ECAI 2008
M. Ghallab et al. (Eds.)
IOS Press, 2008

doi:10.3233/978-1-58603-891-5-157

Exploiting locality of interactions using a policy-gradient approach in multiagent learning

Francisco S. Melo[1]

Abstract.

In this paper, we propose a policy gradient reinforcement learning algorithm to address transition-independent Dec-POMDPs. This approach aims at implicitly exploiting the locality of interaction observed in many practical problems. Our algorithms can be described by an actor-critic architecture: the actor component combines *natural gradient updates* with a *varying learning rate*; the critic uses only local information to maintain a belief over the joint state-space, and evaluates the current policy as a function of this belief using compatible function approximation. In order to speed the convergence of the algorithm, we use an *optimistic initialization* of the policy that relies on a fully observable, single agent model of the problem. We illustrate our approach in some simple application problems.

1 INTRODUCTION

One of the main topics of research in artificial intelligence is the development of autonomous intelligent agents. The ability of an agent to fulfill a certain task in a given environment greatly depends on that agent's ability to *perceive* its environment and *interact* with it. As new and demanding applications appear, there is a natural interest in developing more complex intelligent agents, able to interact not only with the environment but with *other agents* existing in the same environment. In such multiagent applications, it is desirable that a each agent be able to *adapt* and *coordinate* with the other agents.

Reinforcement learning (RL) provides an appealing approach to address such adaptability issues. The "traditional" RL approach to multiagent systems makes use of game theoretic models such as Markov games [8]. This approach has fostered many interesting algorithms in which independent decision-makers are able to successfully adapt their policy to that of other agents and coordinate towards a common goal (*e.g.*, [4, 17]).

However, the Markov game approach is generally unsuited to address problems in which the agents have sensorial limitations, since they rely on several joint-observability assumptions inherent to these models that seldom hold in practice. In fact, many problems found in practice require models that can accomodate for some form of *partial observability*, such as partially observable stochastic games [7] or decentralized-MDPs/POMDPs [2]. Unfortunately, such models are inherently too complex to be solved exactly [2]. It is in face of this inherent complexity of multiagent problems that policy gradient methods may prove of use [4, 14].

In this work, we address multiagent decision problems using a policy-gradient approach. In particular, we adopt an actor-critic architecture similar to that considered in several single-agent works [6, 12] and extend this to multi-agent problems with partial observability. We consider cooperative multiagent tasks in which each agent has only local imperfect perception of its state and cannot observe the actions of other agents. We use a decentralized POMDP (Dec-POMDP) to model the group of agents, but consider several assumptions that aim at exploring the *locality of interaction* present in many problems in practice. Namely, we admit the state transitions of each agent to depend only on *its own actions*, and the observations of each agent to depend only on the *joint state of the agents*.

We apply our actor-critic algorithm to this transition-independent Dec-POMDP. Our critic-component uses *TD-learning with function approximation* to evaluate the policy currently implemented by the actor; the actor, in turn, uses this evaluation to update the policy using an estimate of the *natural gradient* [5] and a win-or-learn-fast (WoLF) update schedule [4]. The setting considered in this work is distinct from other approaches in the literature in that we assume no joint-state or joint-action observability. Our method is, to the extent of our knowledge, *the first learning algorithm* for Dec-POMDPs.

The paper is organized as follows. In Section 2 we review the basic models used in the paper, such as MDPs, POMDPs and Dec-POMDPs. We proceed in Section 3 by introducing our actor-critic algorithm for transition independent Dec-POMDPs. In Section 4 we illustrate the application of our method in simple multi-robot navigation problems and conclude in Section 5 by discussing how this approach can be extended to more complex problems.

2 MARKOV MODELS

In this section we review the basic models used throughout the paper. We start by reviewing Markov decision processes (MDPs) and their partially observable counterparts (POMDPs). We then move to multiagent models such as Markov games and their cooperative, partially observable counterparts, Dec-POMDPs.

