A Distributed Actor-Critic Algorithm and Applications to Mobile Sensor Network Coordination Problems*

Paris Pennesi,† Ioannis Ch. Paschalidis‡

***Abstract*—We introduce and establish the convergence of a distributed actor-critic method that orchestrates the coordination of multiple agents solving a general class of a Markov decision problem. The method leverages the centralized single-agent actor-critic algorithm of [1] and uses a consensus-like algorithm for updating agents' policy parameters. As an application and to validate our approach we consider a reward collection problem as an instance of a multi-agent coordination problem in a partially known environment and subject to dynamical changes and communication constraints.**

***Index Terms*—Markov decision processes, actor-critic methods, consensus, sensor networks, multi-agent coordination.**

I. Introduction

We consider a setting where a *Markov Decision Problem (MDP)* problem is to be cooperatively solved by a group of agents that can simultaneously explore the state-control space. Each agent can communicate and exchange information with agents in its vicinity, thus, having the potential to modify its own policy on the basis of the information received.

The single-agent version of the problem can be in principle solved by stochastic *Dynamic programming (DP)*. To combat Bellman's curse of dimensionality, in this paper we focus on an *Approximate Dynamic Programming (ADP)* approach: *actor-critic algorithms* [1]. In these algorithms one adopts a randomized class of policies parametrized by a (low-dimensional) parameter vector $\boldsymbol{\theta}$ and optimizes policy performance with respect to $\boldsymbol{\theta}$ by using a simulation (or a realization) of the MDP. According to its name, the algorithm interleaves two steps: (i) a policy improvement step at which it descends along the performance gradient with respect to $\boldsymbol{\theta}$ (the actor part), and (ii) a policy evaluation step at which it learns an approximate value function from a sample path that uses the current policy (the critic part).

Our main contribution is that we develop a *Distributed Multi-agent Actor-Critic (D-AC)* algorithm. Our algorithm allows us to use multiple agents to simultaneously explore the state-control space. Each agent maintains its own $\boldsymbol{\theta}$ and updates it based on local information and information received from a subset of other agents (e.g., the ones within a certain communication range). This updating follows a consensus-like algorithm; such algorithms and their analysis go back to [2] and have garnered renewed interest [3], [4]. Under suitable conditions, we show that all agents reach consensus and converge to the optimal $\boldsymbol{\theta}$. In the algorithm we present, agents update their $\boldsymbol{\theta}$'s *asynchronously*.

The D-AC algorithm provides a useful framework for agent coordination in dynamic environments. What is particularly appealing is that we solve a dynamic problem benefiting from the parallel exploration of the state-control space while incurring a relatively small communication overhead (agents only exchange their $\boldsymbol{\theta}$'s). We note that even though many less complex schemes have been proposed for agent coordination (e.g., [5]) they tend to be heuristic or solving a static problem. Our scheme attempts to approximately solve an MDP using an ADP method and it fills a void since there has not been much attention in the ADP literature on distributed approaches.

As an application, we consider multi-robot systems exploiting sensor network capabilities to cope with several tasks, i.e., coverage, surveillance, target tracking, foraging, in partially known environments subject to dynamical changes. To test the D-AC algorithm, we abstract and generalize those problems defining a *reward collection problem*, where both the positions and the values of the rewards change with time. In this problem a robot swarm explores an unknown and changing environment looking for the target points and their relative rewards. Any robot has limited sensing capabilities and is able to communicate locally with other robots. Problems of this type, but adapted to coverage control have also been studied in [5], [6]. We have also considered the coverage problem in our related preliminary work in [7]. One of the main features of our algorithm is its ability to continuously adapt to changes in the environment and the reward structure. Here, we study a different set of questions and identify an interesting trade-off: long-range agent communication (and coordination) leads to faster convergence but at the cost of performance as the resulting policy "averages" over large parts of the state-control space, thus, leading to reduced ability to locally extract as much reward as possible.

The rest of the paper is organized as follows. Sec. II introduces some preliminaries. Sec. III describes the D-AC method and establishes its convergence. Sec. IV states the reward collection problem, proposes a policy, and reports numerical results. Conclusions are in Sec. V.

* Research partially supported by the NSF under grants EFRI-0735974, DMI-0330171, ECS-0426453, and by the DOE under grant DE-FG52-06NA27490.

† RBS Global Banking & Markets 135 Bishopsgate, London EC2M 3UR, UK, e-mail: paris.pennesi@gmail.com.

‡ Corresponding author. Dept. of Electrical and Computer Eng., Division of Systems Eng., and Center for Information & Systems Eng., Boston University, 8 St. Mary's St., Boston, MA 02215, e-mail: yannisp@bu.edu, url: http://ionia.bu.edu/.

Notational Conventions: All vectors are assumed to be column vectors. We use lower case boldface letters for vectors and upper case boldface letters for matrices. Prime denotes transpose, $||\cdot||$ the Euclidean norm, $\mathbf{0}$ the vector/matrix of all zeros, and $\mathbf{e}$ the vector of all ones. The element (i,j) of a matrix $\boldsymbol{A}$ is denoted by $(\boldsymbol{A})_{ij}$.

II. Preliminaries

Consider a Markov decision process with finite state and action spaces $\mathscr{X}$ and $\mathscr{U}$, respectively. Let $c : \mathscr{X} \times \mathscr{U} \mapsto \mathbb{R}$ be a reward function. Let $\{\mu_{\boldsymbol{\theta}}, \boldsymbol{\theta} \in \mathbb{R}^n\}$ be a set of *randomized stationary policies (RSPs)*, parametrized by $\boldsymbol{\theta}$. In particular, $\mu_{\boldsymbol{\theta}}(\mathbf{u}|\mathbf{x})$ denotes the probability of taking the action $\mathbf{u}$ given the state $\mathbf{x}$, under the RSP $\boldsymbol{\theta}$. Both $\{\mathbf{x}_k\}$ and $\{(\mathbf{x}_k, \mathbf{u}_k)\}$ generated by an RSP $\boldsymbol{\theta}$, form Markov chains for every $\boldsymbol{\theta}$. We make the following assumption which is typical for actor-critic algorithms and is also made in [1].

