Building Adaptive Dialogue Systems Via Bayes-Adaptive POMDPs

Shaowei Png, Joelle Pineau, and Brahim Chaib-draa

***Abstract*—Recent research has shown that effective dialogue management can be achieved through the Partially Observable Markov Decision Process (POMDP) framework. However past research on POMDP-based dialogue systems usually assumed the parameters of the decision process were known *a priori*. The main contribution of this paper is to present a Bayesian reinforcement learning framework for learning the POMDP parameters online from data, in a decision-theoretic manner. We discuss various approximations and assumptions which can be leveraged to ensure computational tractability, and apply these techniques to learning observation models for several simulated spoken dialogue domains.**

***Index Terms*—Dialogue management, reinforcement learning, Markov decision process (MDP), partially observable Markov decision process (POMDP), Bayesian inference.**

I. Introduction

In a world where computer workstations, digital assistants, phone lines, and many other devices, can be controlled by the simple use of voice commands, the importance of dialogue systems is no longer in question [1]. Yet there are many situations where using a speech-based interface can be a frustrating experience. Whether it is due to a person's speech pattern, poorly tuned speech-recognition software, ambient noise, or a poorly designed conversation protocol, speech interfaces are often problematic to use in real-world situations.

Dialogue systems have been modeled effectively using Partially Observable Markov decision processes (POMDPs), but past research on POMDP-based dialogue systems typically assumes that the model parameters are known *a priori* [2]–[4]. The primary contribution of this paper is to present a robust framework for joint learning and decision-making of observation models in POMDP-based dialogue management systems[1]. Our approach uses the Bayes-Adaptive POMDP (BAPOMDP) framework [6], [7] to define a Bayesian learning process over the uncertain parameters. While this provides an elegant decision-theoretic framework for the problem of robust dialogue management, it presents several limitations in terms of data and computational complexity. In this paper, we present algorithmic approaches to overcome these limitations, including the use of structural assumptions to reduce the data requirements, and the use of online POMDP search methods to reduce the inference time. We present empirical results with several simulated dialogue domains, including a dialogue manager based on the SACTI1 corpus, and two dialogue domains motivated by deployment of speech-controlled assistive technology devices.

Manuscript received July 19, 2012; revised October 09, 2012; accepted November 05, 2012. Date of publication November 27, 2012; date of current version January 03, 2013. This work was supported by NSERC and FQRNT. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Kai Yu.

S. Png was with the School of Computer Science, McGill University, Montreal, QC H3A 2A7, Canada. He is now with Google, Seattle, WA 98103 USA (e-mail: shaowei@google.com).

J. Pineau is with the School of Computer Science, McGill University, Montreal, QC H3A 2A7, Canada (e-mail: jpineau@cs.mcgill.ca).

B. Chaib-draa is with the Computer Science Department, Laval University, Quebec QC G1V 0A6, Canada (e-mail: chaib@ift.ulaval.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSTSP.2012.2229962

[1]An early version of this work, containing only preliminary empirical results, was presented in [5].

II. Technical Background

A. Dialogue Systems

A standard architecture for a spoken dialogue system includes: speech recognizer, parser/interpreter, dialogue manager, response generator, speech synthesizer [8]. As our main focus is on the development of the dialogue manager, we assume standard public or commercial components for the speech recognizer and speech synthesizer, and simple script-like components for the parser/interpreter and response generator. The role of the dialogue manager is to find a good mapping from parsed input text, to output utterances. For computational reasons, we usually assume discrete input and output sets. These sets can correspond to simple indexed strings, or structured logical forms [9].

One of the main challenges in building a good dialogue manager consists in finding a mapping from input to output which is robust to noise and missing data in the input. In practice, this corresponds to errors in the speech recognition and parsing steps. Another challenge is in handling the sequential aspect of the conversation, meaning that the choice of response must depend not only on the latest utterance, but also on previous interactions. To address these challenges, much of the recent research on developing dialogue managers has focused on probabilistic methods for sequential decision-making.

The early work using probabilistic methods has primarily adopted the Markov Decision Process (MDP) for the dialogue management, in which case the current state of the dialogue is assumed completely observable (known) throughout the interaction [10]–[16]. However this model is inadequate to capture the uncertainty introduced by errors in speech recognition and language understanding. This has led to more recent research efforts towards the Partially Observable Markov Decision Process (POMDP) where the state of dialogue is considered partially observable, thus allowing partial or uncertain observations to be considered by the dialogue manager. POMDPs have been shown to be well suited for computing the optimal dialogue strategy under unreliable automatic speech recognition (ASR) [2]–[4], [17], [18].