2.1 Markov decision processes

A *Markov decision process* (MDP) is a tuple $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ where $\mathcal{X}$ is a finite set of possible states and $\mathcal{A}$ is a finite set of possible actions. $\mathsf{P}_a(x, y)$ represents the probability of moving from state $x \in \mathcal{X}$ to state $y \in \mathcal{X}$ by choosing a particular action $a \in \mathcal{A}$. The function $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is a bounded reward function, assigning the agent a numerical reward $r(x, a)$ for choosing action a in state x. The purpose of the agent is to maximize the expected total sum of discounted rewards, where $0 \leq \gamma < 1$ is a discount-factor assigning greater importance to rewards coming earlier in the future and X_t and A_t denote the state and action at time t. The optimal value function V^* is defined for each state $x \in \mathcal{X}$ as

$$V^*(x) = \max_{\{A_t\}} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(X_t, A_t) \mid X_0 = x\right] \qquad (1)$$

[1] School of Computer Science, Carnegie Mellon University, USA. E-mail: fmelo@cs.cmu.edu

and verifies the well-known Bellman optimality equation. The optimal Q-values $Q^*(x,a)$ are defined for each pair (x,a) as

$$Q^*(x,a) = r(x,a) + \gamma \sum_{y \in \mathcal{X}} \mathsf{P}_a(x,y) V^*(y) \tag{2}$$

The optimal decision rule can be obtained from Q^* as $\pi^*(x) = \arg\max_{a \in \mathcal{A}} Q^*(x,a)$ and the map π^* is the *optimal policy* for the MDP.

More generally, a *policy* is any mapping π_t defined over $\mathcal{X} \times \mathcal{A}$ that generates a control process $\{A_t\}$ verifying $\mathbb{P}[A_t = a \mid X_t = x] = \pi_t(x,a)$. We write $V^{\pi_t}(x)$ instead of $V(\{A_t\}, x)$ if the control process $\{A_t\}$ is generated by a policy π_t. A *stationary policy* is a policy π that does not depend on t. A *deterministic policy* is a policy assigning probability 1 to a single action in each state.

2.2 Partially observable MDPs

We refer to a *partially observable Markov decision process*, or POMDP, as a tuple $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathcal{Z}, \mathsf{P}, \mathsf{O}, r, \gamma)$, where $\mathcal{X}$, $\mathcal{A}$, P, r and γ are as defined before. The fundamental difference between POMDPs and MDPs is that, in the former, the agent is no longer able to decide based on the state X_t at time instant t, since this state not observable. Instead, the agent has access to an observation Z_t that depends on X_t according to the observation probabilities[2]

$$\mathsf{O}(x,z) = \mathbb{P}[Z_t = z \mid X_t = x].$$

A common approach to address POMDPs is to maintain a belief on the current state of the process. The belief at time t, which we denote by b_t, is a vector representing the probability of being in each state $x \in \mathcal{X}$ given the history at time t, *i.e.*, $b_t(x) = \mathbb{P}[X_t = x \mid \mathcal{H}_t]$. This belief-vector summarizes all information so far and, for the purpose of decision-making, is a sufficient statistic for the history of the process. In fact, as in MDPs, it is possible to define for each policy π a value function

$$V^\pi(b) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \hat{r}(B_t, A_t) \mid X_0 \sim b \right],$$

where $X_0 \sim b$ indicates that X_0 is distributed according to the belief vector b and B_t denotes the (random) belief vector at time t. The function $\hat{r}$ is defined from r simply as:

$$\hat{r}(b,a) = \sum_{x \in \mathcal{X}} b(x) r(x,a).$$

Repeating the development in the previous subsection, we obtain similar definitions for Q^π and also for V^* and Q^*, now in terms of beliefs. In fact, a POMDP can be redefined in terms of beliefs as an MDP with continuous state-space (the belief-space). We conclude by remarking that, due to the computational complexity of exact POMDP methods [11], most methods used in practice to solve POMDPs rely on some form of approximation.

2.3 Markov games

A *Markov game* is a tuple $\mathcal{M} = \big(n, \mathcal{X}, (\mathcal{A}^k), \mathsf{P}, (r^k), \gamma\big)$, where n is the number of agents, $\mathcal{X}$ is the state-space, $\mathcal{A} = \times_{k=1}^n \mathcal{A}^k$ is the set of *joint actions*, P represents the controlled transition probabilities and r^k is the reward function for agent k, $k = 1, \ldots, n$. As in (PO)MDPs, in a Markov game each agent tries to maximize its individual expected total discounted reward.

