COOPERATIVE OFF-POLICY PREDICTION OF MARKOV DECISION PROCESSES IN ADAPTIVE NETWORKS

Sergio Valcarcel Macua[†] *Jianshu Chen*[⋆] *Santiago Zazo*[†] *Ali H. Sayed*[⋆]

[†] Escuela Técnica Superior de Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, Madrid 28040, Spain

[⋆] Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA

ABSTRACT

We apply diffusion strategies to propose a cooperative reinforcement learning algorithm, in which agents in a network communicate with their neighbors to improve predictions about their environment. The algorithm is suitable to learn off-policy even in large state spaces. We provide a mean-square-error performance analysis under constant step-sizes. The gain of cooperation in the form of more stability and less bias and variance in the prediction error, is illustrated in the context of a classical model. We show that the improvement in performance is especially significant when the behavior policy of the agents is different from the target policy under evaluation.

Index Terms— adaptive networks, dynamic programming, diffusion strategies, gradient temporal difference, mean-square-error, reinforcement learning.

1. INTRODUCTION

Consider the problem in which a network of autonomous agents collaborate to predict the response of their environment to their actions. The network forms a connected graph (i.e., there is at least one path between every pair of nodes) and communication is only allowed within each neighborhood. The agents operate in an environment that is modeled as a Markov Decision Process (MDP) [1]. The MDP is characterized by a finite and countable set of states, $\mathcal{S}$, a finite and countable set of actions, $\mathcal{A}$, and the kernel of transition probabilities from one state to another given an action, $\mathcal{P}$. Although the agents can communicate with their neighbors, each agent is assumed to operate a similar but independent MDP. Therefore, the state transition probability at each agent $k = 1 \dots N$ is only determined by its own action and the previous state of its environment, i.e., $s_k(i+1) \sim \mathcal{P}(\cdot|s_k(i), a_k(i))$, where $s_k(i) \in \mathcal{S}$ denotes the state of the environment seen by agent k at time i, and $a_k(i) \in \mathcal{A}$ stands for its action.

We assume agents follow some stationary *policy*. A policy π is a mapping $\pi : \mathcal{S} \to \Theta(\mathcal{A})$, where $\Theta(\mathcal{A})$ is the set of all probability distributions over $\mathcal{A}$. We denote by $\pi(a, s) = \mathbb{P}(a|s)$ the probability of an agent choosing action a when it follows policy π and the environment is at state s.

At every time step, there is a *reward* function, $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \Re$, which agents receive and which they want to predict. Let $r(i) = r\big(s(i-1), a(i-1), s(i)\big)$ denote the reward received by a generic agent, for the transition from $s(i-1)$ to $s(i)$, after taking action $a(i-1)$. In order to make predictions of the reward signal, we use state general-value-functions (GVF), $v : \mathcal{S} \to \Re$, which provide the expected cumulative sum of the reward until the agent reaches a state that marks the end of the prediction period, Ω. Introduce the termination-indicator function $\gamma : \mathcal{S} \to \{0, 1\}$, which sets the time-interval of the prediction period so that $\gamma(s(i)) \equiv 0$ indicates that the agent has reached Ω at time i. We denote $\boldsymbol{\phi}$ the random *stopping time* at which the agent reaches the termination state Ω. Then, the GVF for an arbitrary initial state $\xi \in \mathcal{S}$ is defined by:

$$v(\xi) \triangleq \mathbb{E}\left[\sum_{i=0}^{\boldsymbol{\phi}} \gamma\big(\boldsymbol{s}(i)\big) \cdot r\big(\boldsymbol{s}(i), \boldsymbol{a}(i), \boldsymbol{s}(i+1)\big) \Big| s(0) = \xi, \boldsymbol{a}(i) \sim \pi\right] \tag{1}$$

where we are using the boldface notation to denote random variables, and where the notation $\boldsymbol{a}(i) \sim \pi$ means that we are interested in the reward received by following the policy π, which we name the *target policy*. The policy π may be different from the actual behavior policy followed by the agents, and denoted by π_b. The prediction problem that corresponds to the situation $\pi \neq \pi_b$ is called *off-policy* learning. We can see the functions r, π and γ are a grammar that the agents use to ask questions about the expected future response of their environment, and the GVF given by (1) is the answer to these questions (see [2, 3, 4] and Section 4).

In many real applications, agents do not have access to the state, but to a feature vector of the state. Also, in problems with very large state-space dimension, even if the state of the system can be known precisely, it is computationally more efficient to work with carefully chosen features [2, 5] with much smaller dimension M than $|\mathcal{S}|$. Let $x : \mathcal{S} \to \Re^M$ be some mapping from states to features. Then, we denote $x_{k,i} = x(s_k(i))$ the $M \times 1$ feature vector that represents (maybe "roughly") the state of the agent k at time i. Furthermore, note that $v(\xi)$ in (1) could be a nontrivial function of ξ. Therefore, it would be efficient to approximate $v(\xi)$ as a linear function of $x(\xi)$, say, $x(\xi)^T\omega$, where $\omega \in \Re^M$ denotes a parameter vector. In this way, we would only need to learn the parameter ω in order to approximate (or learn) $v(\xi)$. Such an approximation would provide good results if we carefully choose the mapping of features. Stacking the output of (1), for every possible state $\xi \in \mathcal{S}$, into the vector $v = \mathrm{col}\{v(\xi)\} \in \Re^{|\mathcal{S}|}$, we express the linear approximation as $v \approx X\omega$, where $X \in \Re^{|\mathcal{S}| \times M}$ is a non-singular matrix that is formed by stacking the feature row vectors $x^T(\xi)$, for every state $\xi \in \mathcal{S}$, on top of each other.