B. Partially Observable Markov Decision Processes

Formally, a discrete POMDP is specified as a tuple $(S, A, Z, T, O, R, \gamma)$, where S is a set of states, A is a set of actions, and Z is a set of observations [19]–[21]. Throughout this paper, we assume that S, A and Z are discrete sets, which are specified by the dialogue designer. The states, S, capture features of the dialogue task, the actions A correspond to the possible outputs of the dialogue managers (answers to queries, requests for clarification, etc.), and the observations, Z, are defined by the input to the dialogue manager (e.g., parsed strings, logical structures.)

At each time step, the agent is in some state $s \in S$ and when it takes an action $a \in A$, it moves to state $s' \in S$. Uncertainty in the effect of the action a is characterized by the transition probability function $T(s, a, s') = \Pr(s'|s, a), \forall s \in S, \forall a \in A$. While the state is not always observable, the agent receives an observation $z \in Z$, which provides information about the state, as defined by the observation probability function $O(s', a, z) = \Pr(z|s', a), \forall s' \in S, \forall a \in A$. The agent's behavior, or choice of actions, is further characterized by the reward function, $R(s, a)$, defining the reward received for taking action a in state s. The agent's objective is to choose actions such as to maximize the expected cumulative sum of rewards over a given planning horizon T. In the case of infinite planning horizon, we usually consider the *discounted* sum of rewards, with discount factor $\gamma \in [0, 1)$. For finite planning horizons T, we have$\gamma = 1$, whereas for infinite horizon, we use the expected discounted total reward: $E[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)]$, where s_t and a_t denote the agent's state and action at time t.

Since the state is typically not fully observable, the agent can track the *information state*, which captures all experiences up to the current time step. This can be tracked in the form of the *history*, denoted by $h_t = a_0, z_1, a_1, z_2 \ldots, a_{t-1}, z_t$. The history quickly grows in size, and is usually replaced by a sufficient statistic, called the *belief* [19]:

$$\begin{aligned} b'(s') &= \Pr(s'|z, a, b) && (1) \\ &= \frac{\Pr(z|s', a, b)\Pr(s'|a, b)}{\Pr(z|a, b)} && (2) \\ &= \frac{O(s', a, z)\sum_{s \in S} T(s, a, s')b(s)}{\Pr(z|a, b)}. && (3) \end{aligned}$$

Thus, the belief state b can be updated to the successor belief b' given an action a and an observation z according to Bayes theorem (as reflected by previous (3)), which we denote by $b' = \tau(b, a, z)$ such that

$$b'(s') = kO(s', a, z) \sum_{s \in S} T(s, a, s')b(s) \qquad (4)$$

where k is a normalizing constant denoting $1/\Pr(z|a, b)$. Equation (4) shows that the belief is a probability distribution over states, which can be updated online every time the agent acquires a new action/observation.

We now consider three different inference processes pertaining to POMDPs. First, there is the question of efficiently *tracking* the belief; for discrete spaces with known transition, observation and reward models, this can be done in polynomial time using (4) above [21]. Second, there is the *planning* problem which consists in finding a sequence of actions that optimizes the expected cumulative reward; this is discussed in the following subsection (assuming again that S, A, Z, T, O, R are known *a priori*). Third, there is the problem of *learning* the transition, observation and reward parameters from data. This is crucial for domains where the dynamic models (transition and observation parameters) are difficult to characterize analytically, such as in dialogue systems. The main contribution of this paper is a Bayesian learning framework for simultaneously tackling the planning and learning problems in dialogue systems.

C. Planning in POMDPs

The POMDP planning solution is usually expressed in the form of a *policy*, $\pi : b \rightarrow A$, a mapping from beliefs to action choices. We can define the expected value of a given policy π as follows: $V^{\pi}(b) = E_{\pi}[\sum_{t=0}^{T} \gamma^t r_t | b_0]$, where r_t is the reward received at timestep t, γ is the discount factor, and b_0 is the initial belief at time $t = 0$. The optimal policy for a POMDP is one that chooses an action maximizing the expected future discounted cumulative reward: $\pi^*(b_{t_0}) = \arg\max_{\pi} E_{\pi}[\sum_{t=0}^{T} \gamma^t r_t | b_0]$, with associated value function:

$$\begin{aligned} V^*(b) &= E_{\pi^*}\left[\sum_{t=0}^{T} \gamma^t r_t | b_0\right] \\ &= \max_a \left[\sum_{s \in S} b(s)R(s, a)\right. \\ &\quad \left. + \gamma \sum_{z \in Z} \Pr(z|b, a)V^*(\tau(b, a, z))\right]. \qquad (5) \end{aligned}$$

Computing the optimal policy is computationally challenging due to the nature of the belief. In a problem with n physical states, the corresponding belief space has dimensionality $n - 1$. The number of reachable beliefs at time t, b_t, typically grows exponentially with the planning horizon [20], [21]. Most dialogue managers modeled in the POMDP framework are not amenable to exact solutions. However in the case where the model parameters (transitions, observations and rewards) are known *a priori*, dialogue managers can often be solved with more recent point-based approximations [22]–[24], even for fairly large-size dialogue domains.