[2] In this paper we ignore the dependence of the observations on the actions.

In this paper we focus only on cooperative settings. In cooperative settings, where all agents share the same reward (*i.e.*, $r^1 = \ldots = r^n$), there are *deterministic joint policies* that maximize the total expected reward *for all agents*. We henceforth refer to such (joint) policies as the *optimal policies*. Also, for this class of games, the definitions of value-function, Q-function and policy carry without great modification from those of MDPs, bearing in mind that, in Markov games, the action-choices depend on n independent agents.

2.4 Dec-POMDPs

We consider a *Dec-POMDP* as being described by a tuple $\mathcal{M} = \big(n, \mathcal{X}, (\mathcal{A}^k), (\mathcal{Z}^k), \mathsf{P}, (\mathsf{O}^k), r, \gamma\big)$, where n, $\mathcal{X}$, $\mathcal{A}$, P, r and γ are defined as in a Markov game,[3] $\mathcal{Z}^k$ is the set of possible observations for agent k and O^k describes the observation probabilities for agent k. At each time t, each agent k, $k = 1, \ldots, n$, takes an action $a^k \in \mathcal{A}^k$ and receives an observation Z^k according to the probabilities

$$\mathsf{O}^k(x, z^k) = \mathbb{P}\left[Z_t^k = z^k \mid X_t = x \right].$$

As in all previous frameworks, the purpose of each agent is to maximize the expected total discounted reward.

In this paper, we consider *transition independent* Dec-POMDPs. This means that the state-space $\mathcal{X}$ can be partitioned (factored) into individual state-spaces $\mathcal{X}^k$ verifying $\mathcal{X} = \times_{k=1}^n \mathcal{X}^k$. At each time step t, the state of the Dec-POMDP is thus a tuple $X_t = (X_t^1, \ldots, X_t^n)$. Each X_t^k describes the *state of agent k at time t*. Furthermore, this state depends only on the actions of agent k, *i.e.*,

$$\mathbb{P}\left[X_{t+1}^k = y^k \mid X_t = x, A_t = a \right] = \\ = \mathbb{P}\left[X_{t+1}^k = y^k \mid X_t^k = x^k, A_t^k = a^k \right].$$[4]

This allows each agent k, $k = 1, \ldots, n$, to maintain at each time step t an *individual belief* b^k regarding its individual state, updated as

$$b_{t+1}^k(y^k) = \frac{\sum_{x^k} b_t^k(x^k) \mathsf{P}_{a^k}^k(x^k, y^k) \mathsf{O}_{a^k}(y^k, z^k)}{\sum_{x^k, w^k} b_t^k(x^k) \mathsf{P}_{a^k}^k(x^k, w^k) \mathsf{O}_{a^k}^k(w^k, z^k)}, \tag{3}$$

where a^k is the *individual action* taken at time t and z^k was the individual observation received at time $t+1$.

3 POLICY GRADIENT APPROACH TO MULTIAGENT LEARNING

We now describe our learning algorithm for Dec-POMDPs. This algorithm can be seen as an extension of the algorithm in [3] to multiagent settings, using a WoLF policy update schedule and optimistic initialization.

3.1 The actor-critic architecture

Before going into the detailed description of our algorithm, we review some important concepts regarding policy-gradient/actor-critic algorithms. Further details can be found in [16, 6].

Let $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ be a MDP with a compact state-space $\mathcal{X} \subset \mathbb{R}^p$. Let π_θ be a stationary policy parameterized by some

[3] A Dec-POMDP describes, by definition, a cooperative group of agents. Therefore, all agents share the same reward r.

[4] This is often the case, for example, in multi-robot navigation tasks, where the moving actions of one robot do not affect the position of the other robots.