Now, the problem becomes that of seeking a parameter vector ω^o that is optimal in a certain sense using the available data that arrive sequentially at each agent k (i.e., the tuples $\{x_{k,i}, r_k(i), \gamma_k(i)\}$ where $r_k(i) = r\big(s_k(i-1), a_k(i-1), s_k(i)\big)$ and $\gamma_k(i) = \gamma(s_k(i))$).

In the context of a single agent scenario (i.e., without co-

[†]This work was supported in part by the Spanish Ministry of Science and Innovation grants TEC2009-14219-C03-01, TEC2010-21217-C02-02-CR4HFDVL, in the program CONSOLIDER-INGENIO 2010 under the grant CSD2008-00010 COMONSENS; and by the European Commission under the grant FP7-ICT-2009-4-248894-WHERE-2. [⋆]This work was supported in part by NSF grant CCF-1011918.

operation), reference [6] proposed the mean-square-projected-Bellman-error (MSPBE) as a performance criterion for finding ω (see also [7, 8, 9]). However, the MSPBE depends on both π and π_b. Nevertheless, using the importance sampling weights, $\tau_k(i) \triangleq \pi\big(a_k(i), s_k(i)\big)/\pi_b\big(a_k(i), s_k(i)\big)$, reference [10] showed that the MSPBE can be expressed as a product of expectations, which are taken with respect to the same distribution induced by π_b. Specifically, let $e_{k,i} = \tau_k(i)x_{k,i}$, then the MSPBE can be expressed as

$$J_k(\omega) = \mathbb{E}[\boldsymbol{\delta}_k^{\omega}(i)\cdot \boldsymbol{e}_{k,i}]^{\top}\mathbb{E}[\boldsymbol{x}_{k,i}\boldsymbol{x}_{k,i}^{\top}]^{-1}\mathbb{E}[\boldsymbol{\delta}_k^{\omega}(i)\cdot \boldsymbol{e}_{k,i}] \quad (2)$$

where $\boldsymbol{\delta}_k^{\omega}(i)$ is a scalar that estimates the error of the prediction:

$$\boldsymbol{\delta}_k^{\omega}(i) \triangleq \boldsymbol{r}_k(i+1) + \boldsymbol{\gamma}_k(i+1)\boldsymbol{x}_{k,i+1}^{\top}\omega - \boldsymbol{x}_{k,i}^{\top}\omega \quad (3)$$

In order to apply stochastic gradient descent to (2), we cannot sample the three expected values at every iteration, since the random quantities at each expectation would be correlated and their product would be biased. One solution from [11] is to sample only one of the expectations while tracking a long-term, quasi-stationary estimate of the others. In particular, note that the gradient of (2) is given by

$$\frac{1}{2}\nabla_{\omega}J_k(\omega) = \mathbb{E}\left[\big(\boldsymbol{\gamma}_k(i+1)\boldsymbol{x}_{k,i+1} - \boldsymbol{x}_{k,i}\big)\boldsymbol{e}_{k,i}^{\top}\right]\cdot\theta_k^o \quad (4)$$

where

$$\theta_k^o = \mathbb{E}\left[\boldsymbol{x}_{k,i}\boldsymbol{x}_{k,i}^{\top}\right]^{-1}\mathbb{E}\left[\boldsymbol{\delta}_k^{\omega}(i)\cdot\boldsymbol{e}_{k,i}\right] \quad (5)$$

It was noticed in [6] that θ_k^o in (5) is similar to the solution of the normal equations for linear least-mean-squares error estimation, so it can be approached using the least-mean-squares (LMS) algorithm.

2. DIFFUSION ADAPTATION POLICY

In the context of networked agents, we propose to find the optimal parameter vector $\omega^o \in \Re^M$ that minimizes the global cost:

$$J^{\text{glob}}(\omega) = \sum_{k=1}^{N} J_k(\omega) \quad (6)$$

where $J_k(\omega)$, $k = 1, \ldots, N$, are the individual MSPBE given in (2). In order to minimize (6) in a cooperative and fully distributed manner, we apply diffusion strategies [12, 13] on (4) and (5), obtaining the diffusion-off-policy-gradient-temporal-difference (D-OGTD) algorithm given in (7), with step-size parameters μ and η:

