only within each agent's neighborhood; neither fusion-center, nor multi-hop communications are required.

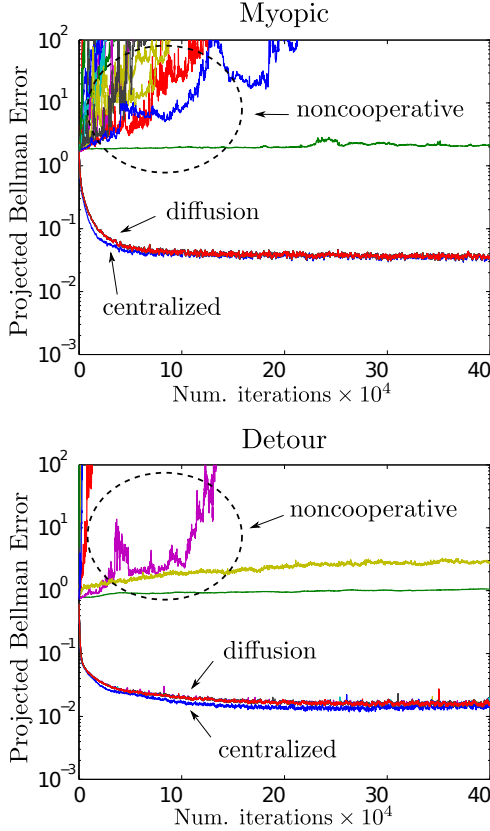


Fig. 2. Projected Bellman error (i.e.,  $J_{PB}$ ) given by Eq. (13) for centralized, diffusion and non-cooperative solutions. Results are obtained with the following parameters. We set the long-term reward to  $\gamma = 0.95$ , which we have found to make  $v^{\pi_1}$  very different from  $v^{\pi_2}$  for the specific values of  $\pi_1$ ,  $\pi_2$  and rewards used during the experiments. For both cooperative and non-cooperative nodes, the step-size and step-size ratio are  $\mu = 1 \times 10^{-3}$  and  $\eta = 0.1$ , which adequately trade bias and convergence-rate to illustrate the behavior of the algorithm. We assume every agent takes one sample at every time-step. Since a central station would have  $N$  times more samples than the rest of the agents, we set its step-size equal to  $\mu/N$ . Results are averaged over 50 independent experiments.

We provided conditions that guarantee convergence of the proposed diffusion GTD and derived performance bounds for sufficiently small step-sizes. Although our analysis assumes stationarity, constant step-sizes are a desirable feature for an adaptive network, since it allows the network to learn continuously, and to track concept drifts in the data.

#### APPENDIX A PROOF OF THEOREM 1

To study the spectrum of  $\mathcal{C}^\top (I_{2MN} - \mu\mathcal{R})$ , we express the combination coefficient matrix in its Jordan canonical form:

$$\mathcal{C}^\top = Y_C J_C Y_C^{-1} \quad (86)$$

Using the property  $(U \otimes V)(Y \otimes Z) = (UY) \otimes (VZ)$ , we obtain

$$\mathcal{C}^\top (I_{2MN} - \mu\mathcal{R}) = (Y_C \otimes I_{2M})(J_C \otimes I_{2M})(I_{2MN} - \mu(Y_C^{-1} \otimes I_{2M})\mathcal{R}(Y_C \otimes I_{2M}))(Y_C^{-1} \otimes I_{2M}) \quad (87)$$

so that, by similarity,

$$\lambda(\mathcal{C}^\top (I_{2MN} - \mu\mathcal{R})) = \lambda((J_C \otimes I_{2M})(I_{2MN} - \mu\mathcal{E})) \quad (88)$$

where

$$\mathcal{E} \triangleq (Y_C^{-1} \otimes I_{2M})\mathcal{R}(Y_C \otimes I_{2M}) \quad (89)$$

As stated in Section III, conditions (34)–(35) ensure that  $C^\top$  is a primitive right-stochastic matrix. Hence, from the Perron-Frobenius Theorem [47], the Jordan canonical form of  $C$  can be expressed as

$$J_C = \text{diag} \{1, J_C^0\} \quad (90)$$

where all the eigenvalues of  $J_C^0$  are strictly inside the unit circle. Moreover, since  $C^\top$  is right-stochastic it has one right-eigenvector of all ones associated with its unit eigenvalue, and its corresponding left eigenvector,  $p$ , has positive entries (i.e.,  $p_k > 0$ ,  $1 \leq k \leq N$ ):

$$C^\top \mathbf{1}_N = \mathbf{1}_N, \quad p^\top C^\top = p^\top, \quad \mathbf{1}_N^\top p = 1 \quad (91)$$