In cases where the parameters are not known in advance, a better option is to consider online POMDP methods, which have the ability to re-plan incrementally, as information is acquired about the model's parameters. Online POMDP planners reduce the complexity of the problem by planning online for only the current information state [25]–[27]. They typically apply forward search, from the current belief state, and form a local approximation to the optimal value function by considering only a short receding horizon of possible scenarios.

The Bayes-Adaptive POMDP framework presented in the next section incorporates Real Time Belief State Search (RTBSS) [28], [29] to perform online POMDP solving. Other online approaches are available in the literature (for a review see [25], and more recently [30]), however RTBSS is particularly simple to implement, and sufficiently efficient for our purposes. RTBSS is a forward branch and bound search in the belief space. The top node corresponds to the current belief, and branches are generated for possible action/observations pairs from that belief. To maintain tractability, RTBSS cuts down redundant sub-trees by pruning away some actions. To decide which branch to prune, RTBSS maintains an upper and lower bound at each node, sorts the actions (l.5 of Algorithm 1) according to those that are most likely to return higher values (this can be approximated by comparing the immediate reward of the actions), and prunes those actions that are unlikely to improve the maximum returned value found so far.

The algorithm is described in Algorithm 1. In line 9, the upper bound is computed as the sum of the current reward (cR) and the heuristic value ($\text{Heuristic }(b, a, d)$). The current reward is defined by:

$$R_B(b,a) = \sum_{s \in S} b(s)R(s,a). \quad (6)$$

At line 10, if the upper bound value is less than the maximum value, the action is then pruned and the subtree is not expanded, reducing the time taken. When the depth d is 0, we simply compute the immediate return of the belief state. This is defined by $U(b) = \max_{a \in A} R_B(b,a)$. The heuristic value $\text{Heuristic }(b, a, d)$ has to be an upper bound on the value function at belief b. Ideally, the heuristic value has to be the maximal value that any algorithm can find if it searches the tree to the maximal depth d. This ensures that we do not prune the wrong actions. The heuristic can be selected using domain-knowledge. For example, for problems where only the terminal state receives a positive reward, a good heuristic might be to consider that we will always reach the terminal state from any belief state, therefore setting the upper bound to the reward one gets by doing an action that reaches the terminal state. In the absence of any domain knowledge, one can always use $R_{\max}(s,a)/(1-\gamma)$, although setting the upper bound to this value will generally yield no pruning. Further details about RTBSS can be found in [28], [29].

Algorithm 1 The RTBSS algorithm [28], [29].

```
1:  Function RTBSS($b, d$) returns the estimated value
    of $b$
    Inputs: $b$: the current belief state;
    $d$: the current depth
    Statics: $D$: the maximal search depth;
    action: the best action
2:  if $d = 0$ then
3:     return $U(b)$
4:  end if
5:  $actionList \leftarrow Sort(b, A)$
6:  $max \leftarrow -\infty$
7:  for all $a \in actionList$ do
8:     $cR \leftarrow R_B(b, a)$
9:     $uBound \leftarrow cR + Heuristic(b, a, d)$
10:    if $uBound > max$ then
11:       for all $z \in Z$ do
12:       $cR \leftarrow cR + \gamma \Pr(z|b, a)\text{RTBSS}(\tau(b, a, z), d-1)$
13:       end for
14:       if $cR > max$ then
15:         $max \leftarrow r$
16:         if $d = D$ then
17:           $action \leftarrow a$
18:         end if
19:       end if
20:    end if
21: end for
22: return $max$
```

D. Learning in POMDPs

The POMDP planning approaches cited above generally rely on having known transition, observation and reward parameters. Such models may be well-known in some domains, for example robotics, or resource management problems. However this is highly impractical in most dialogue domains, where statistical characterization of the uncertainty cannot easily be achieved analytically. Machine learning techniques can be applied to overcome this limitation.