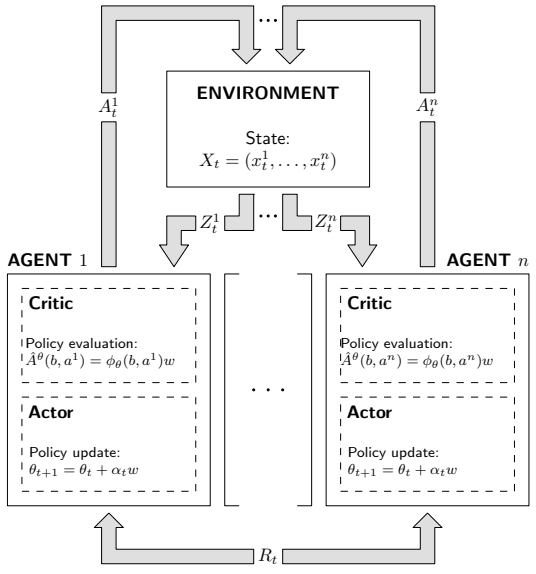


Figure 1. The actor-critic architecture. Each agent maintains at each time-step a belief b_t on the joint state of the process. The critic component estimates the Q-function associated with each local belief and each action. The actor component uses this evaluation to perform a policy update in the direction of the natural gradient, using a WoLF policy update schedule.

finite-dimensional vector $\theta \in \mathbb{R}^M$. We assume that π is continuously differentiable with respect to θ and henceforth write V^θ instead of V^{π_θ} to denote the corresponding value function. Define $\rho(\theta) = \int_{\mathcal{X}} V^\theta(x) p_0(x) dx$, where p_0 is the distribution for the initial state. Notice that we abusively write $\rho(\theta)$ instead of $\rho(\pi_\theta)$ to simplify the notation. The value $\rho(\theta)$ denotes the total expected discounted reward associated with policy π_θ given the initial state distribution p_0. As shown in [16, 6],

$$\nabla\rho(\theta) = \int_{\mathcal{X}} \sum_a p(x) \frac{\partial \pi_\theta}{\partial \theta}(x, a) Q^\theta(x, a) dx, \tag{4}$$

where ∇ denotes the gradient with respect to (w.r.t.) θ and

$$p(x) = \int_{\mathcal{X}} \sum_{t=0}^{\infty} \gamma^t \mathbb{P}\left[X_t = x \mid X_0 = y, \pi_\theta\right] p_0(y) dy. \tag{5}$$

We can now introduce the overall architecture of our actor-critic algorithm, as depicted in Figure 1. Each agent k follows an individual parameterized policy π_θ^k. At each time step t, agent k receives a local observation Z_t^k, used to update the belief b_t on the joint state of the process. The critic component of the architecture uses this belief and the history of individual actions and collected rewards to *evaluate* π_θ^k, by computing the associated Q-function (or, equivalently, the associated advantage function). This Q-function is then used in the actor-component to estimate the gradient $\nabla\rho(\theta)$ and update π_θ^k using the gradient direction.

In the following subsections we describe in greater detail the different components of the algorithm.

3.2 Exploiting local interaction

In many problems found in practice, the interaction/coordination between the several agents occurs only in very particular situations (*e.g.*, when sharing a resource or in avoiding undesirable states). Several recent works have proposed new models [1, 15] and methods [13, 10] that seek to take advantage of this locality of interaction in order to efficiently tackle the prohibitive complexity of general decentralized decision-making problems.

In our algorithm, local interactions are exploited at three different levels, to know:

Optimistic initialization Optimistic initialization consists in "ignoring" partial observability to initialize the parameterized policy. In particular, we compute the optimal Q-function for the fully observable Markov game initialize the policy as a soft-max version of Q-MDP. $Q^*(x, a)$ is a $|X| \times |A|$ matrix, that we consider as the *adjustable parameters for the policy*. Therefore, the general form of the policies considered is

$$\pi_\theta^k(b, a^k) = \frac{\sum_{a \ni a^k} e^{\sum_x b(x)\theta(x,a)}}{\sum_{u \in \mathcal{A}} e^{\sum_x b(x)\theta(x,u)}}, \tag{6}$$

where θ is a $|X| \times |A|$ real parameter matrix that is initialized to the values of Q^*. Notice that, in problems where the agents need only to interact in few, very particular situations, the policy described above can be implemented (most of the time) *independently of the other agents' states and policies*.