We therefore decompose

$$Y_C^{-1} = \text{col}\{p^\top, Y_C^l\}, \quad Y_C = [\mathbf{1}_N \ Y_C^r] \quad (92)$$

and partition  $\mathcal{E}$  as

$$\mathcal{E} = \begin{bmatrix} \bar{G} & \mathcal{E}_{12} \\ \mathcal{E}_{21} & \mathcal{E}_{22} \end{bmatrix} \quad (93)$$

$$\bar{G} \triangleq (p^\top \otimes I_{2M}) \mathcal{R}(\mathbf{1}_N \otimes I_{2M}) = \sum_{k=1}^N p_k G_k \quad (94)$$

$$\mathcal{E}_{12} \triangleq (p^\top \otimes I_{2M}) \mathcal{R}(Y_C^r \otimes I_{2M}) \quad (95)$$

$$\mathcal{E}_{21} \triangleq (Y_C^l \otimes I_{2M}) \mathcal{R}(\mathbf{1}_N \otimes I_{2M}) \quad (96)$$

$$\mathcal{E}_{22} \triangleq (Y_C^l \otimes I_{2M}) \mathcal{R}(Y_C^r \otimes I_{2M}) \quad (97)$$

Introduce the following shorthand in (88):

$$\mathcal{S} \triangleq (J_C \otimes I_{2M})(I_{2MN} - \mu\mathcal{E}) \quad (98)$$

Then, expanding (90) and (93)–(97) into (98) we have

$$\mathcal{S} = \begin{bmatrix} I_{2M} - \mu\bar{G} & -\mu\mathcal{E}_{12} \\ -\mu(J_C^0 \otimes I_{2M})\mathcal{E}_{21} & (J_C^0 \otimes I_{2M})(I_L - \mu\mathcal{E}_{22}) \end{bmatrix} \quad (99)$$

where  $L \triangleq 2M(N-1)$ . Using the same technique proposed in [51], [52], we appeal to eigenvalue perturbation analysis to examine the spectral radius of (99). We introduce the  $N \times N$  diagonal matrix  $\Omega_N^\epsilon \triangleq \text{diag}\{\epsilon, \epsilon^2, \epsilon^3, \dots, \epsilon^N\}$  with parameter  $\epsilon > 0$ . Let  $J_{\bar{G}} = Y_{\bar{G}}^{-1} \bar{G} Y_{\bar{G}}$  be the Jordan canonical form of  $\bar{G}$  and introduce the similarity transformation

$$\Phi = \begin{bmatrix} Y_{\bar{G}} \Omega_{2M}^\epsilon & 0_{2M \times L} \\ 0_{L \times 2M} & \frac{\sqrt{\mu}}{\sigma} \Omega_{N-1}^\beta \otimes I_{2M} \end{bmatrix} \quad (100)$$

with parameters  $\epsilon$ ,  $\beta$ , and  $\sigma$ . We apply the similarity transformation (100) to  $\mathcal{S}$ :

$$\Phi^{-1} \mathcal{S} \Phi = \begin{bmatrix} I_{2M} - \mu J_{\bar{G}}^\epsilon & -\frac{\mu\sqrt{\mu}}{\sigma} \mathcal{S}_{12} \\ -\sigma\sqrt{\mu} \mathcal{S}_{21} & J_C^{0\beta} \otimes I_{2M} - \mu \mathcal{S}_{22} \end{bmatrix} \quad (101)$$

where

$$\mathcal{S}_{12} \triangleq (\Omega_{2M}^\epsilon)^{-1} Y_{\bar{G}}^{-1} \mathcal{E}_{12} (\Omega_{N-1}^\beta \otimes I_{2M}) \quad (102)$$

$$\mathcal{S}_{21} \triangleq \left( ((\Omega_{N-1}^\beta)^{-1} J_C^0) \otimes I_{2M} \right) \mathcal{E}_{21} Y_{\bar{G}} \Omega_{2M}^\epsilon \quad (103)$$

$$\mathcal{S}_{22} \triangleq \left( (\Omega_{N-1}^\beta)^{-1} J_C^0 \right) \otimes I_{2M} \mathcal{E}_{22} \left( \Omega_{N-1}^\beta \otimes I_{2M} \right) \quad (104)$$