$$\begin{aligned}
\widehat{\theta}_{k,i} &= \theta_{k,i-1} - \eta\mu\big(x_{k,i-1}x_{k,i-1}^{\top}\theta_{k,i-1} - \delta_k^{\omega_{k,i-1}}(i-1)e_{k,i-1}\big)\\
\widehat{\omega}_{k,i} &= \omega_{k,i-1} - \mu\big((\gamma_k(i)x_{k,i} - x_{k,i-1})\cdot e_{k,i-1}^{\top}\theta_{k,i-1}\big)\\
\theta_{k,i} &= \sum_{l\in\mathcal{N}_k} b_{lk}\widehat{\theta}_{l,i}\\
\omega_{k,i} &= \sum_{l\in\mathcal{N}_k} b_{lk}\widehat{\omega}_{l,i}
\end{aligned} \quad (7)$$

where $\omega_{k,i}$ denotes the local estimate at node k of ω^o at time i. In order to consider $\theta_{k,i-1}$ constant during the update of $\widehat{\omega}_{k,i}$, the latter must be updated at a lower speed than the former. In other words, the step-size ratio, η, should satisfy $\eta \ll 1$. Note that (7) is the cooperative distributed extension of the single-agent GTD2 algorithm introduced in [6].

The matrix with the combination coefficients, $B = [b_{lk}]$, is constrained by the network topology, in the sense that non-zero elements can appear only at the locations corresponding to the active-links. These elements can be freely chosen by the designer, as long as B is a left stochastic matrix (i.e., the entries on each column of B add up to one).

3. PERFORMANCE ANALYSIS

3.1. Data Model

To analyze the performance of the distributed solution (7), we extend the energy conservation arguments of [12, 14] to carry out a mean-square-error (MSE) analysis; this is in contrast to the ordinary-differential equations method used in [6, 10, 11, 15] for the variations of the single-agent GTD algorithm. In particular, our analysis relies on studying the evolution of the following pair of stochastic equations, which are of the same nature as the updates appearing in (7):

$$\begin{aligned}
\boldsymbol{\psi}_{k,i} &= \boldsymbol{\alpha}_{k,i-1} - \mu\big(\boldsymbol{G}_{k,i}\boldsymbol{\alpha}_{k,i-1} + \boldsymbol{g}_{k,i}\big)\\
\boldsymbol{\alpha}_{k,i} &= \sum_{l\in\mathcal{N}_k} b_{lk}\boldsymbol{\psi}_{l,i}
\end{aligned} \quad (8)$$

Let us introduce the matrices

$$\boldsymbol{C}_{k,i} = \boldsymbol{x}_{k,i}\boldsymbol{x}_{k,i}^{\top},\ \ \boldsymbol{A}_{k,i} = \boldsymbol{e}_{k,i}\big(\boldsymbol{\gamma}_k(i+1)\boldsymbol{x}_{k,i+1} - \boldsymbol{x}_{k,i}\big)^{\top} \quad (9)$$

and the variables:

$$\boldsymbol{\alpha}_{k,i} = \begin{bmatrix}\boldsymbol{\theta}_{k,i}\\ \boldsymbol{\omega}_{k,i}\end{bmatrix},\quad \boldsymbol{\psi}_{k,i} = \begin{bmatrix}\widehat{\boldsymbol{\theta}}_{k,i}\\ \widehat{\boldsymbol{\omega}}_{k,i}\end{bmatrix} \quad (10)$$

$$\boldsymbol{g}_{k,i} = \begin{bmatrix}-\eta\boldsymbol{e}_{k,i-1}\boldsymbol{r}_k(i)\\ 0\end{bmatrix} \quad (11)$$

$$\boldsymbol{G}_{k,i} = \begin{pmatrix}\eta\boldsymbol{C}_{k,i-1} & -\eta\boldsymbol{A}_{k,i-1}\\ \boldsymbol{A}_{k,i-1}^{\top} & 0\end{pmatrix} \quad (12)$$

such that $\boldsymbol{\alpha}_{k,i}$, $\boldsymbol{\psi}_{k,i}$, and $\boldsymbol{g}_{k,i}$ are vectors of length $2M$ and $\boldsymbol{G}_{k,i}$ is a matrix of size $2M \times 2M$. Then, the model (8) is equivalent to (7).

We introduce $C = C_k = \mathbb{E}[\boldsymbol{C}_{k,i}]$, $A = A_k = \mathbb{E}[\boldsymbol{A}_{k,i}]$, $G = G_k = \mathbb{E}[\boldsymbol{G}_{k,i}]$ and $g = g_k = \mathbb{E}[\boldsymbol{g}_{k,i}]$, which are the same for every node $k = 1, \ldots, N$. Then, for our data model we assume the following conditions.

