Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Bayesian Learning of Other Agents' Finite Controllers for Interactive POMDPs

Alessandro Panella and Piotr Gmytrasiewicz
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607

Abstract

We consider an autonomous agent operating in a stochastic, partially-observable, multiagent environment, that explicitly models the other agents as probabilistic deterministic finite-state controllers (PDFCs) in order to predict their actions. We assume that such models are not given to the agent, but instead must be learned from (possibly imperfect) observations of the other agents' behavior. The agent maintains a belief over the other agents' models, that is updated via Bayesian inference. To represent this belief we place a flexible stick-breaking distribution over PDFCs, that allows the posterior to concentrate around controllers whose size is not bounded and scales with the complexity of the observed data. Since this Bayesian inference task is not analytically tractable, we devise a Markov chain Monte Carlo algorithm to approximate the posterior distribution. The agent then embeds the result of this inference into its own decision making process using the interactive POMDP framework. We show that our learning algorithm can learn agent models that are behaviorally accurate for problems of varying complexity, and that the agent's performance increases as a result.

1 Introduction

An autonomous, rational agent operating in a stochastic, partially observable environment selects actions that maximize some objective function, usually the expected sum of future rewards, as in the case of partially observable Markov decision processes (POMDPs) (Kaelbling, Littman, and Cassandra 1998). In multiagent settings, an additional source of uncertainty is represented by the actions of other agents, that affect the state of the environment. In this paper, we consider an agent that maintains explicit models of other agents in order to predict their actions. The interactive POMDP (I-POMDP) framework (Gmytrasiewicz and Doshi 2005) proposes that the agent maintains a belief over *intentional* models of other agents, that recursively describe their I-POMDP structure, including nested beliefs and preferences. An alternative we pursue in this paper is to use *subintentional* models, intended as statistical predictive models of the other agents' behavior, that do not explicitly consider their preferences and beliefs. Specifically, we consider the class of probabilistic deterministic finite-state controllers (PDFCs), in which the transitions between nodes are deterministic and actions are generated stochastically in each node.

Copyright © 2016, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

In the remainder of this paper we assume that there are two agents: the protagonist agent i and the modeled agent j. We assume that agent i does not initially know the model of agent j. Instead, the agent maintains a probability distribution over all possible PDFCs of j. Upon receiving observations from the environment, this belief is updated via Bayesian inference to yield a posterior distribution that concentrates around likely models, to enable accurate prediction of j's actions. We note that agent i's observations do not, in general, expose j's behavior perfectly, but are instead only statistically correlated to j's actions, based on the effect that these have on the environment and how agent i perceives it. In this paper, we describe a two-phase approach: in the first, agent i gathers observations and infers a posterior distribution over j's models, using a batch algorithm; in the second, these models are embedded in i's own decision making process. This allows us to examine the properties of learning and planning in isolation. While we view an online approach that interleaves learning and planning as more realistic in many situations, we leave the exploration of this scenario for future work.

Of primary importance to our work is selecting a suitable prior distribution over PDFCs. Since the true size of agent j's controller is not known, we place a stick-breaking prior over the PDFC's nodes, that allows the posterior distribution to concentrate around PDFCs whose size scales with the complexity of the observed data. This eliminates the problem of having to choose a fixed model size. Since the Bayesian inference problem at hand is too complex to be amenable to conjugate analysis, we employ an ad-hoc MCMC algorithm to approximate the posterior distribution over the set of j's PDFCs. This generates a finite ensemble of possible models of j that are embedded in agent i's own decision making process by extending the state space as in I-POMDPs.

The choice to explicitly model the other agent is motivated by the assumption that agent i is aware of the other agent's presence, and that agent j implements an agent function that maps its own observation history onto actions. Moreover, agent i knows the environment's dynamics in response to the two agents' combined actions, and its own reward model. Without these assumptions, modeling the other

agent explicitly would be ineffective, whereas POMDP reinforcement learning techniques, such as utile suffix memory (Mccallum 1996), Bayes-adaptive POMDPs (Ross, Draa, and Pineau 2007), infinite generalized policy representation (Liu, Liao, and Carin 2011), infinite POMDPs (Doshi-Velez et al. 2013), and others would be more suitable to solve the problem, by implicitly treating the effect of the other agent's actions as noise and fold it into the environment's transition function. Clearly, learning the true model of another agent exactly with probability one is only possible with an infinite amount of observations, and in practice we may need a very large observation sequence to converge to the true model, as we see in our results. However, we show that our method is able to pick up regularities in the modeled agent's behavior even from limited observations, and the prior distribution naturally compensates for data scarcity in the Bayesian way. This allows the agent's performance to greatly improve even when the true model is not actually discovered.