and  $J_C^\epsilon$  and  $J_C^{\beta}$  have the same form as the (upper triangular) Jordan canonical forms  $J_{\bar{G}}$  and  $J_C^0$ , except that the unit entries are replaced by  $\epsilon$  and  $\beta$ , respectively. By applying Gerschgorin theorem [47] to (101), we can identify the regions where the eigenvalues of  $\mathcal{S}$  should lie:

$$|\lambda(\mathcal{S}) - (1 - \mu\lambda_m(\bar{G}))| \leq \mu\epsilon + \frac{\mu\sqrt{\mu}}{\sigma} \sum_{q=1}^L |\mathcal{S}_{12}]_{mq}| \quad (105)$$

$$\begin{aligned} & |\lambda(\mathcal{S}) - (\lambda_{k+1}(C) - \mu[\mathcal{S}_{22}]_{mm})| \\ & \leq \beta + \sigma\sqrt{\mu} \sum_{q=1}^{2M} |\mathcal{S}_{21}]_{mq}| + \mu \sum_{\substack{q=1 \\ q \neq m}}^L |\mathcal{S}_{22}]_{mq}| \end{aligned} \quad (106)$$

where  $[\cdot]_{mq}$  stands for the element at row  $m$  and column  $q$  of a matrix. Although we use the same subscript  $m$  in both equations, note that  $1 \leq m \leq 2M$  in (105) while  $1 \leq m \leq L$  in (106); in addition, recall that  $C$  is an  $N \times N$  matrix, hence, we use subscript  $k = \text{ceil}\{m/(2M)\}$ , where  $\text{ceil}\{\cdot\}$  rounds a real number to its nearest greater or equal integer.

We are looking for sufficient conditions that guarantee that the mean recursion (60) converges for small step-size. Recall from (88) that (60) converges when  $|\lambda(\mathcal{S})| < 1$ . Let us solve for  $|\lambda(\mathcal{S})|$  in (105) first. Since  $|z| - |y| \leq |z - y|$ , we obtain

$$\begin{aligned} |\lambda(\mathcal{S})| - |1 - \mu\lambda_m(\bar{G})| & \leq |\lambda(\mathcal{S}) - (1 - \mu\lambda_m(\bar{G}))| \\ & \leq \mu\epsilon + \frac{\mu\sqrt{\mu}}{\sigma} \chi_{(12)} \end{aligned} \quad (107)$$

where  $\chi_{(12)} \triangleq \sum_{q=1}^L |\mathcal{S}_{12}]_{mq}|$ . Therefore,

$$|\lambda(\mathcal{S})| \leq \mu\epsilon + \frac{\mu\sqrt{\mu}}{\sigma} \chi_{(12)} + |1 - \mu\lambda_m(\bar{G})| \quad (108)$$

Since  $\mathcal{S}$ ,  $\bar{G}$  and  $C$  are not generally guaranteed to be symmetric, their eigenvalues may be complex. Using the fact that  $1 - z \leq (1 - \frac{1}{2}z)^2$  for  $z \in \mathbb{R}$ , we obtain

$$|1 - \mu\lambda_m(\bar{G})| \leq 1 - \mu\text{Re}\{\lambda_m(\bar{G})\} + \frac{\mu^2}{2} |\lambda_m(\bar{G})|^2 \quad (109)$$

where  $\text{Re}\{\cdot\}$  denotes the real part of a complex number. Combining (108) and (109), the stability condition implied by (105) requires finding small step-sizes  $\mu$  such that

$$\mu \frac{|\lambda_m(\bar{G})|^2}{2} + \sqrt{\mu} \frac{\chi_{(12)}}{\sigma} + \epsilon - \text{Re}\{\lambda_m(\bar{G})\} < 0 \quad (110)$$

which leads to

$$\begin{aligned} 0 < \mu < & \quad (111) \\ & \left( \frac{-\frac{\chi_{(12)}}{\sigma} + \sqrt{\left(\frac{\chi_{(12)}}{\sigma}\right)^2 + 2|\lambda_m(\bar{G})|^2(\text{Re}\{\lambda_m(\bar{G})\} - \epsilon)}}{|\lambda_m(\bar{G})|^2} \right)^2 \end{aligned}$$

where, in order to guarantee that the term inside the square root in the right side of (111) is positive, we choose

$$0 < \epsilon < \min_{1 \leq m \leq 2M} \text{Re}\{\lambda_m(\bar{G})\} \quad (112)$$