Assumption A

(a) *For every $\mathbf{x} \in \mathscr{X}, \mathbf{u} \in \mathscr{U}$, and $\boldsymbol{\theta} \in \mathbb{R}^n$, we have $\mu_{\boldsymbol{\theta}}(\mathbf{u} \mid \mathbf{x}) > 0$.*

(b) *For every $(\mathbf{x}, \mathbf{u}) \in \mathscr{X} \times \mathscr{U}$, the mapping $\boldsymbol{\theta} \mapsto \mu_{\boldsymbol{\theta}}(\mathbf{u} \mid \mathbf{x})$ is twice differentiable. Furthermore, the function $\nabla_{\boldsymbol{\theta}} \ln \mu_{\boldsymbol{\theta}}(\mathbf{u} \mid \mathbf{x})$ is bounded and has a bounded first derivative for any fixed $\mathbf{x}$ and $\mathbf{u}$.*

(c) *For every $\boldsymbol{\theta} \in \mathbb{R}^n$ the Markov chains $\{\mathbf{x}_k\}$ and $\{(\mathbf{x}_k, \mathbf{u}_k)\}$ are irreducible and aperiodic, with stationary probabilities $\pi_{\boldsymbol{\theta}}(\mathbf{x})$ and $\eta_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}) = \pi_{\boldsymbol{\theta}}(\mathbf{x})\mu_{\boldsymbol{\theta}}(\mathbf{u}|\mathbf{x})$, respectively.*

(d) *There is a positive integer I, state $\mathbf{x}^* \in \mathscr{X}$, and $\epsilon_0 > 0$ such that for all $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_I$ it follows $\sum_{k=1}^{I} (\mathbf{P}[\boldsymbol{\theta}_1] \cdots \mathbf{P}[\boldsymbol{\theta}_k])_{\mathbf{x}\mathbf{x}^*} \geq \epsilon_0$ for all $\mathbf{x} \in \mathscr{X}$, where $\mathbf{P}[\boldsymbol{\theta}]$ denotes the transition probability for the Markov chain $\{\mathbf{x}_k\}$ under the RSP $\boldsymbol{\theta}$.*

For the setting and policy introduced in Sec. IV-A parts (a), (b) are automatically satisfied and in part (d) we can take $\mathbf{x}^* = \mathbf{0}$.

We are interested in finding a $\boldsymbol{\theta}$ that maximizes the average reward function:

$$\overline{\alpha}(\boldsymbol{\theta}) = \sum_{\mathbf{x} \in \mathscr{X}, \mathbf{u} \in \mathscr{U}} c(\mathbf{x}, \mathbf{u}) \eta_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}). \tag{1}$$

For each $\boldsymbol{\theta}$ define a differential reward function $V_{\boldsymbol{\theta}} : \mathscr{X} \mapsto \mathbb{R}$, as solution of the following Poisson equation:

$$\overline{\alpha}(\boldsymbol{\theta}) + V_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{\mathbf{u} \in \mathscr{U}} \mu_{\boldsymbol{\theta}}(\mathbf{u}|\mathbf{x}) \Big[c(\mathbf{x}, \mathbf{u}) + \sum_{\mathbf{y} \in \mathscr{X}} p(\mathbf{y}|\mathbf{x}, \mathbf{u}) V_{\boldsymbol{\theta}}(\mathbf{y}) \Big], \tag{2}$$

where $p(\mathbf{y}|\mathbf{x}, \mathbf{u})$ is the probability that the next state is $\mathbf{y}$ given that the current state is $\mathbf{x}$ and action $\mathbf{u}$ is taken. $V_{\boldsymbol{\theta}}(\mathbf{x})$ can be interpreted as the relative reward of starting at state $\mathbf{x}$, that is, the excess reward we collect on top of the average reward if we start at $\mathbf{x}$. Define the Q-value function:

$$Q_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}) = c(\mathbf{x}, \mathbf{u}) - \overline{\alpha}(\boldsymbol{\theta}) + \textstyle\sum_{\mathbf{y} \in \mathscr{X}} p(\mathbf{y}|\mathbf{x}, \mathbf{u}) V_{\boldsymbol{\theta}}(\mathbf{y}). \tag{3}$$

The following result is from [8] where for the components of $\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u})$ we write $(\psi_{\boldsymbol{\theta},1}(\mathbf{x}, \mathbf{u}), \ldots, \psi_{\boldsymbol{\theta},n}(\mathbf{x}, \mathbf{u}))$.

Theorem II.1 (Average Reward Gradient) *We have*

$$\nabla \overline{\alpha}(\boldsymbol{\theta}) = \sum_{\mathbf{x} \in \mathscr{X}, \mathbf{u} \in \mathscr{U}} \eta_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}) Q_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}) \boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}), \tag{4}$$

where

$$\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}) = \nabla_{\boldsymbol{\theta}} \ln \mu_{\boldsymbol{\theta}}(\mathbf{u}|\mathbf{x}). \tag{5}$$

The actor-critic algorithm works with a parametrization of the Q-function in terms of a vector $\mathbf{r} = (r_1, \ldots, r_m) \in \mathbb{R}^m$:

$$Q^{\mathbf{r}}_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{u}) = \textstyle\sum_{l=1}^{m} r_l \phi_{\boldsymbol{\theta},l}(\mathbf{x}, \mathbf{u}).$$

For reasons explained in [1], a typical choice for the features $\phi_{\boldsymbol{\theta},l}(\mathbf{x}, \mathbf{u})$ is to set $m = n + 1$, $\phi_{\boldsymbol{\theta},l}(\mathbf{x}, \mathbf{u}) = \psi_{\boldsymbol{\theta},l}(\mathbf{x}, \mathbf{u})$ for $l = 1, \ldots, n$, and fix $\phi_{\boldsymbol{\theta},n+1}(\mathbf{x}, \mathbf{u})$ to the constant function that is everywhere equal to one except at $\mathbf{x} = \mathbf{0}$ where $\phi_{\boldsymbol{\theta},n+1}(\mathbf{0}, \mathbf{u}) = 0$ for all $\mathbf{u}$. The critic estimates the parameter $\mathbf{r}$ on the basis of observations from a sample path of the Markov process while the actor uses $\mathbf{r}$ to compute the performance gradient and to update $\boldsymbol{\theta}$.