In game theory, deterministic finite automata (DFA) have been employed to represent strategies with *bounded rationality* (Rubinstein 1998) and assuming the opponent's actions are observable. (Carmel and Markovitch 1996) provides a heuristic for the on-line inference of a consistent DFA, which does not guarantee any bound on the complexity of the learned model. However, we do not use here the classical game-theoretic solution concept of a Nash equilibrium, building on the growing body of work that uses the decision-theoretic solution concept (Oliehoek and Amato 2014) and behavioral game theory (Wright and Leyton-Brown 2012). Recent work (Conroy et al. 2015) describes a two-phase approach to learning and planning for interactive dynamic influence diagrams (Doshi, Zeng, and Chen 2009), assuming fully observable behavior and limited horizon.

The rest of this paper is organized as follows. Section 2 provides background on POMDPs and PDFCs, and Section 3 introduces the prior over PDFCs. We describe the learning algorithm in Section 4 and the planning framework in Section 5. Section 6 provides our results while Section 7 concludes the paper and provides hints for future work.

2 Background: POMDPs and PDFCs

A Partially observable Markov decision process (POMDP) (Kaelbling, Littman, and Cassandra 1998) is a general model for planning and acting in partially observable, stochastic domains. It is a tuple $P = (S, A, \Omega, T, O, R)$, where: S is the set of possible states of the world; A is the set of agent's actions; Ω is the set of observations the agent can receive; $T : S \times A \to \Delta(S)$ is the state transition function; $O : A \times S \to \Delta(\Omega)$ is the observation function; $R : S \times A \to \mathbb{R}$ is the reward function. Frequently, an optimal POMDP policy can be represented as a deterministic finite state controller. Some solution methods compute POMDP policies by performing a search directly in the space of FSCs (Hansen 1998), or search in the more general space of stochastic FSCs (Meuleau et al. 1999; Poupart and Boutilier 2003).

We use a version of FSCs called probabilistic deterministic finite controllers (PDFC) to model other agent(s) policies. In PDFCs the transitions between the memory states (nodes) are deterministic, but the actions are chosen stochastically in each node of the controller. The actions are stochastic in order to enable more efficient search through the space of all models. Formally, a PDFC is a tuple $c = (\Omega, A, Q, \tau, \theta, q_0)$, where: Ω is the set of observations of the agent; A is the set of actions the agent can execute; Q is the set of nodes, the internal states of the agent; $\tau : Q \times A \times \Omega \to Q$ is the deterministic node transition function; $\theta : Q \to \Delta(A)$ is the probabilistic emission function; q_0 is the initial node.

3 A Prior for PDFCs

In this paper, we propose a Bayesian methodology to learn over the class of possible PDFCs C_j, given an observed trajectory $h_{1:T}$ that provides information about agent j's behavior. We want to compute the posterior distribution $p(c_j|h_{1:T}) \propto p(h_{1:T}|c_j)\ p(c_j)$. Crucial to this task is providing a suitable prior distribution $p(c_j)$ over PDFCs; we do not wish to bound, a priori, the number of nodes of j's PDFC. Since each node k in a PDFC of unbounded size is associated to a real-valued parameter θ_k, C_j is an infinite-dimensional sample space. Bayesian nonparametric (BNP) techniques have increasingly been used in recent years to design priors over infinite-dimensional spaces, such as the case of Dirichlet process mixture models (DPMMs) and variations thereof (Hjort et al. 2010). More similarly to our problem, nonparametric priors have been proposed for probabilistic deterministic finite automata (PDFA) (Pfau, Bartlett, and Wood 2010) and Dec-POMDP controllers (Liu et al. 2015).

In the following, we describe the prior distribution that we adopt. For each starting node $k = 1..\infty$, action $j = 1..|A|$, and observation $h = 1..|\Omega_j|$, the destination node is drawn from a discrete infinite probability vector π, i.e. $\tau_{kjh} \sim \pi$. This vector is in turn distributed according to a stick-breaking process (Sethuraman 1994) with parameter α, which corresponds to repeatedly breaking the remaining part of a stick of initial length 1, each time drawing the breaking point from $\mathrm{Beta}(1, \alpha)$ scaled over such interval. Following the standard convention, we write this as $\pi \sim \mathrm{GEM}(\alpha)$. For each node $k = 1..\infty$, the corresponding emission distribution is drawn from a symmetric Dirichlet distribution $\theta_k \sim \mathrm{Dir}\left(\frac{\lambda}{|A|}, \ldots, \frac{\lambda}{|A|}\right)$.