We now show that  $\text{Re}\{\lambda_m(\bar{G})\}$  is always positive. If we

transform  $\bar{G}$  into a similar matrix:

$$\begin{aligned} \bar{G}_{\sqrt{\eta}} & \triangleq \begin{bmatrix} I_M & 0_{M \times M} \\ 0_{M \times M} & \sqrt{\eta} I_M \end{bmatrix} \bar{G} \begin{bmatrix} I_M & 0_{M \times M} \\ 0_{M \times M} & \frac{1}{\sqrt{\eta}} I_M \end{bmatrix} \\ & = \begin{bmatrix} \sqrt{\eta} X^\top D^\top X & X^\top D^\top (I_S - \gamma P^\pi) X \\ -X(I_S - \gamma P^\pi)^\top D^\top X^\top & 0_{M \times M} \end{bmatrix} \end{aligned} \quad (113)$$

and use [41, Theorem 3.6] on  $\bar{G}_{\sqrt{\eta}}$ , we can establish that  $\text{Re}\{\lambda_m(\bar{G})\} > 0$ .

Now, we solve for  $|\lambda(\mathcal{S})|$  from (106). Let us abbreviate the sums in the right side of (106) as

$$\chi_{(21)} \triangleq \sum_{q=1}^{2M} |\mathcal{S}_{21}]_{mq}|, \quad \chi_{(22)} \triangleq \sum_{\substack{q=1 \\ q \neq m}}^L |\mathcal{S}_{22}]_{mq}| \quad (114)$$

In a manner similar to (107), we have

$$|\lambda(\mathcal{S})| - |\lambda_{k+1}(C) - \mu[\mathcal{S}_{22}]_{mm}| \leq \beta + \sigma\sqrt{\mu} \chi_{(21)} + \mu \chi_{(22)} \quad (115)$$

Using (115) and the fact  $|z - y| \leq |z| + |y|$  yields the following condition on  $\mu$  for stability:

$$\begin{aligned} & \sqrt{\mu}(\sqrt{\mu}(\chi_{(22)} + |\mathcal{S}_{22}]_{mm}|) + \sigma \chi_{(21)}) \\ & < 1 - |\lambda_{k+1}(C)| - \beta \end{aligned} \quad (116)$$

The following conditions on the step-size are jointly sufficient to satisfy (116):

$$0 < \sqrt{\mu} < 1 - |\lambda_{k+1}(C)| - \beta \quad (117)$$

$$0 < \sqrt{\mu} < \frac{1 - \sigma \chi_{(21)}}{\chi_{(22)} + |\mathcal{S}_{22}]_{mm}|} \quad (118)$$

From the Perron-Frobenius, we know that  $\lambda_{k+1}(C) < 1$ , for  $1 \leq k \leq N - 1$ . Moreover, Assumption 3 guarantees that any element of  $\mathcal{S}$  is bounded from below and above. Therefore, there exist parameters  $0 < \beta < 1 - |\lambda_{k+1}(C)|$  and  $0 < \sigma < 1/\chi_{(21)}$  that make the right side of (117) and (118) positive, respectively. Hence, we can square both inequalities and obtain the following conditions:

$$0 < \mu < (1 - |\lambda_{k+1}(C)| - \beta)^2 \quad (119)$$

$$0 < \mu < \left( \frac{1 - \sigma \chi_{(21)}}{\chi_{(22)} + |\mathcal{S}_{22}]_{mm}|} \right)^2 \quad (120)$$

Let bound (111), (119) and (120) by

$$\begin{aligned} \mu^o & = \min \left\{ (1 - |\lambda_{k+1}(C)| - \beta)^2, \right. \\ & \quad \left( \frac{1 - \sigma \chi_{(21)}}{\chi_{(22)} + |\mathcal{S}_{22}]_{jj}|} \right)^2, \\ & \quad \left( \frac{-\frac{\chi_{(12)}}{\sigma} + \sqrt{\left(\frac{\chi_{(12)}}{\sigma}\right)^2 + 2|\lambda_m(\bar{G})|^2(\text{Re}\{\lambda_m(\bar{G})\} - \epsilon)}}{|\lambda_m(\bar{G})|^2} \right)^2 \right\} \end{aligned} \quad (121)$$

for  $1 \leq m \leq 2M$ ,  $1 \leq k \leq N - 1$  and  $1 \leq j \leq L$ . We conclude that if the step-size  $0 < \mu < \mu^o$ , then diffusion GTD is mean-stable.