III. The Distributed Actor-Critic (D-AC) Method

We have N agents; the jth agent uses an RSP parametrized by $\boldsymbol{\theta}^j$ and performs the critic phase of the algorithm on the basis of its own observations. For the actor phase, each agent j uses its own estimate of $\nabla \overline{\alpha}(\boldsymbol{\theta}^j)$ and any information it receives from others. In each iteration, every agent performs a critic update followed by an actor update as described next.

A. Critic update

The critic update for each agent j is the same as in [1]. It uses a *Temporal Difference (TD)* learning algorithm ([9]) and comes in two versions, a $TD(1)$ and a $TD(\lambda)$ version which lead to different convergence results. Specifically, consider agent j and let $\alpha^j_k \in \mathbb{R}$ the average reward estimate at time k, $\mathbf{r}^j_k \in \mathbb{R}^m$ the estimate of parameter vector $\mathbf{r}$ at time k, and $\mathbf{Z}^j_k \in \mathbb{R}^m$ the estimate of Sutton's eligibility trace (see, [1]) at time k. Let also $\boldsymbol{\theta}^j_k \in \mathbb{R}^n$ be the parameter of the actor for the jth agent at time k. The updates take place at state-action pairs visited by a single sample path (potentially generated by a simulation) of the Markov process. Let $(\mathbf{X}^j_k, \mathbf{U}^j_k)$ be the state-action pair sampled at time k from the jth agent. The critic update equations for the jth agent are as follows:

$$\alpha^j_{k+1} = \alpha^j_k + \gamma^j_k (c(\mathbf{X}^j_{k+1}, \mathbf{U}^j_{k+1}) - \alpha^j_k), \tag{6}$$

$$\mathbf{r}^j_{k+1} = \mathbf{r}^j_k + \gamma^j_k d^j_k \mathbf{Z}^j_k, \tag{7}$$

$$\begin{aligned} d^j_k =& c(\mathbf{X}^j_k, \mathbf{U}^i_k) - \alpha^j_k + \mathbf{r}^{j'}_k \boldsymbol{\phi}_{\boldsymbol{\theta}^j_k}(\mathbf{X}^j_{k+1}, \mathbf{U}^j_{k+1}) \\ & - \mathbf{r}^{j'}_k \boldsymbol{\phi}_{\boldsymbol{\theta}^j_k}(\mathbf{X}^j_k, \mathbf{U}^j_k), \end{aligned} \tag{8}$$

where in the $TD(1)$ case

$$\mathbf{Z}^j_{k+1} = \begin{cases} \mathbf{Z}^j_k + \boldsymbol{\phi}_{\boldsymbol{\theta}^j_k}(\mathbf{X}^j_{k+1}, \mathbf{U}^j_{k+1}), & \text{if } \mathbf{X}^j_{k+1} \neq \mathbf{x}^*, \\ \boldsymbol{\phi}_{\boldsymbol{\theta}^j_k}(\mathbf{X}^j_{k+1}, \mathbf{U}^j_{k+1}), & \text{otherwise,} \end{cases} \tag{9}$$

and in the $TD(\lambda)$ case, for $0 < \lambda < 1$,

$$\mathbf{Z}^j_{k+1} = \lambda \mathbf{Z}^j_k + \boldsymbol{\phi}_{\boldsymbol{\theta}^j_k}(\mathbf{X}^j_{k+1}, \mathbf{U}^j_{k+1}). \tag{10}$$

In the above γ_k^j is a positive stepsize parameter, and $\mathbf{x}^*$ is a special state that the Markov process visits infinitely often. (For the application we consider in Section IV-A we can take $\mathbf{x}^* = (0,0)$.) We can think of the critic update as using TD learning to "learn" $\mathbf{r}$ (and the average reward) for a given $\boldsymbol{\theta}$. If we denote by $\boldsymbol{\psi}_{\boldsymbol{\theta},l}$ and $\boldsymbol{\phi}_{\boldsymbol{\theta},l}$ the vectors in $\mathbb{R}^{|\mathscr{X}||\mathscr{U}|}$ $(\psi_{\boldsymbol{\theta},l}(\mathbf{x},\mathbf{u});\ \forall \mathbf{x},\mathbf{u})$ and $(\phi_{\boldsymbol{\theta},l}(\mathbf{x},\mathbf{u});\ \forall \mathbf{x},\mathbf{u})$, respectively, it can be seen that the critic computes an approximate projection of Q onto the subspace spanned by the $\boldsymbol{\phi}_{\boldsymbol{\theta},l}$, $l = 1,\ldots,m$. Given our selection of the $\boldsymbol{\phi}_{\boldsymbol{\theta},l}$'s, this essentially computes a projection of Q onto the subspace spanned by the $\boldsymbol{\psi}_{\boldsymbol{\theta},l}$, $l = 1,\ldots,n$, which according to Thm. II.1 suffices for computing an approximate gradient of $\overline{\alpha}(\boldsymbol{\theta})$.

B. Actor update

The actor update is a gradient descent method which uses the above estimate of $\nabla\overline{\alpha}(\boldsymbol{\theta})$ provided by the critic. Agents communicate with each other and exchange their parameter vectors $\boldsymbol{\theta}_k^j$. Naturally, some of these messages may not be received by agents who happen to be outside the communication range of the transmitting agent. We denote by T^{ij} the set of times that agent i receives a message from agent j. We will assume that T^{ij} is either empty or infinite, that is, either j never sends messages to i or, if it does, it gets close to i often enough for its message to be received by i. Suppose i receives a message sent from j at time k. Then, we let $t^{ij}(k)$ be the time this message was sent, namely, i receives $\boldsymbol{\theta}^j_{t^{ij}(k)}$. The actor update iteration is as follows:

$$\begin{aligned}\boldsymbol{\theta}^i_{k+1} &= \mathbf{A}^{ii}_k\boldsymbol{\theta}^i_k + \textstyle\sum_{j\neq i}\mathbf{A}^{ij}_k\boldsymbol{\theta}^j_{t^{ij}(k)}\\ &+ \beta^i_k\Gamma(\mathbf{r}^i_k)\mathbf{r}^{i'}_k\boldsymbol{\phi}_{\boldsymbol{\theta}^i_k}(\mathbf{X}^i_{k+1},\mathbf{U}^i_{k+1})\boldsymbol{\psi}_{\boldsymbol{\theta}^i_k}(\mathbf{X}^i_{k+1},\mathbf{U}^i_{k+1}),\quad (11)\end{aligned}$$