Strictly speaking, this setup describes a distribution over PDFCs with infinite nodes, and not an unbounded finite number of nodes. However, we are interested only in the finite "connected component" containing of the nodes that are reachable from the initial node, ignoring the infinite subset of nodes that are not connected. It is useful to determine analytically the probability over the effective number of nodes K induced by our prior distribution. It can be shown that:

$$p(K|\alpha) = \frac{\alpha^K (KZ)!}{\alpha^{(KZ+1)}} \sum_{l=K-1}^{(K-1)Z} \frac{\bar{q}(K, l)}{l!}, \qquad (1)$$

where $\alpha^{(KZ+1)}$ indicates the rising factorial, $Z = |A||\Omega|$,

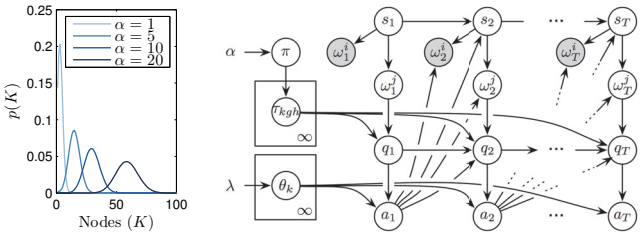

Figure 1: **Left:** Probability of number of nodes K for different values of α. **Right:** Bayesian model for PDFC learning.

and $\bar{q}(K, L)$ is given by the recurrence relation

$$\bar{q}(K, l+1) = l\ \bar{q}(K, l) + \begin{cases} \bar{q}(K-1, l) & \text{if } l < (K-2)Z \\ 0 & \text{o.w.} \end{cases}, \tag{2}$$

starting with $\bar{q}(K, K) = 1$. Figure 1-left shows the distribution of K for various values of α, assuming $|A| = 3$ and $|\Omega| = 2$.

This type of stick-breaking prior over PDFCs has some useful properties that simplify the inference process; in particular, the Chinese restaurant process (CRP) mechanism still holds and is used within the MCMC sampler described in the next section. Nonetheless, other similar priors can be considered, such as ones based on the Pitman-Yor process, which is a slight generalization of the one presented here, and other forms of two-parameters stick-breaking priors (Paisley and Carin 2009). Moreover, some classes of parametric prior distributions could be adapted to PDFCs, such as the hierarchical distribution described in (Green and Richardson 2001). Although these latter traditionally require more convoluted inference based on reversible jump Markov chain Monte Carlo methods, recent work (Miller and Harrison 2015) has shown promising results in deriving simpler inferential methods. We leave the exploration of such prior distributions for future work.

4 Inference

The Bayesian network in Figure 1-right depicts the learning scenario graphically. In general, we may not be able to observe the trajectory of observations and actions $(\omega^j_{1:T}, a_{1:T})$ of agent j, and instead just receive our own sequence of observations from the environment $\omega^i_{1:T}$. The sequences of world states $s_{1:T}$, actions $a_{1:T}$, and j's observations $\omega^j_{1:T}$ are related to the received sequence of observations via the world's transition and both agent's observation functions, which are here assumed to be known. We hence need to infer the posterior distribution $p(\tau, \theta|\omega^i_{1:T}) \propto p(\omega^i_{1:T}|\tau, \theta)\ p(\tau, \theta)$. If agent i is not a passive observer, its actions might also influence the environment's state. They are not considered here so not to clutter the presentation, but their integration is straightforward.

Analytical computation of this posterior distribution is not tractable. The inference mechanism that we propose is a MCMC algorithm inspired by previous research on DPMM inference. The state of the Markov chain is represented by

Algorithm 1 LearnPDFC$(\omega^i_{1:T}, M, R, S, N_{iter})$

```
for n = 1..N_iter do
    τ ← incremental-move(τ, α, λ, a_{1:T}, ω^j_{1:T}, M)
    if n mod R = 0 then
        τ ← split-merge(τ, α, λ, a_{1:T}, ω^j_{1:T}, S)
    end if
    (a_{1:T}, s_{1:T}, ω^j_{1:T}) ← sample-seq(τ, λ, ω^i_{1:T})
    (α, λ) ← sample-hyperpars(τ, α, λ, a_{1:T}, ω^j_{1:T})
end for
```

the tuple $(\tau, s_{1:T}, a_{1:T}, \omega^j_{1:T})$[1]. In order to sample the next state of the Markov chain, we use a Gibbs sampler that considers individual transitions τ_{kgh}. We call these **incremental moves**, that change at most one PDFC transition at a time. Incremental moves alone are sometimes insufficient to quickly converge to the true mode of the posterior probability, since this might require passing through low-probability intermediate regions of the sample space (Jain and Neal 2004). For this reason, we implement **split-merge moves**, that split a whole node or collapse two nodes in a single step, thus enabling a more effective exploration of the sample space. Split-merge moves are computationally more expensive than incremental moves, therefore they are applied only every R^{th} iteration, where R is a parameter of the MCMC algorithm. At each iteration, the sequences $s_{1:T}, a_{1:T}, \omega^j_{1:T}$ and the hyperparameters α and λ are also resampled. Algorithm 1 captures the overall structure of the MCMC sampler, and its operations are described in the following.