As a final remark, note that  $\mu^o$  depends on the eigenvalues of  $\bar{G}$ , the eigenvalues of the weighted-topology matrix  $C$ ,

and the constructed matrix  $\mathcal{S}$  in (99) (note that  $\sigma, \epsilon$  and  $\beta$  are similarity parameters, and the terms  $\chi_{(22)}$  and  $\chi_{(22)}$  are defined in (114) simply as a short hand of sums of the elements in  $\mathcal{S}$ ). Recall that  $\bar{G}$  is given by (94) as the weighted sum of the individual  $G_k$ , which only depend on the data samples, the importance weights and the step-size ratio parameter  $\eta$ . Finally, recall that  $\mathcal{S}$  depends on the Jordan canonical form of  $C$  and  $\mathcal{R}$ , where the latter is defined in (61) from the individual  $G_k$ . Thus, all the terms involved in  $\mu^o$  are input data to the algorithm.

## APPENDIX B PROOF OF THEOREM 3

We follow an argument similar to [18], [53]. It suffices to show that  $\lim_{\mu \rightarrow 0} \frac{\|\tilde{\alpha}_\infty\|}{\mu} = \varepsilon_o$ , where  $\varepsilon_o$  is a constant independent of  $\mu$ . Substituting (87), (89) and (99) into (64) yields

$$\tilde{\alpha}_\infty = \mu(Y_C \otimes I_{2M})(I_{2MN} - \mathcal{S})^{-1}(Y_C^{-1} \otimes I_{2M}) \cdot C^\top(\mathcal{G}\alpha^o + g) \quad (122)$$

Expanding (52), (86) and (92) into (122) leads to

$$\tilde{\alpha}_\infty = \mu(Y_C \otimes I_{2M})(I_{2MN} - \mathcal{S})^{-1}(J_C \otimes I_{2M}) \cdot \begin{bmatrix} (p^\top \otimes I_{2M})(\mathcal{G}\alpha^o + g) \\ (Y_C^l \otimes I_{2M})(\mathcal{G}\alpha^o + g) \end{bmatrix} \quad (123)$$

From now on, assume that the weights used in (29) in defining the global cost are the entries of the Perron eigenvector of  $C$ , i.e.,  $p \triangleq \tau$ . Then, the first row of the last term in (123) stands for the saddle-point conditions of the Lagrangian of the global problem (31):

$$(\tau^\top \otimes I_{2M})(\mathcal{G}\alpha^o + g) = \sum_{k=1}^N \tau_k (G_k \alpha^o + g_k) \quad (124)$$

$$= \begin{bmatrix} \eta X^\top D^\Phi(X\theta^o + (I_S - \gamma P^\pi)Xw^o - r^\pi) \\ -X^\top(I_S - \gamma P^\pi)^\top D^\Phi X\theta^o \end{bmatrix} = 0_{2M} \quad (125)$$

Therefore, expanding  $J_C$  into (123) yields

$$\tilde{\alpha}_\infty = \mu(Y_C \otimes I_{2M})(I_{2MN} - \mathcal{S})^{-1} \cdot \begin{bmatrix} 0_{2M} \\ (J_C^0 \otimes I_{2M})(Y_C^l \otimes I_{2M})(\mathcal{G}\alpha^o + g) \end{bmatrix} \quad (126)$$

From (99), we know that the upper-left block of  $I_{2MN} - \mathcal{S}$  is given by  $\mu\bar{G}$ . By using [41, Theorem 3.6], we establish that the similar matrix  $\bar{G}_{\sqrt{\eta}}$  in (113) is invertible. Therefore,  $\bar{G}$  is also invertible so we can use the following relation

$$\begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix}^{-1} = \begin{bmatrix} Z_{11}^{-1} + Z_{11}^{-1}Z_{12}UZ_{21}Z_{11}^{-1} & -Z_{11}^{-1}Z_{12}U \\ -UZ_{21}Z_{11}^{-1} & U \end{bmatrix}$$

where  $U = (Z_{22} - Z_{21}Z_{11}^{-1}Z_{12})^{-1}$ , to write

$$(I_{2MN} - \mathcal{S})^{-1} = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \quad (127)$$

In this way, relation (126) simplifies to

$$\tilde{\alpha}_\infty = \mu(Y_C \otimes I_{2M}) \begin{bmatrix} H_{12}(J_C^0 Y_C^l \otimes I_{2M})(\mathcal{G}\alpha^o + g) \\ H_{22}(J_C^0 Y_C^l \otimes I_{2M})(\mathcal{G}\alpha^o + g) \end{bmatrix} \quad (128)$$