where $\Gamma(\cdot)$ is a scalar that controls the stepsize β^i_k on the basis of the value of $\mathbf{r}^i_k$. Furthermore, $\mathbf{A}^{ij}_k = diag(a^{ij}_{k,1},\ldots,a^{ij}_{k,n})$, $\mathbf{A}^{ij}_k \geq \mathbf{0}$, $\sum_{j=1}^N a^{ij}_{k,l} = 1$, for all i,l,k, and $\mathbf{A}^{ij}_k = \mathbf{0}$ if $k \notin T^{ij}$ for all $i \neq j$. Notice that we set $\mathbf{A}^{ij}_k = \mathbf{0}$ if agent i does not receive the actor parameter from agent j; otherwise it receives such information and combines it with its own actor parameter as in (11). Equation (11) differs from the centralized single-agent version of the actor-critic algorithm in that it uses a convex combination of available $\boldsymbol{\theta}^j$ to update $\boldsymbol{\theta}^i$. Specifically, the centralized algorithm is a particular case of (11) where $\mathbf{A}^{ij}_k = \mathbf{0}$ if $i \neq j$ and $\mathbf{A}^{ii}_k = \mathbf{I}$ for all i.

C. Convergence

To show the convergence of the D-AC algorithm we rely on the convergence proof of the centralized actor-critic algorithm in [1] and the work on distributed stochastic gradient methods in [10]. The following assumption on information exchange between agents will be crucial in establishing that the agents reach consensus on their $\boldsymbol{\theta}$'s. It is a common assumption for consensus algorithms [4]. To state the assumption, we introduce the directed graph $\mathscr{G} = (\mathscr{N},\mathscr{E})$ with nodes $\mathscr{N} = \{1,\ldots,N\}$ corresponding to the set of agents. An arc $(j,i) \in \mathscr{E}$ if and only if T^{ij} is infinite. The graph $\mathscr{G}$ represents the communication pattern between agents.

Assumption B

(a) *There is a directed path in $\mathscr{G}$ from every node to every other node.*

(b) *there exists a positive constant γ such that $\mathbf{A}^{ii}_k \geq \gamma\mathbf{I}$, $\forall i,k$, and $\mathbf{A}^{ij}_k \geq \gamma\mathbf{I}$, $\forall i,j$ and $k \in T^{ij}$.*

(c) *The time between consecutive transmissions of $\boldsymbol{\theta}^j_k$ from agent j to i is bounded by some $B \geq 0$, $\forall (j,i) \in \mathscr{E}$.*

(d) *Communication delays are bounded by some $B_0 \geq 0$.*

Intuitively, this assumption says that (a) information can flow between any two agents, even indirectly through other agents via update (11); (b) in the update (11) the weights of the (local and directly received) $\boldsymbol{\theta}$'s are bounded away from zero; $(c),(d)$ for every agent i there is a finite delay between two consecutive receptions of a message $\boldsymbol{\theta}^j$ from every agent j directly communicating with i.

Assumption C

For all agents i the stepsizes γ^i_k, β^i_k and the function $\Gamma(\cdot)$ appearing in (11) satisfy

(a) *γ^i_k, β^i_k are deterministic, nonincreasing, and for some $d > 0$ satisfy*

$$\begin{gathered}\textstyle\sum_k \beta^i_k = \infty, \quad \sum_k(\beta^i_k)^2 < \infty,\\ \textstyle\sum_k \gamma^i_k = \infty, \quad \sum_k(\gamma^i_k)^2 < \infty, \quad \sum_k(\beta^i_k/\gamma^i_k)^d < \infty.\end{gathered}$$

(b) *For some positive constants $C_1 < C_2$:*

$$\begin{gathered}||\mathbf{r}||\Gamma(\mathbf{r}) \in [C_1,C_2], \qquad \forall \mathbf{r} \in \mathbb{R}^m,\\ ||\Gamma(\mathbf{r}) - \Gamma(\mathbf{s})|| \leq \frac{C_2||\mathbf{r}-\mathbf{s}||}{1+||\mathbf{r}||+||\mathbf{s}||}, \qquad \forall \mathbf{r},\mathbf{s} \in \mathbb{R}^m.\end{gathered}$$

Theorem III.1 (Distributed Actor-Critic) *Under Assumptions A–C the sequences $\{\boldsymbol{\theta}^i_k\}$ generated by each agent i according to the distributed actor-critic algorithm satisfy:*

(a) *in the $TD(1)$ case*

$$\liminf_{k\to\infty} ||\nabla\overline{\alpha}(\boldsymbol{\theta}^i_k)|| = 0,\ \forall i\ w.p.1;$$

b) *in the $TD(\lambda)$ case, for each $\epsilon > 0$, there exists λ such that*

$$\liminf_{k\to\infty} ||\nabla\overline{\alpha}(\boldsymbol{\theta}^i_k)|| < \epsilon,\ \forall i,\ w.p.1.$$

Proof: Consider agent i and as in [1, Sec. 6] define

$$\begin{aligned}\mathbf{H}_{\boldsymbol{\theta}^i}(\mathbf{x},\mathbf{u}) &= \boldsymbol{\psi}_{\boldsymbol{\theta}^i}(\mathbf{x},\mathbf{u})\boldsymbol{\phi}'_{\boldsymbol{\theta}^i}(\mathbf{x},\mathbf{u}),\\ \bar{\mathbf{H}}(\boldsymbol{\theta}^i) &= \textstyle\sum_{\mathbf{x},\mathbf{u}}\eta_{\boldsymbol{\theta}^i}(\mathbf{x},\mathbf{u})\boldsymbol{\psi}_{\boldsymbol{\theta}^i}(\mathbf{x},\mathbf{u})\boldsymbol{\phi}'_{\boldsymbol{\theta}^i}(\mathbf{x},\mathbf{u}).\end{aligned}$$