Assumption 1. *Samples $\{x_{k,i-1}, x_{k,i}, r_k(i), \gamma_k(i)\}$ are assumed to be drawn i.i.d. from the steady-state visitation probability distribution of the underlying MDP (induced by π_b).*

Assumption 2. *Matrices C and A are non-singular.*

Assumption 1 is customary and it is reasonable if the Markov chain of the state-process is mixing fast enough [16]; it renders $\boldsymbol{G}_{k,i}$ and $\boldsymbol{g}_{k,i}$ independent of $\boldsymbol{\alpha}_{k,i-1}$. Assumption 2 guarantees the existence and uniqueness of the fixed point of (8), α^o, such that $G\alpha^o + g = 0$; it should be satisfied when the features capture the structure of the state-space and the sample set is rich enough. Reference [6] proposed a slightly different set of aggregated variables for studying the performance of the single-agent algorithm; it showed that, under Assumption 2, the real part of the complex eigenvalues of the coefficient matrix is always positive. It turns out that G is a similarity transformation of the coefficient matrix used in [6], so we conclude that the real part of the eigenvalues of G is also always positive.

3.2. Convergence in the Mean

Introduce the following error quantities

$$\tilde{\boldsymbol{\psi}}_{k,i} \triangleq \alpha^o - \boldsymbol{\psi}_{k,i},\quad \tilde{\boldsymbol{\alpha}}_{k,i} \triangleq \alpha^o - \boldsymbol{\alpha}_{k,i} \quad (13)$$

In order to describe these relations more compactly we introduce the following *network* error vectors of length $2MN$:

$$\tilde{\boldsymbol{\psi}}_i \triangleq \text{col}\{\tilde{\boldsymbol{\psi}}_{1,i}, \ldots, \tilde{\boldsymbol{\psi}}_{N,i}\},\ \ \tilde{\boldsymbol{\alpha}}_i \triangleq \text{col}\{\tilde{\boldsymbol{\alpha}}_{1,i}, \ldots, \tilde{\boldsymbol{\alpha}}_{N,i}\} \quad (14)$$

Let $\mathcal{B} \triangleq B \otimes I_{2M}$ and $\boldsymbol{\mathcal{R}}_i \triangleq \mathrm{diag}\{\boldsymbol{G}_{1,i}, \ldots, \boldsymbol{G}_{N,i}\}$ be of size $2MN \times 2MN$ each, and $\boldsymbol{\mathcal{G}}_i \triangleq \mathrm{col}\{\boldsymbol{G}_{1,i}, \ldots, \boldsymbol{G}_{N,i}\}$ of size $2MN \times 2M$. Finally, we aggregate the network signal $\boldsymbol{g}_i \triangleq \mathrm{col}\{\boldsymbol{g}_{1,i}, \ldots, \boldsymbol{g}_{N,i}\}$ into another vector of length $2MN$, and introduce the network noise term $\boldsymbol{n}_i \triangleq \boldsymbol{\mathcal{G}}_i \alpha^o + \boldsymbol{g}_i$. Then, the individual error recursions (13) lead to the following network recursion:

$$\tilde{\boldsymbol{\alpha}}_i = \mathcal{B}^\top \big(I_{2MN} - \mu \boldsymbol{\mathcal{R}}_i\big)\tilde{\boldsymbol{\alpha}}_{i-1} + \mu \mathcal{B}^\top \boldsymbol{n}_i \tag{15}$$

Since $\mathbb{E}\boldsymbol{n}_i = \mathbb{E}\boldsymbol{\mathcal{G}}_i \alpha^o + \mathbb{E}\boldsymbol{g}_i = 0$, taking expectation of both sides of (15), we obtain, under the assumption that $\tilde{\boldsymbol{\alpha}}_{i-1}$ and $\boldsymbol{\mathcal{R}}_i$ are independent of each other,

$$\boxed{\mathbb{E}\tilde{\boldsymbol{\alpha}}_i = \mathcal{B}^\top \big(I_{2MN} - \mu \mathcal{R}\big)\mathbb{E}\tilde{\boldsymbol{\alpha}}_{i-1}} \tag{16}$$

where $\mathcal{R} \triangleq \mathbb{E}\boldsymbol{\mathcal{R}}_i$. Recursion (16) converges to zero if the matrix $\mathcal{B}^\top \big(I_{2MN} - \mu \mathcal{R}\big) = B^\top \otimes \big(I_{2MN} - \mu G\big)$ is stable. Let $\lambda_m(\cdot)$ denote the m-th eigenvalue of a matrix. Since $0 < \lambda_m(B^\top) \leq 1$ for $m = 1, \ldots, N$ we only need to ensure that the spectral radius of $\big(I_{2MN} - \mu G\big)$ is less than one. Note that G is not symmetric and therefore, it can have complex eigenvalues. Then, the stability condition can be expressed as

$$|1 - \mu\lambda_m(G)| = \sqrt{(1 - \mu \mathrm{Re}\left[\lambda_m(G)\right])^2 + \mu^2 \mathrm{Im}\left[\lambda_m(G)\right]^2} < 1 \tag{17}$$

for $m = 1, \ldots, 2M$. After some straightforward manipulations on (17), we obtain the following quadratic inequality:

$$1 - 2\mu \mathrm{Re}\left[\lambda_m(G)\right] + \mu^2 |\lambda_m(G)|^2 < 1 \tag{18}$$