Independent belief tracking In independent belief tracking, *each agent* maintains an *individual belief estimate* for *every other agent*. Therefore, each agent k maintains at each time t a *vector of beliefs* $b_t = (b_t^1, \ldots, b_t^n)$, estimating the individual state of each agent. Since the agent has no knowledge of the actions of the other agents, the estimates regarding their individual state will often be very innacurate. However, in most situations, the action choice of agent k can be carried out *independently of the other agents* and, therefore, the innacuracy in the belief estimates b^j with $j \neq k$ does not affect the action-choice for agent k.

On the other hand, in those situation where interaction must occur, the innacuracy in other agents' belief estimates may have a negative impact on the performance of agent k (and, thus, of the group). Notice, however, that the observation model as described in Section 2 depends on the joint state of the process. It is possible to "minimize" the innacuracy in the belief estimates of other agents if, at those situations where interaction occurs, the observations for each agent k provide it information on the state of other agents that leads to a more accurate belief estimate.[5]

Optimistic policy estimation Finally, in conducting independent belief tracking, each agent k can estimate the policy followed by other agents, thus trying to include more information that may possibly yield more accurate belief estimates. As in the optimistic policy initialization, each agent k will estimate agent j's policy, $j \neq k$, as

$$\pi^k(b, a^k) = \arg\max_{a_k \in \mathcal{A}^k} \sum_{a \ni a^k} \sum_{x \in \mathcal{X}} b(x) Q^*(x, a),$$

where b is the joint belief estimate.

3.3 The actor: Combining natural gradients and WoLF updates

From (4) it is evident that, in order to compute the gradient $\nabla\rho(\theta)$, the function Q^θ needs to be computed. However, since our policy is defined in terms of beliefs (which are continuous quantities), so is Q^θ, and some form of function approximation is needed.

[5] Consider, for example, multirobot navigation tasks, where each robot can move, most of the time, relying only on its own position estimates. Only when two or more robots are close must they coordinate to avoid collisions. However, in these situations, sensorial information allows accurate estimates of the position of the other robots.

Let $\{\phi_i, i = 1, \ldots, M\}$ be a set of M linearly independent functions and $\mathcal{L}(\phi)$ its linear span. Let $\hat{Q}^\theta$ be the best approximation of Q^θ in $\mathcal{L}(\phi)$.[6] As any function in $\mathcal{L}(\phi)$, $\hat{Q}^\theta$ can be written as $\hat{Q}^\theta(x,a) = \phi^\top(x,a)w$. The following result can be found in [16].

Theorem 1 *Given an MDP $\mathcal{M} = (\mathcal{X}, \mathcal{A}, \mathsf{P}, r, \gamma)$ and a set of basis functions $\{\phi_i, i = 1, \ldots, M\}$ as defined above, if*

$$\phi(x,a) = \frac{\partial \log(\pi_\theta)}{\partial \theta}(x,a) \tag{7}$$

then

$$\nabla\rho(\theta) = \sum_{x,a} d(x)\frac{\partial \pi_\theta}{\partial \theta}(x,a)\hat{Q}^\theta(x,a).$$

Now, as observed in [5], the parameterized policy space can be seen as a manifold that can be endowed with an adequate Riemannian metric. From this metric, a *natural gradient* is defined, expressed in terms of the *Fisher information matrix*. Peters et al. [12] showed the natural gradient of $\rho(\theta)$ w.r.t. θ is, simply, $\tilde{\nabla}\rho(\theta) = w$, where w is the parameter vector corresponding to $\hat{Q}^\theta$. Bearing all this in mind, the update rule for our algorithm becomes:

$$\theta_{t+1} = \theta_t + \alpha_t w.$$

The step-size α_t is chosen according to the WoLF (win-or-learn fast) schedule: when "winning", a smaller learning rate is used; when "loosing", a larger learning rate is used. In other words, if the performance of the current policy is better than that of an "average" policy, a smaller learning rate is used (indicating that the policy may be close to a locally optimal policy). If, on the other hand, the performance of the current policy is worse than that of an average policy, the learning rate is set to a higher value, leading to a faster learning. In practical terms, we use a step-size sequence similar to that in [4]:

$$\alpha_t = \begin{cases} \alpha_w & \text{if } \sum_a \pi_\theta(b_t,a)\hat{Q}^\theta(b_t,a) > \sum_a \bar{\pi}(b_t,a)\hat{Q}^\theta(b_t,a) \\ \alpha_l & \text{otherwise} \end{cases}$$

where $\alpha_l > \alpha_w$ are, respectively, the loosing and winning learning rates and $\bar{\pi}$ is the "average" policy, obtained by using in (6) the average parameter vector over time.

3.4 The critic: Advantage estimation in belief space

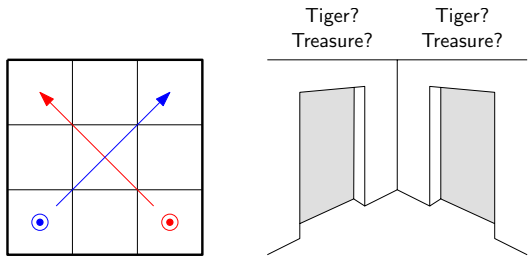
In the gradient expressions (4) and in Theorem 1, one can add an arbitrary function $F(x)$ to Q^θ and $\hat{Q}^\theta$. Such function is known as a *baseline function* and, as shown in [3], if F is to be chosen so as to minimize the mean-squared error between $\hat{Q}^\theta$ and Q^θ, the optimal choice of baseline function is $F(x) = V^\theta(x)$. Recalling that the *advantage function* associated with a policy π is defined as $A^\pi(x,a) = Q^\pi(x,a) - V^\pi(x,a)$, the performance of the overall algorithm can be improved by estimating the *advantage function* instead of the Q-function [3], .

As seen in the previous subsection, the actor component will update the parameter along the direction of the parameter vector w corresponding to the orthogonal projection of Q^θ (or, equivalently, A^θ) on the linear space spanned by the compatible basis functions, defined in (7). However, unlike Q^θ or V^θ, the advantage function does not verify a Bellman-like recursion and, therefore, it is necessary to independently estimate the value function V^θ. for which we also consider a linear approximation. In particular, we admit that $A^\theta(b,a) \approx \phi_\theta^\top(b,a)w$ and $V^\theta(b) \approx \xi^\top(b)v$, where ϕ_θ are the compatible basis functions defined according to (7) and each component ξ_i belongs to a second set of linearly independent basis functions that we use to approximate the value function.

Since we are considering multiagent problems, where multiple independent decision makers interact in a common environment, it is best that each agent k computes this estimate *online*, since the transition data sampled from the process reflects (although implicitly) the eventual learning process taking place in the other agents. Therefore, our critic uses a TD-based update to estimate *both* the value function V^θ and the advantage function A^θ by means of the following recursion (similar in spirit to that in [3])[7]

$$v_{t+1} = v_t + \beta_t\xi_t^\top\big[r_t + \gamma\xi_{t+1}v_t - \xi_t v_t\big];$$
$$w_{t+1} = (\mathbf{I} - \beta_t\phi_t^\top\phi_t)w_t + \beta_t\phi_t\big[r_t + \gamma\xi_{t+1}v_t - \xi_t v_t\big],$$

where $\mathbf{I}$ is the identity matrix, ξ_t is the row-vector $\xi^\top(b_t)$, $\xi_{t+1} = \xi^\top(b_{t+1})$ and $\phi_t = \phi_\theta^\top(b_t,a_t)$.



a) Grid world b) Dec-Tiger problem

Figure 2. Two simple problems used to illustrate the application of our algorithm.

4 EXPERIMENTAL RESULTS

To illustrate the working of our algorithm, we tested it in several very simple Dec-POMDPs scenarios.