Let $\bar{\mathbf{r}}^i(\boldsymbol{\theta}^i)$ be the limit of the critic parameter $\mathbf{r}^i_k$ if the policy parameter $\boldsymbol{\theta}^i$ was held fixed. The critic part of the algorithm is identical with the single agent version in [1], hence this limit exists. The actor update can be written as follows:

$$\begin{aligned}\boldsymbol{\theta}^i_{k+1} =& \mathbf{A}^{ii}_k\boldsymbol{\theta}^i_k + \textstyle\sum_{j\neq i}\mathbf{A}^{ij}_k\boldsymbol{\theta}^j_{t^{ij}(k)}\\ &+ \beta^i_k\bar{\mathbf{H}}(\boldsymbol{\theta}^i_k)(\bar{\mathbf{r}}^i(\boldsymbol{\theta}^i_k)\Gamma(\bar{\mathbf{r}}^i(\boldsymbol{\theta}^i_k)))\\ &+ \beta^i_k(\mathbf{H}_{\boldsymbol{\theta}^i_k}(\mathbf{X}^i_{k+1},\mathbf{U}^i_{k+1}) - \bar{\mathbf{H}}(\boldsymbol{\theta}^i_k))(\mathbf{r}^i_k\Gamma(\mathbf{r}^i_k))\\ &+ \beta^i_k\bar{\mathbf{H}}(\boldsymbol{\theta}^i_k)(\mathbf{r}^i_k\Gamma(\mathbf{r}^i_k) - \bar{\mathbf{r}}^i(\boldsymbol{\theta}^i_k)\Gamma(\bar{\mathbf{r}}^i(\boldsymbol{\theta}^i_k))).\quad (12)\end{aligned}$$

Setting

$$\mathbf{f}^i(\boldsymbol{\theta}) = \bar{\mathbf{H}}(\boldsymbol{\theta})\bar{\mathbf{r}}^i(\boldsymbol{\theta}), \tag{13}$$

$$\mathbf{e}_k^{i,(1)} = (\mathbf{H}_{\boldsymbol{\theta}_k^i}(\mathbf{X}_{k+1}^i, \mathbf{U}_{k+1}^i) - \bar{\mathbf{H}}(\boldsymbol{\theta}_k^i))\mathbf{r}_k^i\Gamma(\mathbf{r}_k^i), \tag{14}$$

$$\mathbf{e}_k^{i,(2)} = \bar{\mathbf{H}}(\boldsymbol{\theta}_k^i)(\mathbf{r}_k^i\Gamma(\mathbf{r}_k^i) - \bar{\mathbf{r}}^i(\boldsymbol{\theta}_k^i)\Gamma(\bar{\mathbf{r}}^i(\boldsymbol{\theta}_k^i))), \tag{15}$$

$$\mathbf{e}_k^{i,(3)} = \mathbf{A}_k^{ii}\boldsymbol{\theta}_k^i + \textstyle\sum_{j\neq i}\mathbf{A}_k^{ij}\boldsymbol{\theta}_{t^{ij}(k)}^j - \boldsymbol{\theta}_k^i, \tag{16}$$

the actor update can be written as

$$\boldsymbol{\theta}_{k+1}^i = \boldsymbol{\theta}_k^i + \mathbf{e}_k^{i,(3)} + \beta_k^i\mathbf{e}_k^{i,(1)} + \beta_k^i\mathbf{e}_k^{i,(2)} + \beta_k^i\bar{\mathbf{H}}(\boldsymbol{\theta}_k^i)(\bar{\mathbf{r}}^i(\boldsymbol{\theta}_k^i)\Gamma(\bar{\mathbf{r}}^i(\boldsymbol{\theta}_k^i))). \tag{17}$$

Now, the update equation above and the error terms $\mathbf{e}_k^{i,(1)}$, and $\mathbf{e}_k^{i,(2)}$ can be handled exactly as in [1, Section 6]. Specifically, as in shown in [1, Section 6], $\mathbf{f}^i(\boldsymbol{\theta})$ approximates $\nabla\bar{\alpha}(\boldsymbol{\theta})$ with a maximum error that is independent of λ, and the error terms $\mathbf{e}_k^{i,(1)}$, and $\mathbf{e}_k^{i,(2)}$ converge w.p.1. What remains to be shown is that $\lim_k \mathbf{e}_k^{i,(3)} = \mathbf{0}$ w.p.1.

To that end, we will use the analysis in [10]. Notice that our actor update (11) has the same form as Eq. (2.1) in [10], i.e., the form of a consensus algorithm perturbed by the gradient which can be viewed as a "noise" term. In Eq. (2.1) of [10], we will take the stepsize $\gamma^i(n)$ to be equal to 1 and the gradient $s^i(n)$ to be equal to $\beta_k^i\mathbf{H}_{\boldsymbol{\theta}_k^i}(\mathbf{X}_{k+1}^i, \mathbf{U}_{k+1}^i)(\mathbf{r}_k^i\Gamma(\mathbf{r}_k^i))$. Let $\mathbf{y}_k$ be defined as in Eqs. (2.13) and (2.14) of [10]. That is, in our setting, $\mathbf{y}_k$ is the value of $\boldsymbol{\theta}$ that all agents would agree to if at time k they switched to a new actor update iteration that has only the first two terms of (11), namely, a consensus algorithm. Define, also,

$$b_k = \textstyle\sum_{i=1}^N \beta_k^i||\mathbf{H}_{\boldsymbol{\theta}_k^i}(\mathbf{X}_{k+1}^i, \mathbf{U}_{k+1}^i)(\mathbf{r}_k^i\Gamma(\mathbf{r}_k^i))||. \tag{18}$$

Following the same reasoning as in [10] (and using Assumption B) one can establish an inequality equivalent to (A.2) in [10], namely, there exists a $d \in [0, 1)$ such that

$$||\mathbf{y}_k - \boldsymbol{\theta}_k^i|| \leq A\textstyle\sum_{n=1}^k d^{k-n}b_n, \tag{19}$$

where A is some constant. Let now

$$\tilde{\mathbf{H}}_k = (\beta_k^1||\mathbf{H}_{\boldsymbol{\theta}_k^1}(\mathbf{X}_{k+1}^1, \mathbf{U}_{k+1}^1)(\mathbf{r}_k^1\Gamma(\mathbf{r}_k^1))||, \ldots, \beta_k^N||\mathbf{H}_{\boldsymbol{\theta}_k^N}(\mathbf{X}_{k+1}^N, \mathbf{U}_{k+1}^N)(\mathbf{r}_k^N\Gamma(\mathbf{r}_k^N))||).$$