There are many approaches for learning from data in POMDPs. The first, **supervised learning** (or model-based reinforcement learning), requires large amounts of annotated data to achieve good performance [31], [32] this is particularly challenging in dialogue-type tasks, because the annotations need to convey the underlying *state* of the system, which in many applications may depend on a user's *intent* which is definitely not an easy annotation to make on a large dataset. An alternative approach is to use **reinforcement learning** to jointly learn the model policy using gradient methods [33], [34], which can be useful when there is a good prior on the policy class. The data requirements of supervised and reinforcement learning methods can be alleviated somewhat by using off-policy learning [35].

Other researchers have advocated the use of **unsupervised learning** approaches which generally use Expectation-Maximization (EM), to infer parameters from an un-annotated conversational dataset [36], [37]. While the EM approach is well-established for estimating the parameters of Hidden Markov Models (HMMs) used for speech recognition, it also suffers from well-known local minima problems. The issue is exacerbated in the POMDP context, where parameters are further conditioned on action choices (i.e., they depend on the exploration policy). More recently, an alternative representation was proposed, called **predictive-state representation** (PSR) [38], which offers the expressiveness of POMDPs without explicit specification of hidden states. The framework appears more intuitive from a learning perspective. However current algorithms seem to require large amounts of data, even for small domains, thus making such a framework impractical, at this stage, for rich dialogue domains. On the other hand, **Bayesian reinforcement learning** provides a more scalable paradigm, since it can incorporate domain knowledge (in the form of a prior) to guide and constrain the learning process. It is also easily amenable to an online formulation. The main limitation is that most recent Bayesian reinforcement learning approaches deal with the fully observable case. In the next section, we present the Bayes-Adaptive POMDP (BAPOMDP) framework, which can handle the partially observable case.

III. Bayes-Adaptive POMDPs for Dialogues

The aim of Bayesian reinforcement learning (BRL) is to maintain a posterior distribution over possible models parameters and to compute an action selection policy which is optimal with respect to this posterior [39]. A general formulation for Bayesian RL is shown in Algorithm 2. Initially, distributions are initialized over the unknown parameters of the models. Thus, using the current information about those parameters, an action is selected. After the action is executed, the agent takes the resulting information about the environment and updates the distributions over the parameters.

Algorithm 2 A general framework describing Bayesian reinforcement learning. Various approaches address the challenges in each step through different techniques.

```
Initialize distributions over unknown parameters
loop
  Select action based on distributions
  Execute action
  Observe resulting reward and observation
  Update posterior of unknown parameters based on observations
end loop
```

A. Bayesian Inference in Partially Observable Domains

For Bayesian Reinforcement learning in fully observable Markov Decision Processes [39]–[41], the key idea is to maintain a posterior distribution over the unknown model parameters, and to apply planning over the full posterior (rather than over only the maximum likelihood parameters). In discrete domains, a convenient way of representing the posterior is by using Dirichlet distributions, which are probability distributions over the parameters of multinomial distributions. Given ψ_i, the frequency at which event e_i occurs over n trials, the probabilities p_i of each event has a Dirichlet distribution, i.e., $(p_1, p_2, \ldots, p_k) \sim \mathrm{Dir}(\psi_1, \psi_2, \ldots \psi_k)$. If the counts $(\psi_1, \psi_2, \ldots \psi_k)$ are observed over n trials $(n = \sum_{i=1}^{k} \psi_i)$, the distribution represents the probability of a discrete random variable according to the probability distribution $(p_1, p_2, \ldots, p_k)$. The probability density function is $f(p, \psi) = (1)/(B(\psi)) \prod_{i=1}^{k} p_i^{\psi_i - 1}$, where B is the multinomial beta function. The expected value of p_i is $E(p_i) = (\psi_i)/(\sum_{j=1}^{k} \psi_j)$.

Applying this to a fully observable Markov decision process, we can define counts $\phi_{ss'}^{a}$ of the number of times the transition $s \rightarrow_a s'$ is observed for each action a, starting from prior ϕ_0. The next state, s', becomes a combination of the physical state, $s \in S$, with the information state, ϕ. In this case, T' describes a probability of update from $(s, \phi) \rightarrow_a (s', \phi')$, that is from hyperstate (s, ϕ) to hyperstate (s', ϕ') when performing action a. Planning is applied to this extended model, using a joint state space $S \times \Phi$ (where Φ is the set of ϕ_i) and the extended transition function T'.

In a POMDP, besides having the counts $\phi_{ss'}^{a}$, we also have the observation counts, $\psi_{s'z}^{a}$, representing instances of seeing z at s' after doing action a. The state is defined over (s, ϕ, ψ), and the decision problem is defined over the joint space $S \times \Phi \times \Psi$ (where Ψ is the set of ψ_i). One of the main challenges in solving such a system is how to update the Dirichlet count parameters, ϕ and ψ, when the state is a hidden variable. At each time step, s is not observable, and neither are ϕ and ψ, since without knowing the state, we cannot know which count parameter to update. Thus we need to maintain a belief over the joint space, (s, ϕ, ψ). The objective is to learn an optimal policy, such that actions are chosen to maximize reward with respect to the posterior captured by the Dirichlet distribution.