Hence, since $\mathrm{Re}\left[\lambda_m(G)\right]$ is positive (see Assumption 2), we can guarantee that the mean-error recursion (16) is stable and converges to zero (i.e., $\lim_{i\to\infty} \mathbb{E}\tilde{\boldsymbol{\alpha}}_{\boldsymbol{i}} = 0$) when the step-size satisfies:

$$0 < \mu < \min_{1 \leq m \leq 2M} \frac{2\mathrm{Re}\left[\lambda_m(G)\right]}{|\lambda_m(G)|^2} \tag{19}$$

3.3. Mean-Square Stability

To ensure the error has bounded fluctuations around the zero mean value, we study the evolution and steady-state-value of the variance $\mathbb{E}\|\tilde{\boldsymbol{\alpha}}_i\|^2$. For some symmetric non-negative definite weighting matrix, Σ, taking $\|\cdot\|^2_\Sigma$ of both sides of (15) and applying the expectation operator, we obtain the following variance relation

$$\begin{aligned}\mathbb{E}\|\tilde{\boldsymbol{\alpha}}_i\|^2_\Sigma &= \mathbb{E}\|\tilde{\boldsymbol{\alpha}}_{i-1}\|^2_{\Sigma'} + 2\mu \cdot b_\Sigma^\top \, \mathbb{E}\tilde{\boldsymbol{\alpha}}_{i-1} \\ &\quad + \mu^2 \mathrm{Tr}\big(\Sigma \mathcal{B}^\top \mathcal{R}_n \mathcal{B}\big)\end{aligned} \tag{20}$$

where the weight matrix, Σ', and the cross-correlation term, b_Σ, take the form:

$$\begin{aligned}\Sigma' &= (I_{2MN} - \mu\mathcal{R}^\top)\mathcal{B}\Sigma\mathcal{B}^\top(I_{2MN} - \mu\mathcal{R}) \\ &\quad + \mu^2 \mathbb{E}\left[(\boldsymbol{\mathcal{R}}_i^\top - \mathcal{R}^\top)\mathcal{B}\Sigma\mathcal{B}^\top(\boldsymbol{\mathcal{R}}_i - \mathcal{R})\right]\end{aligned} \tag{21}$$

$$b_\Sigma = \mathbb{E}\left[\big(I_{2MN} - \mu\boldsymbol{\mathcal{R}}_i^\top\big)\mathcal{B}\Sigma\mathcal{B}^\top \boldsymbol{n}_i\right] \tag{22}$$

and the noise covariance matrix across the network is defined by

$$\mathcal{R}_n \triangleq \mathbb{E}\left[\boldsymbol{n}_i \boldsymbol{n}_i^\top\right] = \mathbb{E}\left[(\boldsymbol{\mathcal{G}}_i \alpha^o + \boldsymbol{g}_i)(\boldsymbol{\mathcal{G}}_i \alpha^o + \boldsymbol{g}_i)^\top\right] \tag{23}$$

Let $\sigma = \mathrm{vec}(\Sigma)$ denote the vectorization operation that stacks the columns of a matrix Σ on top of each other. We can vectorize Σ' in (21), leading to $\sigma' \triangleq \mathrm{vec}(\Sigma') = \mathcal{F}\sigma$, where

$$\begin{aligned}\mathcal{F} &\triangleq \Big(\big(I_{2MN} - \mu\mathcal{R}^\top\big)\mathcal{B}\Big) \otimes \Big(\big(I_{2MN} - \mu\mathcal{R}^\top\big)\mathcal{B}\Big) \\ &\quad + \mu^2 \mathbb{E}\left[\Big(\big(\boldsymbol{\mathcal{R}}_i^\top - \mathcal{R}^\top\big)\mathcal{B}\Big) \otimes \Big(\big(\boldsymbol{\mathcal{R}}_i^\top - \mathcal{R}^\top\big)\mathcal{B}\Big)\right]\end{aligned} \tag{24}$$

Therefore, we rewrite (20) in the following compact form, where we are replacing the weighting matrices by their vector representations:

$$\boxed{\mathbb{E}\|\tilde{\boldsymbol{\alpha}}_i\|^2_\sigma \;=\; \mathbb{E}\|\tilde{\boldsymbol{\alpha}}_{i-1}\|^2_{\mathcal{F}\sigma} + 2\mu \cdot \sigma^\top \mathcal{U} \cdot \mathbb{E}\tilde{\boldsymbol{\alpha}}_{i-1} + \mu^2 h^\top \sigma} \tag{25}$$

where

$$\mathcal{U} \triangleq \mathbb{E}\left[\Big(\mathcal{B}^\top\left(\boldsymbol{\mathcal{G}}_i \alpha^o + \boldsymbol{g}_i\right)\Big) \otimes \Big(\mathcal{B}^\top\left(I_{2MN} - \mu\boldsymbol{\mathcal{R}}_i\right)\Big)\right] \tag{26}$$

$$h \triangleq \mathrm{vec}\big(\mathcal{B}^\top \mathcal{R}_n \mathcal{B}\big) \tag{27}$$

Note that the mean-square-error recursion in (25) is not a true recursion because the norms are different. Moreover, it is coupled with the mean-error recursion in (16). We can expand (25) into a state-space model [12, 14, 17] that can be aggregated with (16).