We have

$$\begin{aligned}
&\mathbf{E}[\textstyle\sum_k(b_k)^2] = \mathbf{E}[\textstyle\sum_k(\mathbf{e}'\tilde{\mathbf{H}}_k)^2]\\
&\leq ||\mathbf{e}||^2\textstyle\sum_i\sum_k(\beta_k^i)^2\mathbf{E}[||\mathbf{H}_{\boldsymbol{\theta}_k^i}(\mathbf{X}_{k+1}^i, \mathbf{U}_{k+1}^i)(\mathbf{r}_k^i\Gamma(\mathbf{r}_k^i))||^2]\\
&\leq NC\textstyle\sum_i\sum_k(\beta_k^i)^2\mathbf{E}[||\mathbf{H}_{\boldsymbol{\theta}_k^i}(\mathbf{X}_{k+1}^i, \mathbf{U}_{k+1}^i)||^2]\\
&\leq NC'\textstyle\sum_i\sum_k(\beta_k^i)^2 < \infty.
\end{aligned}$$

The 2nd inequality above holds for some $C > 0$ and is due to Assumption C. For some constant $C' > 0$, the 3rd inequality above follows from [1, Lemma 4.3] which states that $\mathbf{E}[||\mathbf{H}_{\boldsymbol{\theta}_k^i}(\mathbf{X}_{k+1}^i, \mathbf{U}_{k+1}^i)||^2]$ is bounded. The finiteness of $\mathbf{E}[\sum_k(b_k)^2]$ follows from Assumption C(a). We conclude that b_k converges to zero almost surely. The convergence of b_k establishes (due to (19)) that $\mathbf{y}_k - \boldsymbol{\theta}_k^i$ converges to zero for all i almost surely which in turn implies the convergence of $\mathbf{e}_k^{i,(3)}$ to zero. This completes the proof as the convergence of $\{\boldsymbol{\theta}_k^i\}$ follows from [1, Thm. 6.3]. ■

IV. A MOBILE SENSOR NETWORK COORDINATION PROBLEM

As an application we next consider a fairly general class of coordination problems arising in sensor networks with mobile nodes. Assume a 2-dimensional *mission space*, $\mathscr{S} \subset \mathbb{R}^2$, in which there is a set of M *target points* indexed by $i = 1, \ldots, M$ whose positions are indicated, at time k, by $\mathbf{m}_k^i \in \mathscr{S}$. These positions may change over time; we assume that $\{\mathbf{m}_k^i;\ k = 1, \ldots, \}$ is a stationary stochastic process for each i. To each *target point* i we associate a *reward* $R_k^i \in \mathbb{R}_+$.

The mission space is to be explored by N mobile sensor nodes (agents) indexed by $j = 1, \ldots, N$, whose positions at time k are indicated by $\mathbf{x}_k^j \in \mathscr{S}$. To each node j we associate a capacity $C_k^j \in \mathbb{R}_+$. When a node "visits" a target point it collects a reward which depends on the available reward at the target point and the capacity of the node. Every visit has also the effect of depleting a part of the node's capacity. We assume that nodes start their exploration of the mission space from the origin $\mathbf{0} \in \mathscr{S}$. Every node j navigates in $\mathscr{S}$ and, from time to time, returns to the origin which has the effect of replenishing the node's capacity to its initial value C_0^j.

In particular, the dynamics of the sequences $\{R_k^i\}$ and $\{C_k^j\}$ are described as follows. For all $i = 1, \ldots, M$, *(i)* $R_{k+1}^i = \max(R_k^i - C_k^j, 0)$, if $\exists j \in 1, \ldots, N$ such that $\mathbf{m}_k^i = \mathbf{x}_k^j$, and *(ii)* $R_{k+1}^i = R_k^i + w_k$, otherwise, where $\{w_k\}$ is a sequence of i.i.d. random variables. For all $j = 1, \ldots, N$, *(i)* $C_{k+1}^j = \max(C_k^j - R_k^i, 0)$, if $\exists i \in 1, \ldots, M$ such that $\mathbf{m}_k^i = \mathbf{x}_k^j$, *(ii)* $C_{k+1}^j = C_0^j$, if $\mathbf{x}_k^j = \mathbf{0}$, and *(iii)* $C_{k+1}^j = C_k^j + g_k$, otherwise, where $\{g_k\}$ is also a sequence of i.i.d. random variables. The sequence of rewards, Φ_k^j, collected by each node j over time is characterized as $\Phi_k^j = \min(R_k^i, C_k^j)$, if $\exists i \in 1, \ldots, M$ such that $\mathbf{m}_k^i = \mathbf{x}_k^j$, and 0 otherwise.

The scenario we are considering is completed by the sensing capabilities of the nodes. The nodes can *sense* the reward located at the *target points* depending on the "intensity" of the target and their distance from it. We assume that nodes can identify from these signals (or obtain otherwise) the number of targets present in the mission space and can pick up a "signal" from each target. Specifically, for all $i = 1, \ldots, M$

$$s_k^i(\mathbf{y}) = \frac{R_k^i}{2\pi\det(\boldsymbol{\Sigma}^i)^{1/2}}e^{-(\mathbf{y}-\mathbf{m}_k^i)'(\boldsymbol{\Sigma}^i)^{-1}(\mathbf{y}-\mathbf{m}_k^i)/2} \tag{20}$$

is the signal associated with the *target point* i and measured, at time k, at the position $\mathbf{y} \in \mathscr{S}$ of the *mission space*. In (20) $\boldsymbol{\Sigma}^i$ is a positive definite weight matrix associated with the *target point* i and $\det(\boldsymbol{\Sigma}^i)$ denotes its determinant.

Given this setup we are interested in a policy that guides the nodes in the mission space so that we maximize the long-term average total reward collected given by

$$\lim_{k\to\infty}\frac{1}{k}\textstyle\sum_{\tau=1}^k\sum_{j=1}^N\Phi_\tau^j. \tag{21}$$

A. A Parametric class of policies

We will first discretize the mission space $\mathscr{S}$ by superimposing a 2-dimensional grid. We assume that the positions of the target points $\mathbf{m}_k^i$, for all k and i, are always on this grid and all nodes move on the grid.