The Bayes-Adaptive POMDP (BAPOMDP) framework allows belief inference, planning, and learning in POMDPs, under these assumptions [6]. Through the remainder of the paper, we focus on the case where only $O(s', a, z)$ is unknown, and assume that $T(s, a, s'), R(s, a)$ are known. This is done simply because the observation parameters are most relevant for practical dialogue systems, and to simplify exposition. In general, the BAPOMDP can solve problems when $T(s, a, s'), R(s, a)$ are also unknown without substantial complications [6].

Recall that uncertainty on the distribution $O(s', a, \cdot)$ is captured by the counts $\psi_{s'z}^{a}, \forall z$, representing the number of times observation z was made in state s' after doing action a, with ψ the vector of all the observation counts. The expected transition probability for $O(s', a, z)$ is $O_\psi(s', a, z) = (\psi_{s'z}^{a})/(\sum_{z' \in Z} \psi_{s'z'}^{a})$. The state space S' of the BAPOMDP is therefore defined as $S' = S \times \psi$ where $\psi \in N^{|S||A||Z|} | \forall (s, a), \sum_{z \in Z} \psi_{sz}^{a} > 0$.

It has been shown that the BAPOMDP is an instance of a POMDP [6], and as such, we need to track the belief state b. The belief state of the BAPOMDP represents a distribution over both states and count values, that is $b(s, \psi)$. If b_0 is the initial belief state of the unknown POMDP, and the count vector ψ_0 represents the prior knowledge on this POMDP, then the initial belief of the BAPOMDP is: $b_0'(s, \psi_0) = b_0(s)$, if $(\psi) = (\psi_0)$; 0, otherwise. The exact belief update for the BAPOMDP is given in Algorithm 3. To maintain tractability over longer horizons, we only keep the K most probable belief states in the new belief b', and re-normalize b' accordingly. The algorithm is linear in the number of actions/observations, and quadratic in the number of states. Exact tracking is achieved as long as fewer than K state-count pairs are possible.

Algorithmic 3 Exact Belief Update in BAPOMDP

```
Let $b$ be the current belief, $S_b'$ the set of hyper-states in $b$, $\mathcal{U}(\psi, s', a, z)$ a fn incrementing $\psi_{s,z}^{a}$ by 1 in the set of counts.
Initialize $b'$ as a 0 vector.
for all $(s, \psi) \in S_b'$ do
  for all $s' \in S$ do
    $\psi' \leftarrow \mathcal{U}(\psi, s', a, z)$
    $b'(s', \phi', \psi') \leftarrow b'(s', \phi', \psi') + b(s, \phi, \psi) T(s, a, s') O_\psi(s', a, z)$
  end for
end for
return normalized $b'$
```

B. Planning With RTBSS in BAPOMDPs

The planning portion of the initial BAPOMDP algorithm uses a simple forward search, as described in [6]. When applying BAPOMDPs to dialogue systems, we have modified this component to use the RTBSS algorithm. This allows us to cut down redundant sub-trees, thus reducing the time for the online search. In the above RTBSS, the heuristic value described in Line 9 of Algorithm 1 has to be an admissible heuristic (in the A* sense), so in our case, an upper bound on the value function. Ideally, the heuristic value has to be the maximal value that any algorithm can find if it searches the tree to the maximal depth d. This ensures that we do not prune the wrong actions.

Now we should adapt RTBSS so that we can take into consideration BAPOMDP, combining thus learning and planning. First, we use RTBSS with $\mathrm{Heuristic}(b, a, d) = \sum_{i=1}^{d} \gamma^i R_{\max}$, where d is the depth of the search and γ is the discount factor. $R_{\max}$ is the maximal reward for all states and all actions of the underlying MDP. Another important modification is that we need to interleave updates to the observation counts within the planning algorithm, this is embedded in line 12 of the RTBSS algorithm above, where we apply the belief update operator $\tau(b, a, z)$. Recall that in the case of BAPOMDP, b is expressed over the joint state (or hyperstate) (s, ψ) and consequently $R_B(b, a)$ as reflected by (6) (and line 8 of Algorithm 1), becomes in BAPOMDP: $R_B(b, a) = \sum_{(s,\phi)\in S'_b} b(s, \phi)R(s, a)$.