Let $L \triangleq 2MN$ and let $p(x)$ be the characteristic polynomial of the $L^2 \times L^2$ matrix $\mathcal{F}$. By the Cayley-Hamilton Theorem [17], we know that every matrix satisfies its characteristic equation (i.e. $p(\mathcal{F}) = 0$), so we have

$$\mathcal{F}^{L^2} = -p_0 I_{L^2} - p_1 \mathcal{F} - \ldots - p_{L^2-1}\mathcal{F}^{L^2-1} \tag{28}$$

Replacing σ with $\mathcal{F}^j\sigma$, $j = 0, \ldots, L^2 - 1$, we obtain the following state-space model:

$$\begin{aligned}\underbrace{\begin{bmatrix} \mathbb{E}\|\tilde{\boldsymbol{\alpha}}_i\|^2_\sigma \\ \mathbb{E}\|\tilde{\boldsymbol{\alpha}}_i\|^2_{\mathcal{F}\sigma} \\ \vdots \\ \mathbb{E}\|\tilde{\boldsymbol{\alpha}}_i\|^2_{\mathcal{F}^{L^2-1}\sigma} \end{bmatrix}}_{\mathcal{W}_i} &= \underbrace{\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & & \ddots & 0 \\ 0 & 0 & 0 & \cdots & 1 \\ -p_0 & -p_1 & -p_2 & \cdots & -p_{L^2-1} \end{bmatrix}}_{\mathcal{T}} \underbrace{\begin{bmatrix} \mathbb{E}\|\tilde{\boldsymbol{\alpha}}_{i-1}\|^2_\sigma \\ \mathbb{E}\|\tilde{\boldsymbol{\alpha}}_{i-1}\|^2_{\mathcal{F}\sigma} \\ \vdots \\ \mathbb{E}\|\tilde{\boldsymbol{\alpha}}_{i-1}\|^2_{\mathcal{F}^{L^2-1}\sigma} \end{bmatrix}}_{\mathcal{W}_{i-1}} \\ &\quad + 2\mu \underbrace{\begin{bmatrix} \sigma^\top \mathcal{U} \\ \sigma^\top \mathcal{F}\mathcal{U} \\ \vdots \\ \sigma^\top \mathcal{F}^{L^2-1}\mathcal{U} \end{bmatrix}}_{\mathcal{Q}} \mathbb{E}\tilde{\boldsymbol{\alpha}}_{i-1} + \mu^2 \underbrace{\begin{bmatrix} h^\top \sigma \\ h^\top \mathcal{F}\sigma \\ \vdots \\ h^\top \mathcal{F}^{L^2-1}\sigma \end{bmatrix}}_{\mathcal{Y}}\end{aligned} \tag{29}$$

Finally, aggregating the mean-square-error recursion (29) with the mean-error recursion (16), we obtain

$$\begin{bmatrix} \mathcal{W}_i \\ \mathbb{E}\tilde{\boldsymbol{\alpha}}_i \end{bmatrix} = \begin{bmatrix} \mathcal{T} & 2\mu\mathcal{Q} \\ 0 & \mathcal{B}^\top(I_{2MN} - \mu\mathcal{R}) \end{bmatrix} \begin{bmatrix} \mathcal{W}_{i-1} \\ \mathbb{E}\tilde{\boldsymbol{\alpha}}_{i-1} \end{bmatrix} + \mu^2 \begin{bmatrix} \mathcal{Y} \\ 0 \end{bmatrix} \tag{30}$$

Observe that the stability of the joint recursion (30) is equivalent to the stability of the matrices $\mathcal{T}$ and $\mathcal{B}^\top(I_{2MN} - \mu\mathcal{R})$. We note from (29) that $\mathcal{T}$ is in companion form, and it is known that its eigenvalues are also eigenvalues of $\mathcal{F}$. When the step-sizes are small enough, we have the following approximation for $\mathcal{F}$:

$$\begin{aligned}\mathcal{F} &\approx \Big(\big(I_{2MN} - \mu\mathcal{R}^\top\big)\mathcal{B}\Big) \otimes \Big(\big(I_{2MN} - \mu\mathcal{R}^\top\big)\mathcal{B}\Big) \\ &= \Big(\mathcal{B}^\top\left(I_{2MN} - \mu\mathcal{R}\right)\Big)^\top \otimes \Big(\mathcal{B}^\top\left(I_{2MN} - \mu\mathcal{R}\right)\Big)^\top\end{aligned} \tag{31}$$

which is stable if, and only if, $\mathcal{B}^\top\left(I_{2MN} - \mu\mathcal{R}\right)$ is stable. Therefore, sufficiently small step-sizes guarantee stability in the mean and mean-square-error senses.