Incremental Moves. In order to perform an incremental Gibbs move, we first sample a single transition source (k, g, h) uniformly at random out of the $K|A||\Omega|$ transitions of the PDFC in the current state, where K is the current number of PDFC nodes. We then proceed to sample the destination of this transition from the probability distribution:

$$\begin{aligned} &p(\tau_{kgh}|\alpha, \tau_{-(kgh)}, \omega^j_{1:T}, a_{1:T}) \\ &\quad \propto p(\tau_{kgh}|\alpha, \tau_{-(kgh)})\ p(a_{1:T}|\tau, \omega^j_{1:T}), \end{aligned} \tag{3}$$

where $\tau_{-(kgh)}$ denotes all current values of τ except the one being sampled. The first term of the RHS side of Equation 3 is the conditional prior distribution, given by:

$$\begin{aligned} p(\tau_{kgh} = i|\alpha, v) &\propto v_i \quad \text{for existing node } i \\ p(\tau_{kgh} = \bar{i}|\alpha, v) &\propto \alpha \quad \text{for new node } \bar{i}, \end{aligned} \tag{4}$$

where v_i is the number of transitions in $\tau_{-(kgh)}$ that point to node i. This formula implements the so-called Chinese restaurant process (CRP) rule, that follows strictly from the stick-breaking prior described in Section 3.

The second term of the RHS is the likelihood that the new assignment awards to the action sequence $a_{1:T}$, given j's observation sequence $\omega^j_{1:T}$. For existing nodes, this can be computed by considering that τ, $a_{1:T}$, and $\omega^j_{1:T}$ jointly determine the values of the node sequence $q_{1:T}$. Let us introduce

[1] θ and π are not present because, as we will see, they can be integrated out analytically. Moreover, the node sequence $q_{1:T}$ is not part of the state since it depends deterministically on the other variables.

a count matrix d, where each element d_{kg} represents how many times action g is generated in node k in such sequence. Given that each action is conditionally independent, we can use the properties of the Dirichlet-multinomial model (Gelman et al. 2003), and marginalize over the parameters θ_k's:

$$p(a_{1:T}|\tau, \omega^j_{1:T}) = \prod_{k=1}^{K} \left[\frac{\Gamma(\lambda)}{\Gamma(d_{k\cdot} + \lambda)} \prod_{g=1}^{|A|} \frac{\Gamma(d_{kg} + \lambda/|A|)}{\Gamma(\lambda/|A|)} \right], \tag{5}$$

where $d_{k\cdot} = \sum_{g=1}^{|A|} d_{kg}$ is the number of times k is visited.

Computing the likelihood term for the new node is more complicated, since when a new node is considered, its own outgoing transitions need to be evaluated. According to the prior, such transitions can in turn point to some other new node, and so on recursively. It is therefore unfeasible to sum over all the countably infinite transition configurations that stem out of the new node. This situation is akin to DPMMs with non-conjugate priors, where the components' parameters cannot be integrated analytically. A simple solution to this problem could be to use a Metropolis-Hastings (MH) step instead of Gibbs to sample the new transition, similar to the algorithm proposed in (Pfau, Bartlett, and Wood 2010). In our case, however, such method leads to slow mixing rates. A better solution is to adapt the *auxiliary variables* algorithm of (Neal 2000). The key idea is to approximate the integration over possible new nodes by sampling M candidate transition configurations for the new node from the conditional prior distribution, which is obtained by recursively sampling from the CRP until no new node is generated. Once the likelihood of these candidates is evaluated, we sample the transition τ_{kgh} from Equation 3, distributing α uniformly among the M candidates, so that the total prior probability of generating a new node is still proportional to α.

Splitting and Merging Nodes. This step starts by sampling two transition sources uniformly at random. If these transitions point to the same node, a split of such node is proposed, otherwise a merge of the two destination nodes is proposed. Once a split or merge is proposed, it is accepted or rejected using the MH criterion. In order to propose high-likelihood splits, the algorithm described in (Jain and Neal 2007) is adapted to our case. When splitting a node, its incoming transitions are re-directed towards either one of the two newly created nodes using S iterations of a "restricted Gibbs sampler" (S is a parameter of the algorithm,) that also samples the new nodes' outgoing transitions. This produces a split that reflects to some extent the observed data instead of being just randomly sampled, and hence has a higher chance of being accepted. A merge is proposed using a similar method, that collapses two nodes into one and samples its outgoing transitions.

Sampling Sequences. The sequences $a_{1:T}$, $s_{1:T}$, and $\omega^j_{1:T}$ are sampled together and in block, given the PDFC in the current state. For each timestep, the following formula can be derived:

$$\begin{aligned} &p(a_{t-1}, s_t, \omega^j_t | s_{t-1}, q_{t-1}, \omega^i_{t:T}, \tau) \\ &\quad \propto p(a_{t-1}|q_{t-1})\ p(s_t|s_{t-1}, a_{t-1})\ p(\omega^j_t|a_{t-1}, s_t) \\ &\quad \times p(\omega^i_t|a_{t-1}, s_t)\ p(\omega^i_{t+1:T}|s_t, q_t = \tau_{q_{t-1}a_{t-1}\omega^j_t}). \end{aligned} \tag{6}$$