We are interested in a policy for each node that is based on its current position, the signals it measures from all target points, and, potentially, any information it receives from other nodes. At time k the jth node can choose a control action $\mathbf{u}_k^j \in \mathscr{U} = \{(1,0),(-1,0),(0,0),(0,1),(0,-1)\}$ to move from the position $\mathbf{x}_k^j$ to the position $\mathbf{x}_{k+1}^j = \mathbf{x}_k^j + \mathbf{u}_k^j$. We adopt the convention that if $\mathbf{x}_k^j$ is on the border of the grid then the control set $\mathscr{U}$ contains only the feasible control actions. We consider the following class of RSPs where each node j at time k and position $\mathbf{x}_k^j$ selects control $\mathbf{u} \in \mathscr{U}$ with probability

$$\mu_{\boldsymbol{\theta}^j}(\mathbf{u}|\mathbf{x}_k^j) = \frac{\exp(\xi_{\boldsymbol{\theta}^j}(\mathbf{u},\mathbf{x}_k^j))}{\sum_{\mathbf{v}\in\mathscr{U}}\exp(\xi_{\boldsymbol{\theta}^j}(\mathbf{v},\mathbf{x}_k^j))}, \tag{22}$$

where

$$\xi_{\boldsymbol{\theta}^j}(\mathbf{u},\mathbf{x}) = \sum_{i=1}^M \theta_i^j C_k^j s_k^i(\mathbf{x}+\mathbf{u}) + \theta_0^j e^{-||\mathbf{x}+\mathbf{u}||}, \tag{23}$$

and where $\boldsymbol{\theta}^j = (\theta_0^j,\ldots,\theta_M^j)$. The vector $\boldsymbol{\theta}^j$ parametrizes node's j policy. Notice that the structure of the policy favors control actions that lead to targets emitting stronger signals. When the node's capacity or the available rewards are low then we favor control actions that tend to bring the node closer to the origin so that it can replenish its capacity. It is important to notice that the class of RSPs introduced automatically satisfies Assumptions A(a)-(b)-(c) and for A(d) we can take $\mathbf{x}^* = \mathbf{0}$.

We remark that (22) represents one potential structure of the RSP, which we find appealing due to the minimal a priori information required. If additional information, for instance the approximate location of the targets points, is available then it can be easily incorporated. Furthermore, the mission space can be readily rendered more complex and include inaccessible regions and obstacles that restrict the agents movement and communications. To that end, the agents need local "visibility" to restrict their control actions to only the feasible ones.

B. Numerical Results

The D-AC algorithm runs as follows:

1) *Initialization phase:* for each node i, $i = 1,\ldots,N$ assign, randomly, an initial position $\mathbf{X}_0^i$ and the initial values for the parameters $\boldsymbol{\theta}_0^i, \mathbf{r}_0^i, \mathbf{Z}_0^i$.
2) Each node i chooses the control actions $\mathbf{U}_k^i$ according to the RSP $\boldsymbol{\theta}_k^i$.
3) Each node i receives the value of the parameters $\boldsymbol{\theta}_k^j$, $j \in \Omega_k^i = \{j = 1,\ldots,N|\ ||\mathbf{X}_k^i - \mathbf{X}_k^j|| \le \delta_k\}$ from all other nodes within a δ_k range.
4) Each node updates its own parameters according to the D-AC algorithm (cf. (7)–(11)) and using the information received from other nodes.
5) iterate from 2).

The parameter δ_k can be interpreted as the maximum communication radius and as we will see it greatly affects the performance and convergence rate of the algorithm. In one extreme, when $\delta_k \ll 1$ for all k, there is no communication between the mobile nodes and every node essentially runs the single-agent version of the actor-critic method based on just local observations. In the other extreme, when $\delta_k \gg 1$ for all k, all nodes communicate and information from any part of the mission space directly affects decisions in every other part.

We consider a 20×20 grid with three *target points* as the *mission space* $\mathscr{S}$. The *target points* are at $(15,5)$, $(5,15)$ and $(15,15)$, and their initial rewards are 500, 100 and 200, respectively (see Fig. 1 (Left)). In this example we assume that rewards do not vary over time (i.e., $w_k = 0$ for all k) and only decrease due to the action of the nodes. We have 16 nodes in our disposal and we assume that $g_k = 0$ for all k.

To evaluate the D-AC algorithm, we take as performance indices the *amount of reward collected in a fixed number of iterations* and the *number of iterations for all node parameter vectors to converge*. For comparison purposes, we run the algorithm in two different settings. In one setting, we have one single team of nodes exploring the mission space with the only communication constraint imposed by the maximum communication radius δ_k. In the other setting, we divided the nodes into four different teams of four. The policy structure of every team has an additional attraction term (cf. (23)) that favors the exploration of one particular area of the mission space (distinct for each team). Namely, the teams are configured to specialize in a part of the mission space. In this multi-team setting the nodes can exchange information only with nodes of the same team.

Table I reports statistics (mean, variance, minimum, and maximum) for the number of iterations needed for convergence, in both the single and the multiple-team case. Results are provided for different values of the parameter δ based on 50 simulation runs for each combination of the parameters. We use $\delta = 0.5$ as a "proxy" for the centralized version of the algorithm. In particular, nodes do not communicate and each one runs the centralized algorithm but they do explore in parallel the same mission space. In this case, we compute the number of iterations each node needs to converge to a $\boldsymbol{\theta}^*$ and average over the number of nodes and the simulation runs. For $\delta > 1$ the nodes do communicate and run the D-AC algorithm; we compute the number of iterations needed for all nodes to converge to the consensus $\boldsymbol{\theta}^*$ and average over the 50 simulation runs. Table II reports the same statistics (mean, variance, minimum, and maximum) for the amount of reward collected in 1000 iterations by all nodes, again in the single and multiple-team scenarios. Results are again averaged over 50 simulation runs for each case.

The results point to an intuitive trade-off between the two performance indices. Increasing the communication radius leads to faster convergence as information from each node reaches faster all other nodes. Note, also, that no communication leads to faster convergence than limited communication. However, a larger communication radius decreases the reward collected. Our understanding is that the nodes converge to a policy that *"averages"* the views of all nodes on what constitutes a good reward-collecting policy. These views are formed from the exploration of different parts of the state-control space and the collective view formed leads to sub-optimal decisions for most of the nodes as they navigate in the mission space, resulting in a slower reward collection rate.