3.4. Mean-Square Performance

If we take the limit of both sides of (25), and use the facts that $\lim_{i\to\infty} \mathbb{E}\tilde{\boldsymbol{\alpha}}_i = 0$ and $\lim_{i\to\infty} \mathbb{E}\|\tilde{\boldsymbol{\alpha}}_{i-1}\|^2_{\mathcal{F}\sigma} = \lim_{i\to\infty} \mathbb{E}\|\tilde{\boldsymbol{\alpha}}_i\|^2_{\mathcal{F}\sigma}$ we obtain

$$\lim_{i\to\infty} \mathbb{E}\|\tilde{\boldsymbol{\alpha}}_i\|^2_\sigma = \mu^2 h^\top (I-\mathcal{F})^{-1}\sigma \tag{32}$$

Expression (32) is useful because it allows us to derive several performance metrics through the proper selection of the free weighting parameter vector σ (or, equivalently, the parameter matrix Σ). For example, the network mean-square-deviation (MSD) is defined as the average of the error of the nodes across the network:

$$\mathsf{MSD}^{\text{network}} \triangleq \lim_{i\to\infty} \frac{1}{N}\sum_{k=1}^{N} \mathbb{E}\|\tilde{\boldsymbol{\alpha}}_{k,i}\|^2 = \mathbb{E}\|\tilde{\boldsymbol{\alpha}}_i\|^2_{\frac{1}{N}I_{2MN}} \tag{33}$$

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

$$\boxed{\mathsf{MSD}^{\text{network}} = \frac{\mu^2}{N} h^\top (I-\mathcal{F})^{-1}\text{vec}(I_{2MN})} \tag{34}$$

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

$$\mathsf{MSD}_k \triangleq \lim_{i\to\infty} \mathbb{E}\|\tilde{\boldsymbol{\alpha}}_{k,i}\|^2 \tag{35}$$

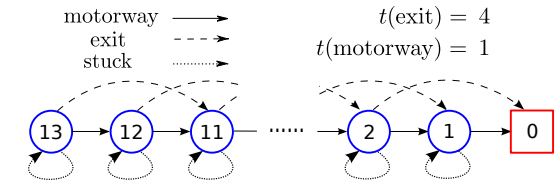
Introduce a block diagonal matrix $\mathcal{J}_k$, of $N \times N$ blocks of size $2M \times 2M$ each, such that all its blocks are zero except for block k which is the identity matrix. Then, we obtain:

$$\boxed{\mathsf{MSD}_k = \mu^2 h^\top (I-\mathcal{F})^{-1}\text{vec}(\mathcal{J}_k)} \tag{36}$$

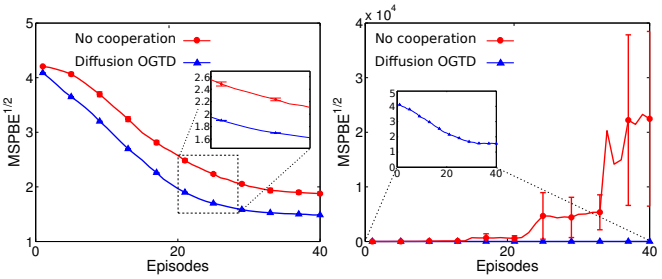
4. SIMULATIONS

Consider the following example. An automated taxi is driving on a motorway that is 14 miles long, from the suburbs to downtown. During the journey, the taxi can exchange information with other cars within its communication range. Every mile along the road, there is an exit that the autopilot can choose to take or not. If the car takes the exit, it can move towards the goal at a slower speed than in the motorway, but it could also avoid a jam so that the total trip time may be shorter. We model this example as a Markov chain with $|\mathcal{S}| = 14$ states, numbered inversely from the starting point ($\xi = 13$) to the destination ($\Omega = 0$) (this example is inspired by the classical "Boyan chain" used as benchmark in [6], see Figure 1a). If the taxi remains in the motorway, it could move to the goal at a high speed. Whenever it takes an exit, it can also advance further before getting back to the motorway, but at a slower speed. In either case, the taxi can get stuck in the same state. When an exit is taken, the probability of moving towards the destination is high and constant, P_{mov}. However, in the motorway, P_{mov} decays exponentially when getting closer to the goal. The probability of getting stuck is $1 - P_{\text{mov}}$. Let us assume that all cars follow the same conservative behavior policy, opting for the motorway most of the time. Passengers may ask whether it would be faster to take 80% of the exits. To answer this question, the agent can build a GVF setting $\gamma(\{13,\ldots,1\}) = 1$ and $\gamma(0) = 0$, with $r(i) = t(i)$, and with target policy $\pi = [0.2, 0.8]$ (i.e., $\mathbb{P}(a = \text{motorway}) = 0.2$, $\mathbb{P}(a = \text{exit}) = 0.8$ for every state). The internal representation of the state is a vector of $M = 4$ features denoting the relative position to the destination. We study two cases: in Figure 1b, all agents behave very conservatively, taking some exits only 5% of the time (i.e., $\pi_b = [0.95, 0.05]$); while in Figure 1c agents become extremely conservative using the motorway 99% of the time, which is still more biased (i.e., $\pi_b = [0.99, 0.01]$).