The first set of results was obtained in a small grid-world problem, as represented in Figure 2.a. In this problem, each of two robots must reach the opposite corner in a 3×3 maze. When both agents reach the corresponding corners, they receive a common reward of 20. If they "collide" in some state, they receive a reward of -10. Otherwise, they receive a reward of -1. The robots can move in one of four directions, N, S, E and W. The transitions in each direction have some uncertainty associated: with probability 0.8 the movements succeeds and, with probability 0.2 it fails. The robots can observe "Null", indicating that nothing is detected; "Goal" indicating that the robot has reached its individual target position; and "Crash", indicating that both robots are in the same position. After successfully reaching the goal, the position of the robots is reset.

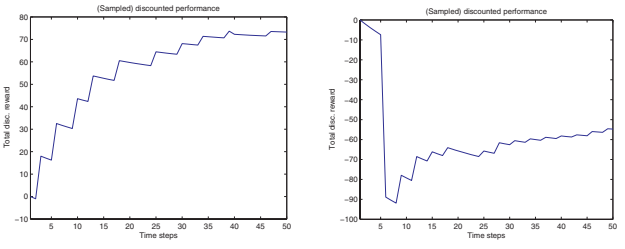
We ran the algorithm for 10^4 learning steps and then tested the learnt policy on the environment for 50 time-steps. In Figure 3.a, we present the total discounted reward obtained during a sample run. Notice that the robots are able to quickly reach the goal, which clearly indicates that they were able to learn the desired task. Notice also that the robots are able to avoid collisions, which indicates that they

[6] We take $\hat{Q}^\theta$ as the orthogonal projection of Q^θ on $\mathcal{L}(\phi)$ with respect to the inner product

$$\langle f, g\rangle = \int_{\mathcal{B}}\sum_a f(b,a)\cdot g(b,a)\pi^\theta(b,a)p(b)db,$$

where $\mathcal{B}$ denotes the belief-space and p is the distribution introduced in (5), with the beliefs b playing the role of x.

[7] We remark, however, that we are using a discounted framework, unlike the average per-step reward framework featured in [3].



a) Grid world b) Dec-Tiger

Figure 3. Sample runs with the learnt policies in the two test problems.

were able to *coordinate* without communicating and using only local information during learning.

The second problem is the well-known Dec-Tiger problem [9]. In this problem, two agents must choose between two doors, behind one of which is hidden a tiger. The other door hides a treasure. The purpose of the two agents is to figure out behind which door the treasure is hidden, by listening the noises behind the doors. They must act in a coordinated fashion *at all times*, since their performance greatly depends on this ability to coordinate.

We remark that this problem, unlike the grid-world problem, is not particularly suited to be addressed by our algorithm. In fact, the Dec-Tiger problem is not transition independent: the state-space cannot be factored and the actions of each agents have a large influence on both states, observations and rewards received by the other agent. Nevertheless, we applied our algorithm to this problem, to better understand the general applicability of the method.

Once again, we ran the algorithm for 10^4 learning steps and then tested the learnt policy on the environment for 50 time-steps. In Figure 3.b, we present the total discounted reward obtained during a sample run. Notice that, although some miscoordinations sometimes occur (which are impossible to overcome since each agent only has available local information), the agents are, nevertheless, able to attain many coordinated action choices. And, the remarkable thing is that, once again, this was achieved without communication and using only local information during learning (and execution).

Finally, to conclude this section, we summarize in Table 1 the average total discounted reward obtained during a 50-step run. The results presented correspond to the average over $2,000$ independent Monte-Carlo trials.

Environment	Total disc. reward
Grid world	34.001
Dec-Tiger	11.049

Table 1. Total discounted reward obtained in the two problems. The results correspond to the average over $2,000$ independent Monte Carlo runs.

5 CONCLUSIONS

We conclude the paper with several important remarks. First of all, the algorithm introduced here is closely related to the Gra-WoLF algorithm in [4]. The main differences lie on our usage of natural gradients and on our ability to address problems with partial state observability and no joint-action observability. Partial observability is addressed by considering the problem to be described by a transition independent Dec-POMDP. We take advantage of this fact by proposing several strategies that allow the agents to maintain independent beliefs that can be used for decision-making.