All but the last term of the RHS are known from either the current PDFC or the world's dynamics. The last term can be efficiently pre-computed as a backward probability message, using the following recursion:

$$\begin{aligned} &p(\omega^i_{t+1:T}|s_t, q_t) = \xi_t(s_t, q_t) \\ &\quad = \textstyle\sum_{a_t} p(a_t|q_t) \sum_{s_{t+1}} p(s_{t+1}|s_t, a_t)\ p(\omega^i_{t+1}|a_t, s_{t+1}) \\ &\qquad \times \textstyle\sum_{\omega^j_{t+1}} p(\omega^j_{t+1}|a_t, s_{t+1})\ \xi_{t+1}(s_{t+1}, \tau_{q_t a_t \omega^j_{t+1}}), \end{aligned} \tag{7}$$

starting at $\xi_T(\cdot, \cdot) = 1$ and proceeding backward. Once $\xi_{1:T}$ is computed, the new values of $a_{1:T}$, $s_{1:T}$, and $\omega^j_{1:T}$ can be sampled using Equation 6, moving forward from $t = 1$ to T.

Sampling Hyperparameters. In order to make the learning more flexible, we place a hierarchical $\exp(0.1)$ prior on both the concentration parameter α and the Dirichlet distribution parameter λ. At each iteration the parameters are resampled using MH with lognormal proposal distribution. The likelihood terms in the MH acceptance ratios are computed as in Equations 1 and 5, respectively for α and λ.

5 Interactive POMDPs

As in the interactive POMDP framework (Gmytrasiewicz and Doshi 2005), we extend the definition of POMDP to multiagent settings by defining a tuple $\bar{P}_i = (\bar{S}_i, A, \Omega_i, T, O_i, R_i)$, where A is the set of *joint* actions $a = (a_i, a_j)$ and the transition function T describes how the world evolves as an effect of joint actions; similarly, O_i and R_i specify how agent i receives observation and rewards, depending on the state and the joint action performed. The *interactive state space* $\bar{S}_i = S \times M_j$ is the cross product of the physical state and the set of possible models of agent j, each of which is a tuple $m_j = (h_j, f_j, O_j)$, where $f_j : H_j \to \Delta(A_j)$ is an agent function mapping observation histories to distributions over actions, and h_j is a particular observation history. Here, we consider models where the agent function is implemented by a PDFC $c_j \in C_j$ and the history of observations is replaced by the internal state q_j, i.e. $m_j = (q_j, c_j, O_j)$. Note that, with respect to the more general I-POMDP case, we only consider a class of *subintentional* models for other agents (i.e. their reward function is not explicitly modeled.)

An interactive state is therefore a tuple $\bar{s} = (s, q_j, c_j, O_j)$. We assume that the PDFC of the other agent does not change during execution, and its observation model of O_j is known. Moreover, we consider a finite set of PDFCs C_j (it will be the finite ensemble of models obtained during the learning phase.) The belief update function returns the probability of an interactive state when action a_i is executed and observation ω_i is received, given the previous belief over $\bar{S}_i$. The formula can be derived as in I-POMDPs, by considering agent i's prediction over j's action, as follows:

$$p(\bar{s}'|\bar{b}, a_i, \omega_i) = \beta \sum_{\bar{s}\,:\,c_j=c'_j} \bar{b}(\bar{s}) \sum_{a_j} p(a_j|q_j, c_j)\; O_i(a_i, a_j, s', \omega_i) \times T(s, a_i, a_j, s') \sum_{\omega_j} O_j(a_i, a_j, s', \omega_j)\; p(q'_j|c_j, q_j, a_j, \omega_j), \tag{8}$$

where β is a normalization constant, and $p(a_j|q_j, c_j)$ and $p(q'_j|q_j, c_j, \omega_j)$ are derived from the components θ and τ of PDFC c_j. The value function is defined similarly to POMDPs, and its property of piece-wise linearity and convexity carries over from the single-agent POMDP case.

Standard POMDP algorithms can be adapted to solve subintentional I-POMDPs since there are no nested intentional beliefs. However, the size of the interactive state space can be very large, even for simple problems, since we are considering a potentially large number of models of agent j. In this paper, we use the Monte Carlo tree search method for POMDPs (POMCP) described in (Silver and Veness 2010) to solve the subintentional I-POMDPs. The running time of the algorithm is virtually independent from the size of the problem, since it makes use of a generator implementation of the I-POMDP rather than a flat or factorized specification.