We simulate a network of 10 nodes (with random topology and average degree 6) plus 10 non-cooperative agents. We can expect that when the agents follow a biased policy, many state-transitions will not be sampled properly. This measurement noise leads to a wrong estimate of the optimal parameter. Cooperation through diffusion alleviates this error. In case (b), though diffusion shows an improvement in the quality of the estimate, non-cooperative agents are also able to make good predictions. It is in case (c) where the gain from cooperation is greatly appreciated: while the bias and variance (vertical bars) of the non-cooperative nodes diverge, collaborative agents achieve a good, stable estimate of the value.



(a) Markov chain of the taxi example



(b) Biased: $\pi_b = [0.95, 0.05]$ (c) Very biased: $\pi_b = [0.99, 0.01]$

Fig. 1. Taxi example. (a) State diagram, and (b)-(c) mean-square-projected-Bellman-error (MPSBE) for a target policy $\pi = [0.2, 0.8]$, estimated by 10 cooperative versus 10 non-cooperative agents. In (b), though we see some improvement in the cooperative agents, non-cooperative agents are still good at finding the estimate. In (c), when the behavior policy is very biased, there is a clear benefit of diffusion: while non-cooperative nodes diverge, cooperative agents achieve a good stable estimate. The feature vectors for states 13, 9, 5 and 1 are $x(13) = [1,0,0,0]^\top$, $x(9) = [0,1,0,0]^\top$, $x(5) = [0,0,1,0]^\top$ and $x(1) = [0,0,0,1]^\top$ respectively, for the rest of states, the features are obtained interpolating linearly between these (i.e., $x(2) = [0,0,1/4,3/4]^\top$, $x(3) = [0,0,1/2,1/2]^\top$, $x_4 = [0,0,3/4,1/4]^\top$, and so on). Step-sizes are constant $\mu = 0.1$ and $\eta = 10$. The combination coefficients of the estimates, B, are obtained using the Metropolis method [13]. Finally, $P_{\text{mov}}(\text{exit}) = 0.8, \forall s \in \mathcal{S}$, and $P_{\text{mov}}(s, \text{motorway}) = e^{\frac{s-13}{13}}, s \in \mathcal{S}$.

5. CONCLUSIONS

We proposed a distributed diffusion strategy for off-policy learning and provided a mean-square-error analysis showing that sufficiently small step-sizes guarantee convergence in the mean-square-error sense. This result complements and extends the ODE analysis of the original single-agent GTD algorithm, which requires diminishing step-sizes [10, 11, 15].

6. REFERENCES

[1] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[2] 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*, Taipei, Taiwan, 2011, vol. 2, pp. 761–768.

[3] J. Modayil, A. White, and R. S. Sutton, "Multi-timescale nexting in a reinforcement learning robot," in *From Animals to Animats 12*, T. Ziemke, C. Balkenius, and J. Hallam, Eds., vol. 7426 of *Lecture Notes in Computer Science*, pp. 299–309. Springer Berlin Heidelberg, 2012.

[4] T. Degris and J. Modayil, "Scaling-up knowledge for a cognizant robot," in *Notes of the AAAI Spring Symposium Series*, Palo Alto, CA, USA, March 2012.

[5] D. Silver, R. S. Sutton, and M. Müller, "Temporal-difference search in computer go," *Machine Learning*, vol. 87, pp. 183–219, 2012.

[6] 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*, Montreal, Quebec, Canada, 2009, pp. 993–1000.

[7] A. Antos, C. Szepesvari, and R. Munos, "Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path," *Machine Learning*, vol. 71, pp. 89–129, 2008.

[8] 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*, Haifa, Israel, June 2010, pp. 959–966, Omnipress.

[9] M. Geist and O. Pietquin, "Parametric value function approximation: a unified view," in *Proc. IEEE Symp. on Adaptive Dynamic Programming and Reinforcement Learning*, Paris, France, April 2011, pp. 9–16.

[10] H. R. Maei and R. S. Sutton, "GQ(lambda): A general gradient algorithm for temporal-difference prediction learning with eligibility traces," in *Proc. Conf. on Artificial General Intelligence*, pp. 91–96. Lugano, Switzerland, 2010.

[11] 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 *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., pp. 1609–1616. 2009.

[12] 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.

[13] A. H. Sayed, "Diffusion adaptation over networks," in *E-Reference Signal Processing*, R. Chellapa and S. Theodoridis, Eds. Elsevier, 2013. Also available as arXiv:1205.4220v1, May 2012.

[14] 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, 2010.

[15] H. R. Maei, C. Szepesvari, S. Bhatnagar, D. Precup, D. Silver, and R. S. Sutton, "Convergent temporal-difference learning with arbitrary smooth function approximation," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, Eds., pp. 1204–1212. 2009.

[16] P. Diaconis, "The mathematics of mixing things up," *Journal of Statistical Physics*, vol. 144, no. 3, pp. 445–458, 2011.

[17] A. H. Sayed, *Adaptive Filters*, John Wiley & Sons, 2008.