The belief update operation is an integral component of any POMDP planning. In the case of RTBSS for BAPOMDPs, the belief update operation over the joint state is applied using a particle filter, maintaining the K most probably belief states, as follows: $b' \leftarrow \mathrm{ParticleFilterKProbable}\,(b, a, z, k)$. It is applied just after line 11 in Algorithm 1.

C. Defining Priors in BAPOMDPs for Dialogue Systems

An important requirement of the BAPOMDP framework is the need to specify a prior distribution over model parameters, in this case the observation counts ψ. Earlier applications of BAPOMDPs assumed a low uniform count value, and then allowed each $\psi^a_{s'z}$ to vary independently. This affords substantial freedom to the posterior, but may require more data than is available. When applying BAPOMDPs to dialogue systems, it can be preferable to consider a more constrained prior. In particular, in a typical dialogue system, many parameters are likely to be similar, either because the observations correspond to acoustically related inputs, and thus the noise level is similar, or because the observations correspond to semantically related inputs, and thus the meaning is similar. We can use the knowledge that certain observation parameters have similar values to introduce constraints on the learning problem via the prior. We do this by imposing symmetry constraints (parameter tying) on the Dirichlet parameters, whereby sets of parameters are constrained to have the same posterior. Practically speaking, counts corresponding to the parameters that have been constrained to have the same value are all updated simultaneously whenever any of the observations in the set are observed. This can dramatically reduce the number of parameters to learn. In dialogue systems, this could involve assuming that the recognition error rate is independent of state information (e.g., command type or user location).

D. A Procedure for Applying BAPOMDP in Dialogues

We summarize the steps in applying BAPOMDPs to dialogue systems as follows:

1) **Represent the dialogue system as a POMDP.**
   a) Represent the state space S as a set of the user intentions and dialogue information.
   b) Represent the action space A as a set of the possible responses by the dialogue system.
   c) Represent the observation space Z as a set of the possible percepts.
   d) Define the transition probabilities $T : \Pr(s'|s, a)$ based on domain knowledge and previous data.
   e) Define the observation probabilities $Z : \Pr(z|s, a)$ based on domain knowledge and previous data. Label the observation probabilities that are unknown.
   f) Define the reward function $R(s, a)$ based on domain knowledge and previously collected data.
2) **Represent the dialogue system as a BAPOMDP.**
   a) For the unknown observation parameters, define Dirichlet counts with prior knowledge. If there is no prior knowledge, assume uniform initial priors.
3) **Using domain knowledge, look for symmetry in the observation parameters that are unknown.**
   a) Label the similar parameters.
   b) Write simple routines to perform parameter tying for the similar parameters. Whenever a parameter count is updated, the corresponding parameters (tied to it) should also be updated.
4) **Solve the BAPOMDP and learn the unknown observation parameters.**
   a) Whenever an action associated with the unknown observation parameters is picked, increment the Dirichlet counts accordingly.
   b) Perform belief tracking by choosing the K most probable belief states and normalizing them.
   c) Pick the best action to perform based on the new posterior, and repeat step 4.

IV. Empirical Evaluation

Our primary empirical goal is to show that by exploiting certain known components of the dialogue system, such as knowledge of symmetrical properties, and an approximate online planning algorithm (RTBSS), we are able to apply Bayesian RL on several simulated spoken dialogue system domains. We consider four different experimental domains: (1) a small synthetic data case, where we illustrate several properties of the approach; (2) a dialogue manager based on the well-known SACTI1 corpus; (3) a large dialogue manager aimed at assisting patients with dementia achieve effective hand washing, (4) a real-world dialogue manager designed to help patients to operate a wheelchair. Throughout, we discuss practical issues faced with using these techniques.

A. Empirical Methodology

In this empirical evaluation, each BAPOMDP simulation consists of 100 episodes, during which the agent must select actions and can use observations to improve its estimate of the model parameters. Each episode is a short dialogue sequence trial, which may be terminated by a specific action or when a prefixed number of actions is taken. At this point, the POMDP state (the user's intent) is reset, but the posterior distribution over the observation count vector is carried over to the next episode for the purposes of learning.

We measure the empirical returns of the policy, which corresponds to the total rewards achieved by the system under a test trajectory. We also consider the L1-distance, measured by $\sum_{s\in S} |b(s) - \hat{b}(s)|$, as an indication of the accuracy of the estimated model. The smaller the distance between the real belief and the estimated belief, the more accurate the model is. We also measure the Observations-L1-distance, measured by $\sum_{z\in Z} |O(s', a, z) - \hat{O}(s', a, z)|$, as an indication of the accuracy of the learnt observation parameters. The smaller the distance between the real observation parameters and the learnt observation parameters, the more accurate the model is. As both the L1-distance and the Observations-L1-distance vary depending on the episode, we take the maximum distance among all the episodes as the distance for the simulation. Each experiment is repeated for 1000 trials to assess the empirical mean of the estimates. For all the graphs reporting the empirical return, the standard errors were very large due to variance in the reward functions, so we do not show them. For all the graphs reporting the L1 error (on the model or on the belief), the standard errors over the 1000 trials were very small, and not visible on the plot.