Another important observation is that the optimistic initialization considered will naturally bias the initial policy of the agents *towards the goal*. This bias may potentially lead to more frequent initial visits to the rewarding states and thus allowing the learning process to converge more rapidly.

Finally, it is important to remark that the results presented herein allow for little comprehension of the actual potential of the algorithm. We are currently testing this algorithm in much larger problems, which will allow us to infer how well our algorithm can cope with the high dimensionality arising from the consideration of large problems. We remark, however, that the fact that our algorithm does not take into account any *global* information, it is reasonable to expect that its complexity to grow linearly with the number of agents (instead of the exponential growth in fully coupled approaches).

It is also important to somehow compare the performance of our algorithm with that of the several planning methods in the literature, in the particular class of problems that can adequately be addressed by our algorithm. We remark, however, that these algorithms compute the policy off-line which difficults direct comparison.

ACKNOWLEDGEMENTS

This research was partially sponsored by the Portuguese Fundação para a Ciência e a Tecnologia under the Carnegie Mellon-Portugal Program and the Information and Communications Technologies Institute (ICTI), www.icti.cmu.edu. The views and conclusions contained in this document are those of the author only.

References

[1] R. Becker, S. Zilberstein, V. Lesser, and C. Goldman, 'Transition-independent decentralized Markov decision processes', in *Proc. AAMAS*, pp. 41–48, (2003).
[2] D. Bernstein, S. Zilberstein, and N. Immerman, 'The complexity of decentralized control of Markov decision processes', *Mathematics of Operations Research*, **27**(4), 819–840, (2002).
[3] S. Bhatnagar, R. Sutton, M. Ghavamzadeh, and M. Lee, 'Incremental natural actor-critic algorithms', in *Proc. NIPS 20*, pp. 105–112, (2007).
[4] M. Bowling and M. Veloso, 'Scalable learning in stochastic games', in *Workshop on Game & Decision Theor. Agents*, pp. 11–18, (2000).
[5] S. Kakade, 'A natural policy gradient', in *Proc. NIPS 14*, pp. 1531–1538, (2001).
[6] V. Konda and J. Tsitsiklis, 'On actor-critic algorithms', *SICON*, **42**(4), 1143–1166, (2003).
[7] H. Kuhn, 'Extensive games and the problem of information', *Annals of Mathematics Studies*, **28**, 193–216, (1953).
[8] M. Littman, 'Value-function reinforcement learning in Markov games', *J. Cognitive Systems Research*, **2**(1), 55–66, (2001).
[9] R. Nair, D. Pynadath, M. Yokoo, M. Tambe, and S. Marsella, 'Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings', in *Proc. IJCAI*, pp. 705–711, (2003).
[10] F. Oliehoek, M. Spaan, S. Whiteson, and N. Vlassis, 'Exploiting locality of interaction in factored Dec-POMDPs', in *Proc. AAMAS*, pp. 517–524, (2008).
[11] C. Papadimitriou and J. Tsitsiklis, 'The complexity of Markov chain decision processes', *Mathematics of Operations Research*, **12**(3), 441–450, (1987).
[12] J. Peters, S. Vijayakumar, and S. Schaal, 'Natural Actor-Critic', in *Proc. ECML*, pp. 280–291, (2005).
[13] M. Roth, R. Simmons, and M. Veloso, 'Exploiting factored representations for decentralized execution in multi-agent teams', in *Proc. AAMAS*, pp. 469–475, (2007).
[14] S. Singh, M. Kearns, and Y. Mansour, 'Nash convergence of gradient dynamics in general-sum games', in *Proc. UAI*, pp. 541–548, (2000).
[15] M. Spaan and F. Melo, 'Interaction-driven Markov games for decentralized multiagent planning under uncertainty', in *Proc. AAMAS*, pp. 525–532, (2008).
[16] R. Sutton, D. McAllester, S. Singh, and Y. Mansour, 'Policy gradient methods for reinforcement learning with function approximation', in *Proc. NIPS 13*, pp. 1057–1063, (2000).
[17] X. Wang and T. Sandholm, 'Reinforcement learning to play an optimal Nash equilibrium in team Markov games', in *Proc. NIPS 15*, pp. 1571–1578, (2002).