The only terms in (128) that depend on  $\mu$  are  $H_{12}$  and  $H_{22}$ . In the limit, when  $\mu \rightarrow 0$  these two terms become independent of  $\mu$ :

$$\lim_{\mu \rightarrow 0} H_{12} = -\bar{G}^{-1}\mathcal{E}_{12}(I_{2MN} - J_C^0 \otimes I_{2M})^{-1} \quad (129)$$

$$\lim_{\mu \rightarrow 0} H_{22} = (I_{2MN} - J_C^0 \otimes I_{2M})^{-1} \quad (130)$$

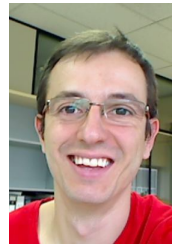
Hence, we can conclude that  $\lim_{\mu \rightarrow 0} \frac{\|\tilde{\alpha}_\infty\|}{\mu} = \varepsilon_o$ .

## REFERENCES

- [1] S. V. Macua, J. Chen, S. Zazo, and A. H. Sayed, "Cooperative off-policy prediction of Markov decision processes in adaptive networks," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, British Columbia, Canada, May 2013, pp. 4539–4543.
- [2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.
- [3] R. S. Sutton, J. Modayil, M. Delp, T. Degris, P. M. Pilarski, A. White, and D. Precup, "Horde: a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction," in *Proc. Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, vol. 2, Taipei, Taiwan, 2011, pp. 761–768.
- [4] T. Degris and J. Modayil, "Scaling-up knowledge for a cognizant robot," in *Notes AAAI Spring Symposium Series*, Palo Alto, CA, USA, 2012.
- [5] J. Modayil, A. White, and R. S. Sutton, "Multi-timescale nexting in a reinforcement learning robot," *Adaptive Behavior*, vol. 22, no. 2, pp. 146–160, 2014.
- [6] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.
- [7] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 4th ed. Athena Scientific, 2012, vol. 2.
- [8] R. S. Sutton, C. Szepesvari, and H. R. Maei, "A convergent  $O(n)$  temporal-difference algorithm for off-policy learning with linear function approximation," in *Proc. Advances in Neural Information Processing Systems (NIPS) 21*, Vancouver, British Columbia, Canada, 2008, pp. 1609–1616.
- [9] R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvari, and E. Wiewiora, "Fast gradient-descent methods for temporal-difference learning with linear function approximation," in *Proc. Int. Conf. on Machine Learning (ICML)*, Montreal, Quebec, Canada, 2009, pp. 993–1000.
- [10] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986.
- [11] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.
- [12] S. Kar and J. Moura, "Convergence rate analysis of distributed gossip (linear parameter) estimation: Fundamental limits and tradeoffs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 674–690, 2011.
- [13] S. Stankovic, M. Stankovic, and D. Stipanovic, "Decentralized parameter estimation by consensus based stochastic approximation," *IEEE Transactions on Automatic Control*, vol. 56, no. 3, pp. 531–543, 2011.
- [14] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, pp. 1520–1533, Sep 2004.
- [15] C. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3122–3136, July 2008.
- [16] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS Strategies for Distributed Estimation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1035–1048, March 2010.
- [17] J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4289–4305, Aug. 2012.
- [18] —, "Distributed Pareto optimization via diffusion strategies," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 2, pp. 205–220, Apr. 2013.
- [19] A. H. Sayed, "Diffusion adaptation over networks," in *Academic Press Library in Signal Processing*, R. Chellappa and S. Theodoridis, Eds. Elsevier, 2014, vol. 3, pp. 323–454. Also available as arXiv:1205.4220v1, May 2012.