We run most experiments using two different ways of updating the counts ψ for estimating the observation probability parameters. The first is the usual way of updating each parameter independently. The second approach makes use of symmetry with the parameters, as described in Section III.C.

B. Small Synthetic Dialogue Domain

This problem was introduced in [42] and it consists in a human operator instructing an assistive robot to move to one of two locations, bedroom or bathroom. Even though the human intent (the *state*) is one of these goals, the observation received by the robot through a speech recognizer is not accurate. The robot has the option to ask again to ensure the goal was understood correctly. In this model, we assume the probability of a wrong observation is 0.15. The reward is $+10$ for correctly identifying the goal, -100 for identifying the wrong goal, and -1 for asking again. This model is modeled on the classic Tiger problem [43], the main difference is that here we consider a finite horizon.

In the dialogue illustrating an human operator instructing an assistive robot, there are four unknown parameters (OEE, OEA, OAE, OAA) to learn.

$$OEE := \Pr(z = \text{bedroom}|s = \text{bedroom}, a = \text{ask}) = 0.85$$
$$OEA := \Pr(z = \text{bathroom}|s = \text{bedroom}, a = \text{ask}) = 0.15$$
$$OAE := \Pr(z = \text{bedroom}|s = \text{bathroom}, a = \text{ask}) = 0.15$$
$$OAA := \Pr(z = \text{bathroom}|s = \text{bathroom}, a = \text{ask}) = 0.85$$

Since $OEE = OAA$ and $OEA = OAE$, we can make use of symmetry in this model to perform parameter tying. Whenever the count for OEE is updated, OAA is also updated, and vice versa. Similarly, whenever the count for OAE is updated, OEA is also updated, and vice versa.

We allow planning time of 1 second per action, and at most 20 actions per episode. We consider two different priors, corresponding to $OEE = OAA = \{0.65, 0.80\}$. Experimental results are shown in Figs. 1 and 2. For the returns, as shown in Fig. 1, using a prior of 0.80 gives better returns than a prior of 0.65 at the beginning, but returns converge after about 10 episodes regardless of prior. We also note that using symmetry leads to faster convergence no matter whether we use a prior of 0.65 or 0.80. In Fig. 2, the L1-distance is smaller when we use a prior of 0.80, as compared to a prior of 0.65, but the L1-distance also converges after about 10 episodes. These empirical results confirm that using symmetry in the model leads to more efficient learning of the model parameters, even if the initial priors are more noisy. This suggests that for learning in a dialogue system, it might be more effective to look for symmetry in the model than to come up with more accurate initial priors.

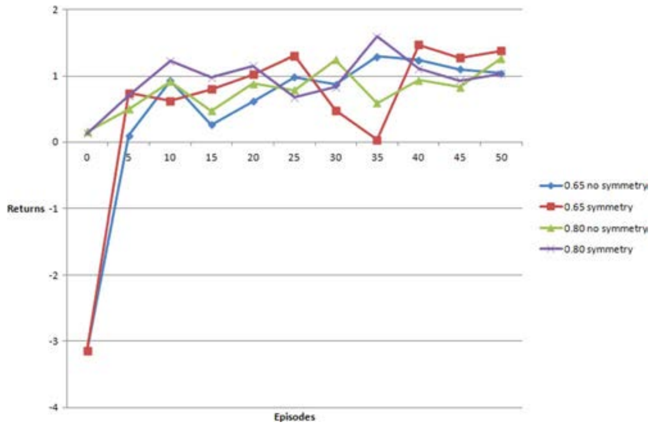

Fig. 1. Small POMDP Dialogue Manager: Returns against different priors with and without symmetry. The empirical return for the optimal policy is 1. 72. The empirical return when planning with the 0.65 prior and without further learning is -3.86. The empirical return when planning with the 0. 8 prior and without further learning is 0. 19. Standard errors for all lines in the plot are very large due to the high variance in the reward function.