The results in Tables I and II also indicate that the above negative effect of a large communication radius can be reduced or eliminated by the introduction of multiple teams exploring the mission space. Using multiple teams, and favoring the

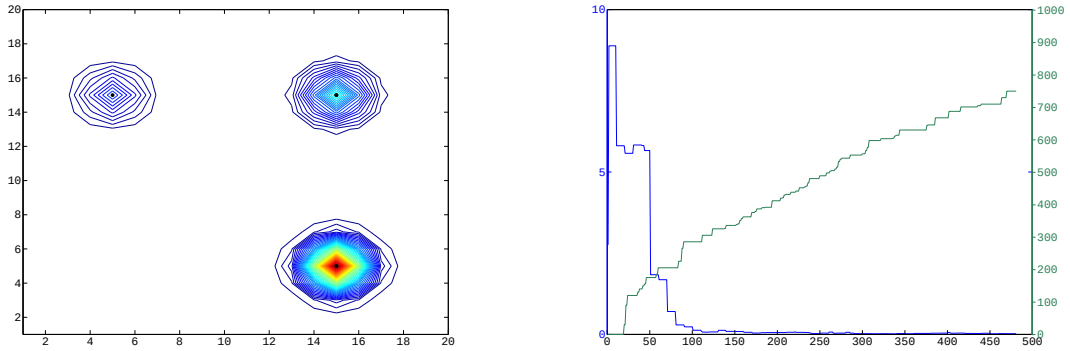

Fig. 1. (Left:) The initial mission space. Each target is indicated by a dot and the contours depict the landscape of the emitted signals, where red colors indicate a stronger signal. (Right:) The green line taking values on the right vertical axis represents total reward collected by a single team of 16 nodes over time (horizontal axis) when $\delta_k = 5$ for all k. The blue line converging to zero plots $\max_{i,j} ||\boldsymbol{\theta}_k^i - \boldsymbol{\theta}_k^j||$ over k and takes values in the left vertical axis.

TABLE I
NUMBER OF ITERATIONS NEEDED FOR CONVERGENCE OF THE NODES' PARAMETER VECTORS.

	Single Team				Multiple Teams			
δ	Average	Variance	Best	Worst	Average	Variance	Best	Worst
0.5	150.41	292.12	18.23	>1000	155.26	275.87	22.56	>1000
2	595.18	428.72	21.00	>1000	18.34	6.85	11.00	34.00
5	121.90	265.51	11	>1000	11.80	2.74	11.00	21.00
10	11.00	0.00	11.00	11.00	11.60	2.40	11.00	21.00

TABLE II
AMOUNT OF REWARD COLLECTED IN 1000 ITERATIONS.

	Single Team				Multiple Teams			
δ	Average	Variance	Best	Worst	Average	Variance	Best	Worst
0.5	765.04	6.13	777.53	760.03	768.14	9.15	781.15	758.18
2	738.99	115.42	783.30	90.00	786.43	20.98	839.07	761.58
5	732.75	89.45	779.05	384.04	783.26	16.41	821.51	760.48
10	693.33	183.56	772.30	346.34	783.63	22.31	838.78	761.45

exploration of a particular area of the mission space by each team, it is possible to increase the cohesion (both in the physical space and in the parameters space) of the nodes belonging to the same team. This allows teams to "specialize" and act more effectively in their designated part of the mission space. In addition, the relatively small number of nodes in each team leads to faster parameter convergence. Note that the multi-team scenario with a communication radius in the 5–10 range yields much faster convergence than the centralized version ($\delta = 0.5$) and dominates in terms of reward collected.

This leads to the observation that there exists an optimal relationship between the design variables of the algorithm: the number of agents, the number of teams and the communication radius. These parameters can be tuned by the designer to fit the specific problem and application.

We believe that a key advantage of the algorithm we proposed is its capability to "learn" and adapt to different environments. The algorithm requires minimal a priori knowledge (the number of targets) and all reward information is learned based on the signals received by the nodes. The agents reach consensus on $\boldsymbol{\theta}$ as long as the environment is "stationary", and then, reacting to a change in the environment and to the arrival of new observations, collectively update the value of their $\boldsymbol{\theta}$'s.

V. CONCLUSIONS

We developed a distributed algorithm that guarantees consensus and convergence to an optimal parametric control policy, in the context of a multi-agent coordination problem. The numerical results we report reveal an interesting trade-off between performance and the speed of convergence. Specifically, a large communication radius leads to faster convergence but leads to suboptimal actions. The effect of a large communication radius on the reward collection time can be reduced or eliminated by dividing the agents into multiple teams that are designated to explore different parts of the state-control space. The proposed algorithm provides a framework for distributively solving Markov decision problems by teams of agents and can have far greater applicability than the particular application we presented.

REFERENCES

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

[2] J. N. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Department of EECS, MIT, 1984.

[3] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 2953–2958, 2003.

[4] V. Blondel, J. Hendrickx, A. Olshevsky, and J. Tsitsiklis, "Convergence in Multiagent Coordination, Consensus, and Flocking," in *44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC'05)*, Seville, Spain, 2005, pp. 2996–3000.

[5] W. Li and C. G. Cassandras, "Distributed Cooperative Coverage Control of Sensor Networks," in *Proc. of 44th IEEE Conf. on Decision and Control*, 2005, pp. 2542 – 2547.

[6] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[7] P. Pennesi and I. C. Paschalidis, "Solving sensor network coverage problems by distributed asynchronous actor-critic methods," in *Proceedings of the 46th IEEE Conference on Decision and Control*, New Orleans, Louisiana, December 2007, pp. 5300–5305.

[8] P. Marbach and J. Tsitsiklis, "Simulation-based optimization of Markov reward processes," *IEEE Trans. Automat. Contr.*, vol. 46, no. 2, pp. 191–209, 2001.

[9] J. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Trans. Automat. Contr.*, vol. 42, no. 5, pp. 674–690, 1997.

[10] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Automat. Contr.*, vol. 31, no. 9, pp. 803–812, 1986.