- [20] A. H. Sayed, S.-Y. Tu, J. Chen, X. Zhao, and Z. J. Towfic, "Diffusion strategies for adaptation and learning over networks," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 155–171, May 2013.
- [21] A. H. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, April 2014.
- [22] S.-Y. Tu and A. H. Sayed, "Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 12, pp. 6217–6234, 2012.
- [23] H. R. Maei and R. S. Sutton, "GQ( $\lambda$ ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces," in *Proc. Conference on Artificial General Intelligence (AGI)*, vol. 1, Lugano, Switzerland, 2010, pp. 91–96.
- [24] S. Kar, J. M. F. 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.
- [25] L. Baird, "Residual algorithms: Reinforcement learning with function approximation," in *Proc. Int. Conf. on Machine Learning (ICML)*, Tahoe City, CA, USA, 1995, pp. 30–37.
- [26] 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.
- [27] S. Bhatnagar, "The Borkar-Meyn theorem for asynchronous stochastic approximations," *Systems and Control Letters*, vol. 60, no. 7, pp. 472–478, 2011.
- [28] J. Schneider, W.-K. Wong, A. Moore, and M. Riedmiller, "Distributed value functions," in *Proc. Int. Conf. on Machine Learning (ICML)*, Bled, Slovenia, 1999, pp. 371–378.
- [29] P. Varshavskaya, L. Kaelbling, and D. Rus, "Efficient distributed reinforcement learning through agreement," in *Distributed Autonomous Robotic Systems 8*, H. Asama, H. Kurokawa, J. Ota, and K. Sekiyama, Eds. Springer Berlin Heidelberg, 2009, pp. 367–378.
- [30] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov Chains and Mixing Times*. American Mathematical Society, 2006.
- [31] B. Scherrer, "Should one compute the temporal difference fix point or minimize the Bellman residual? The unified oblique projection view," in *Proc. Int. Conf. on Machine Learning (ICML)*, Haifa, Israel, 2010, pp. 959–966.
- [32] M. Geist and O. Pietquin, "Parametric value function approximation: A unified view," in *IEEE Symp. on Adaptive Dynamic Programming And Reinforcement Learning (ADPRL)*, Paris, France, 2011, pp. 9–16.
- [33] I. Menache, S. Mannor, and N. Shimkin, "Basis function adaptation in temporal difference reinforcement learning," *Annals of Operations Research*, vol. 134, pp. 215–238, 2005.
- [34] R. Parr, L. Li, G. Taylor, C. Painter-Wakefield, and M. Littman, "An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning," in *Proc. Int. Conf. on Machine Learning (ICML)*, Helsinki, Finland, 2008, pp. 752–759.
- [35] H. Yu and D. P. Bertsekas, "Basis function adaptation methods for cost approximation in MDP," in *Proc. IEEE Symp. on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, Nashville, TN, USA, 2009, pp. 74–81.
- [36] S. Mahadevan, "Learning representation and control in Markov decision processes: New frontiers," *Foundations and Trends in Machine Learning*, vol. 1, no. 4, pp. 403–565, Apr. 2009.
- [37] B. Boots and G. J. Gordon, "Predictive state temporal difference learning," in *Proc. Advances in Neural Information Processing Systems (NIPS)* 23, 2010, pp. 271–279.
- [38] M. G. Bellemare, J. Veness, and M. Bowling, "Sketch-based linear value function approximation," in *Proc. Advances in Neural Information Processing Systems (NIPS)* 25, 2012, pp. 2222–2230.
- [39] A. H. Sayed, *Adaptive Filters*. John Wiley & Sons, 2008.
- [40] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
- [41] M. Benzi, G. H. Golub, and J. Liesen, "Numerical solution of saddle point problems," *Acta Numerica*, vol. 14, pp. 1–137, 2005.
- [42] B. T. Polyak, *Introduction to Optimization*. Optimization Software Inc., 1987.
- [43] K. J. Arrow, L. Hurwicz, and H. Uzawa, *Studies in Linear and Non-linear Programming*. Stanford University Press, 1958.
- [44] D. P. Kroese, T. Taimre, and Z. I. Botev, *Handbook for Monte Carlo methods*. Wiley, 2011.
- [45] J. Chen and A. H. Sayed, "On the limiting behavior of distributed optimization strategies," in *Proc. Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, USA, October 2012, pp. 1535–1542.
- [46] E. Seneta, *Non-negative Matrices and Markov Chains*. Springer, 2006.
- [47] R. Horn and C. Johnson, *Matrix Analysis*. Cambridge University Press, 1990.
- [48] V. S. Borkar and S. Meyn, "The O.D.E. method for convergence of stochastic approximation and reinforcement learning," *SIAM Journal on Control and Optimization*, vol. 38, pp. 447–469, 1999.
- [49] T. Y. Al-Naffouri and A. H. Sayed, "Transient analysis of data-normalized adaptive filters," *IEEE Transactions on Signal Processing*, vol. 51, no. 3, pp. 639–652, 2003.
- [50] V. Blondel, J. Hendrickx, A. Olshevsky, and J. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in *Proc. IEEE Conf. on Decision and Control, and European Control Conf. (CDC-ECC)*, Seville, Spain, 2005, pp. 2996–3000.
- [51] X. Zhao and A. H. Sayed, "Asynchronous adaptation and learning over networks — Part II: Performance analysis," *submitted for publication*. Also available as arXiv:1312.5438, Dec. 2013.
- [52] J. Chen and A. H. Sayed, "The learning behavior of adaptive networks — Part I: Transient analysis," *submitted for publication*. Also available as arXiv:1312.7581, Dec. 2013.
- [53] X. Zhao and A. H. Sayed, "Performance limits for distributed estimation over LMS adaptive networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 10, pp. 5107–5124, 2012.