6 Experimental Results

We evaluate our approach on three problems of varying complexity. The first is the multiagent **Tiger Problem** introduced in (Gmytrasiewicz and Doshi 2005). The optimal (true) controller used by agent j has 5 nodes. The second problem is a variation of the **3x4 Maze** problem described in (Russell and Norvig 2009). Agent i is tasked with chasing j (reward 1), while j receives a reward of 20 when reaching the top-right corner and pays 1 when caught. Each move costs the agents 0.04, and is successful with 0.8 probability. Agent j receives warning signals when adjacent to i but is unaware of its own position, except for a specific location where it is revealed. The true controller used by j has 42 nodes. Agent i knows its own position and its observations reveal j's position with 0.8 accuracy. The third problem is a 5×5 instance of the **UAUV** reconnaissance problem described in (Zeng and Doshi 2012), with the difference that observations are received by i at each timestep and reveal j's position with 0.9 accuracy. Agent j tries to reach a safehouse (reward 1) while i is tasked with intercepting j (reward 1). Each move costs 0.04 and is deterministic for both agents. Agent j only perceives its location relative to the safehouse when adjacent to it with accuracy 0.8. The true controller of agent j has 36 nodes. For all problems, we assume that j is unaware of i's presence, except for the Maze in which j models i as stationary at a given location. Our evaluation has two purposes. First, we want to validate our PDFC learning algorithm: if we can observe j's behavior directly, are we able to learn PDFCs that are close to the true one? Second, we want evaluate agent's i performance gain when learning and planning under realistic observability conditions.

6.1 PDFC Learning

We evaluate how similar the learned PDFCs are to j's true controller, when i is able to observe perfectly j's action and observation sequence (making line 6 in Algorithm 1 superfluous.) To derive a similarity measure, suppose that a hypothetical agent operates according to the true controller c_T, and another does so according to the learned controller c_L. Let us denote as $\eta_{q_L q_T}$ the co-frequency, or probability of the two agents being simultaneously in nodes q_L and q_T of the respective controllers. We then define the *weighted Kullback-Leibler distance* between the two controllers as:

$$wKL(c_L, c_T) = \sum_{(q_L, q_T) \in Q_L \times Q_T} \eta_{q_L q_T}\; KL(\theta_{q_L}, \theta_{q_T}), \tag{9}$$

which is a measure of similarity between the distributions over actions sequences induced by the two controllers (de la Higuera 2010), hence reflecting a *behavioral* similarity between PDFCs. For each of the considered problems and for different lengths of observed history (T_{learn}), we performed 10 learning trials. The following parameters were used for the MCMC sampler: $M = 50, R = 50, S = 2$. In each trial, the MCMC sampler was run for 5000 iterations, and the second halves of the generated sample chains were subsampled every 100 iterations, resulting in ensembles of 25 PDFCs per trial. We computed the overall mean wKL across trials, which is reported in Figure 2 along with the median size of the learned PDFCs.

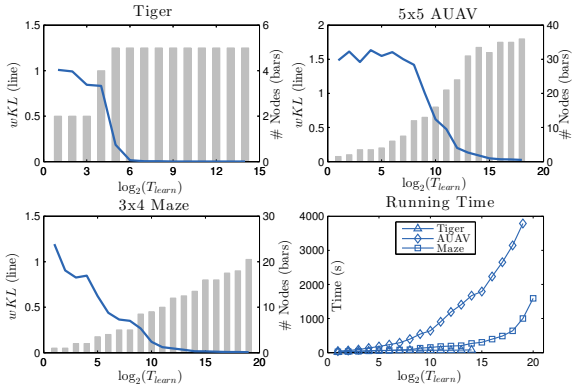

Figure 2: Weighted KL distance between the learned and the true controllers (line) and number of nodes of learned PDFC (bars). The fourth panel reports the timing results.

For the Tiger problem, we see that the wKL quickly approaches zero ($T_{learn} \geq 64$) and the number of nodes stabilizes at 5, the size of the true controller. For the other two problems, the wKL decreases more gradually, eventually converging towards zero. For the AUAV problem, the number of nodes approaches the size of the true controller (36 nodes) for long sequences. In the Maze problem, the size of the PDFCs grows steadily but remains lower than the true size (42 nodes), even when the wKL approaches zero. While it seems that for this problem we may need an impossibly long sequence to eventually learn the true number of nodes, the learned PDFCs are behaviorally very close to j's true controller. In order to shed some light over this result, Figure 3 reports the fraction of time spent in each node of the true controller, sorted in decreasing order. We can see that the distribution decreases rapidly for the Maze

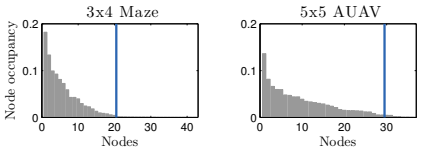

Figure 3: Node occupancy for Maze and AUAV problems. The vertical line indicates the 99% percentile.

problem, with less than 1% of the time spent in more than 50% of the nodes. This means that most of the complexity of the observed agent's behavior can be captured with fewer nodes. For the AUAV problem, the distribution decreases more gradually, meaning that more nodes are required to accurately describe the observed behavior. The relation between node occupancy and convergence to the true controller is important to establish theoretical properties or learning, and will be explored more in depth in future work.