**Sergio Valcarcel Macua** received his B.S. in Telecommunications Engineering and M.S. in Electronic Systems from Technical University of Madrid (UPM), where he is currently a PhD candidate.

Before starting his PhD, he worked several years for wireless, broadcasting and telecom companies. He has been Research Associate at Robotics Institute, Carnegie Mellon University, and Research Visitor at the Adaptive Systems Laboratory at University of California, Los Angeles, and at SequeL team at INRIA Nord, Lille. His research interests include

reinforcement learning, nonconvex optimization, bandit theory and statistical signal processing in distributed multiagent systems.



**Jianshu Chen** (S'10) received his B.S. and M.S. degrees in Electronic Engineering from Harbin Institute of Technology in 2005 and Tsinghua University in 2009, respectively. He received his Ph.D degree in Electrical Engineering from University of California, Los Angeles (UCLA) in 2014.

Between 2009 and 2014, he was a member of the Adaptive Systems Laboratory (ASL) at UCLA. He is currently a postdoctoral researcher in Microsoft Research, Redmond, WA. His research interests include statistical signal processing, stochastic approximation, parallel and distributed optimization, large-scale machine learning, deep learning, reinforcement learning, and smart grid.



**Santiago Zazo** is Telecom Engineer and Dr. Engineer by the Universidad Politécnica de Madrid (UPM) in 1990 and 1995 respectively. From 1991 to 1994 he was with the University of Valladolid and from 1995 to 1997 with the University Alfonso X El Sabio in Madrid. In 1998 he joined UPM as Associate Professor in Signal Theory and Communications. His main research activities are in the field of Signal Processing with applications to Audio, Communications and Radar. More recently, he has been mostly focused on MIMO communications and

Wireless Sensor Networks from both physical layer and networking points of view. Since 1990 he is author / coauthor of more than 20 journal papers and more than 100 conference papers. He has also led many private and national Spanish projects and has participated in six European projects, leading three of them.



**Ali H. Sayed** (S'90M'92SM'99F'01) is Professor and former Chairman of electrical engineering at the University of California, Los Angeles, where he directs the UCLA Adaptive Systems Laboratory. His research activities involve several areas including adaptation and learning, network science, information processing theories, distributed processing, statistical data analysis, and biologically-inspired designs. He is coauthor of the textbook *Linear Estimation* (Prentice Hall, 2000), author of the textbooks *Fundamentals of Adaptive Filtering* (Wiley, 2003),

and *Adaptive Filters* (Wiley, 2008), and author of the monograph *Adaptation, Learning, and Optimization over Networks* (NOW, 2014).

He is also a Fellow of the American Association for the Advancement of Science (AAAS). His work has been recognized with several awards including the 2014 Athanasios Papoulis Award from the European Association for Signal Processing, the 2013 Meritorious Service Award and the 2012 Technical Achievement Award from the IEEE Signal Processing Society, the 2005 Terman Award from the American Society for Engineering Education, the 2003 Kuwait Prize, and the 1996 Donald G. Fink Prize from IEEE. He served as a Distinguished Lecturer of the IEEE Signal Processing Society in 2005. His articles received best paper awards from the IEEE Signal Processing Society in 2002, 2005, and 2012. He has been active in serving the Signal Processing community in various roles. Among other activities, he served as Editor-in-Chief of the *IEEE Transactions on Signal Processing* (2003–2005), Editor-in-Chief of the *EURASIP J. on Advances in Signal Processing* (2006–2007), General Chairman of ICASSP (Las, Vegas, 2008), and Vice-President of Publications of the IEEE Signal Processing Society (2009–2011). He also served as member of the Board of Governors (2007–2011), Awards Board (2005), Publications Board (2003–2005), Conference Board (2007–2011), and Technical Directions Board (2008–2009) of the same Society. He is recognized as a Highly Cited Researcher by Thomson Reuters.