The bottom-right panel of Figure 2 reports the running time of the MCMC algorithm[2], which is at most linear in the amount of data considered. Notice that the growth rate for the AUAV problem is almost constant for large values of T_{learn}, while it increases for the Maze problem, indicating higher dependence on the PDFC's size than on T_{learn}. This is due to optimizations in the computation of the quantities d used in Equation 5, which contains the only dependency of running time on T_{learn} (since line 6 is not executed here.)

6.2 Learning and Planning

In this section, we drop the assumption of perfect observability of agent j's behavior. Even so, we show how PDFC learning allows agent i to improve its performance. In our setting, j is oblivious to i's actions, and always operates according to its true controller. We consider the reward collected by agent i with respect to the amount of observations used for learning j's models, and compare it against the following baselines: (**U**) i models j's actions uniformly at random, $p(a_j) = |A|^{-1}$; (**P**) i predicts j's action proportionally to their long-term frequency; and (**T**) i knows j's true PDFC. For each problem and observation size, we performed 20 learning and planning trials; for each trial, the MCMC algorithm was run for 5000 iterations and 25 sample PDFCs were retained as before, which constitute C_j in our I-POMDP model. The performance of the resulting I-POMDPs was then computed by averaging the total reward collected during 1000 runs of the POMCP algorithm, with discount factor 0.9 and using 2^{10} simulations for exploring the search tree at each step; all other POMCP parameters were set as in (Silver and Veness 2010).

Figure 4 reports agent i's mean total reward for the three problems and the median size of j's PDFCs. Numbers along the x-axes indicates the base-2 logarithm of the observation size, while letters identify baseline models. Notice that, since the Tiger problem is much smaller, we consider shorter

[2]Implemented in MATLAB® and running on an Intel® Xeon® 2.27 GHz processor.

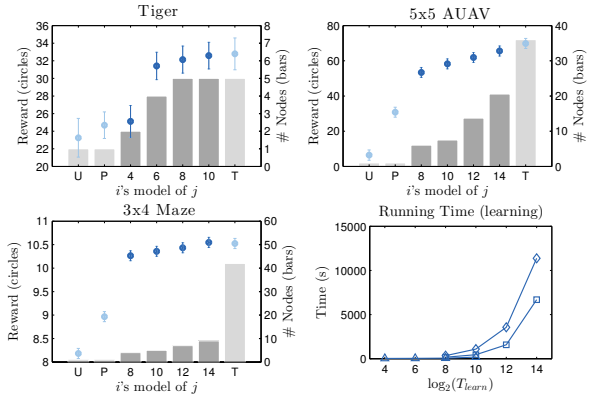

Figure 4: Reward by agent i with different models of agent j (lines) and number of nodes of learned PDFCs (bars). The fourth panel reports learning time results.

training sequences. For all problems, the performance obtained when using the learned models of j is always higher than using uniform or proportional models, and approaches the upper bound (known j's true model) with longer training sequences. This is because, with more information available, agent i is able to learn more accurate models that better predict j's actions. This is also reflected in the size of the inferred controllers, which as expected grows with the amount of data. However, the PDFCs learned in this settings are usually smaller than the ones learned with perfect observability of j's behavior (previous section.) This is because now i's perception of j's behavior is filtered by noisy world's dynamics, and therefore longer observations are needed to allow identification of the same behavioral patterns.

We underline how, even with shorter observation sequences, agent i is able to learn models that provide a large performance gain over the random or proportional models. In particular for the Maze problem, using only 256 observations for learning, agent i's performance grows to about 90% of the difference between using the true model and the random model. Similar, albeit less extreme jumps are also observable for the other problems. This is a demonstration that, even though learning the exact model of another agent is unattainable, especially with realistic observability assumptions, we can still largely improve our performance by recognizing behavioral patterns that are statistically significant and encode them in a compact model.

The fourth panel of Figure 4 reports the learning times. With respect to the previous section, we observe that sampling sequences makes the procedure more time consuming. Unfortunately, this sampling is an iterative procedure that cannot be vectorized or optimized algorithmically. However, other choices of implementation language can make the procedure much faster, and preliminary results have shown no noticeable loss in performance if sequences are sampled less frequently than every iteration.

7 Conclusion and Future Work

In this paper, we have introduced a suitable Bayesian prior for PDFCs that model the behavior of other agents, and characterized such distribution in terms of number of nodes. We presented an ad-hoc MCMC inference algorithm that works with imperfect observations, and explained how the learned models of other agents can be embedded in the modeling agent's decision making process. Our results show that it is possible to learn accurate models using our approach, and that the reward of the modeling agent increases as a result of this learning even when the behavior is not directly observable. Future work will be focused on developing an online method that interleaves learning and planning, in order to extend the applicability of our work to more realistic settings.

References

Carmel, D., and Markovitch, S. 1996. Learning models of intelligent agents. In *Proceedings of the 13th National Conference on Artificial intelligence*, 62–67.

Conroy, R.; Zeng, Y.; Cavazza, M.; and Chen, Y. 2015. Learning behaviors in agents systems with interactive dynamic influence diagrams. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 39–45.

de la Higuera, C. 2010. *Grammatical Inference: Learning Automata and Grammars*. New York, NY, USA: Cambridge University Press.

Doshi-Velez, F.; Pfau, D.; Wood, F.; and Roy, N. 2013. Bayesian nonparametric methods for partially-observable reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 99(PrePrints):1.

Doshi, P.; Zeng, Y.; and Chen, Q. 2009. Graphical models for interactive POMDPs: Representations and solutions. *Autonomous Agents and Multi-Agent Systems* 18(3):376–416.

Gelman, A.; Carlin, J. B.; Stern, H. S.; and Rubin, D. B. 2003. *Bayesian Data Analysis, Second Edition*. Chapman and Hall/CRC, 2 edition edition.

Gmytrasiewicz, P. J., and Doshi, P. 2005. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research* 24(1):49–79.

Green, P. J., and Richardson, S. 2001. Modelling heterogeneity with and without the Dirichlet process. *Scandinavian Journal of Statistics* 28(2):355–375.

Hansen, E. 1998. Solving POMDPs by searching in policy space. In *Proceedings of the 14th International Conference on Uncertainty In Artificial Intelligence*, 211–219.

Hjort, N. L.; Holmes, C.; Müller, P.; and Walker, S. G., eds. 2010. *Bayesian Nonparametrics*. Cambridge University Press.

Jain, S., and Neal, R. M. 2004. A split-merge markov chain Monte Carlo procedure for the Dirichlet process mixture model. *Journal of Computational and Graphical Statistics* 13(1):pp. 158–182.

Jain, S., and Neal, R. M. 2007. Splitting and merging components of a nonconjugate Dirichlet process mixture model. *Bayesian Anal.* 2(3):445–472.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101:99–134.

Liu, M.; Amato, C.; Liao, X.; Carin, L.; and How, J. P. 2015. Stick-breaking policy learning in Dec-POMDPs. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 2011–2018.

Liu, M.; Liao, X.; and Carin, L. 2011. The infinite regionalized policy representation. In Getoor, L., and Scheffer, T., eds., *Proceedings of the 28th International Conference on Machine Learning*, 769–776.

Mccallum, A. K. 1996. *Reinforcement Learning with Selective Perception and Hidden State*. Ph.D. Dissertation, The University of Rochester.

Meuleau, N.; Peshkin, L.; Kim, K.-e.; and Kaelbling, L. P. 1999. Learning finite-state controllers for partially observable environments. In *Proceedings of the 15th International Conference on Uncertainty In Artificial Intelligence*, 427–436.

Miller, J. M., and Harrison, M. T. 2015. Mixture models with a prior on the number of components. *CoRR* arXiv:1502.06241v1 [stat.ME]. preprint.

Neal, R. M. 2000. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics* 9(2):pp. 249–265.

Oliehoek, F. A., and Amato, C. 2014. Best response Bayesian reinforcement learning for multiagent systems with state uncertainty. In *AAMAS Workshop on Multiagent Sequential Decision Making Under Uncertainty*.

Paisley, J., and Carin, L. 2009. Hidden Markov models with stick-breaking priors. *Trans. Sig. Proc.* 57(10):3905–3917.

Pfau, D.; Bartlett, N.; and Wood, F. 2010. Probabilistic deterministic infinite automata. In *Advances in Neural Information Processing Systems*, 1930–1938.

Poupart, P., and Boutilier, C. 2003. Bounded finite state controllers. In *Advances in Neural Information Processing Systems 16*.

Ross, S.; Draa, B. C.; and Pineau, J. 2007. Bayes-adaptive POMDPs. In *Proc.˜of the Conference on Neural Information Processing Systems*.

Rubinstein, A. 1998. *Modeling Bounded Rationality*. MIT Press.

Russell, S., and Norvig, P. 2009. *Artificial Intelligence: A Modern Approach (3rd Edition)*. Prentice Hall, 3rd edition edition.

Sethuraman, J. 1994. A constructive definition of Dirichlet priors. *Statistica Sinica* 4:639–650.

Silver, D., and Veness, J. 2010. Monte-Carlo planning in large POMDPs. In Lafferty, J.; Williams, C.; Shawe-Taylor, J.; Zemel, R.; and Culotta, A., eds., *Advances in Neural Information Processing Systems 23*, 2164–2172. Curran Associates, Inc.

Wright, J. R., and Leyton-Brown, K. 2012. Behavioral game theoretic models: a Bayesian framework for parameter analysis. In *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 4-8, 2012 (3 Volumes)*, 921–930.

Zeng, Y., and Doshi, P. 2012. Exploiting model equivalences for solving interactive dynamic influence diagrams. *J. Artif. Int. Res.* 43(1):211–255.