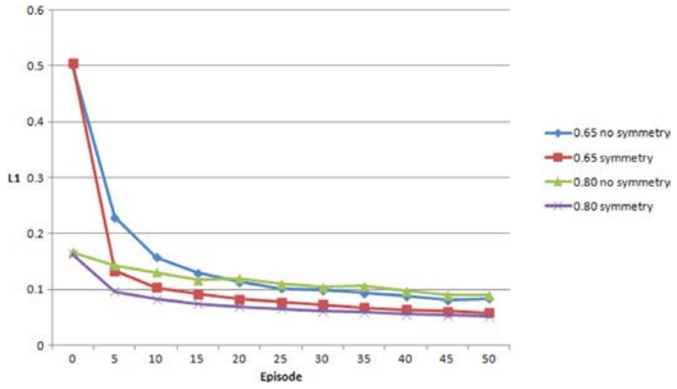

Fig. 2. Small POMDP Dialogue Manager: Belief L1-distance against different priors with and without symmetry. Standard errors for all lines in the plot are too small to be visible.

C. SACTI Dialogue Domain

This second domain is drawn from the dialogue literature [44] and is based on real-world dialogue recordings. It contains 144 dialogues between 36 users and 12 experts (who have the role of a dialogue manager), covering 24 tasks[2] The utterances from users to human experts are first confused using a speech recognition error simulator. Hidden Topic Markov Models (HTMM) was used to design the dialogue POMDP [45]. The generated POMDP contains 5 states, 14 actions, and 5 meta observations drawn by the HTMM using 817 primitive observations (words). In this model, users can have 3 different intentions. They represent the machine states: *transportation, visiting area*, and *food*. There are also 2 other states: *success* and *failure*, depending on whether the dialogue finished successfully or not. The reward is $+50$ if the conversation terminates in the *success* state and -50 if the conversation ends in *failure*; reward for other states is -1.

[2]*http://mi.eng.cam.ac.uk/projects/sacti/corpora*

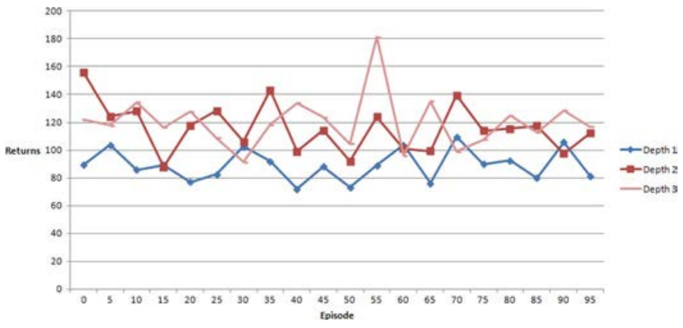

Fig. 3. Sacti1: Returns against different depths. The empirical return for the optimal policy is $174 + / - 15$. The empirical return for the prior, without further learning appears at the $x = 0$ point. The standard error on all lines is very large due to the variance in the reward.

The 14 actions include *inform, request, greeting farewell, request repeat*, and others. We assume that we do not know the observation parameters for the action *inform*, and we want to learn them. For this problem, we have a planning time of 1 second per action, and a maximum of 20 actions per episode.

Our objective here is to investigate the effects of the depth of the online search on the returns obtained. In solving a POMDP, we usually obtain better returns as we increase the depth d of RTBSS. Increasing depth leads to better actions being chosen because we take into account further actions in the future. However, in a BAPOMDP model, our search tree is a lot larger than a similar POMDP problem because the BAPOMDP model takes into account the Dirichlet counts. Solving a BAPOMDP exactly for all belief states is impossible in most domains due to the dimensionality of the state space. In theory, the number of possible Dirichlet count vectors can grow exponentially with the planning horizon. In practice, we maintain a constant size by keeping only the K most probably beliefs. It is interesting for us to observe if increasing the depth of the online search will lead to better returns in a BAPOMDP.

In our experiments, we consider a depth of 1 to 3; at depth 4 and beyond, the computations are too time-consuming. We note some interesting experimental results. In Fig. 3 we observe that using an online search of depth 2 and 3 seems to gives slightly better returns than using depth 1. Increasing the search beyond depth 2 does not give any obvious improvements. This suggests that this particular problem can be solved with a relatively short planning horizon.

In Fig. 4, we observe that for the L1-distance, using an online search of depth 1 gives drastically poorer results as compared to using depth 2 or 3. This difference is actually due to the actions selected by the search with depth 1. When we use an online search of depth 1, our search always returns action *inform*, which is the action with observation parameters we want to learn. Due to the noise in these parameters, the L1-distance on the belief increases drastically if we perform action *inform* right at the beginning. As we take the maximum L1-distance on the belief among all the episodes, the error observed at the very beginning dominates. However, when we perform a search of depth 2 or depth 3, the search tends to give us actions other than *inform*. Since these actions yield observations with little noise, our L1-distance on the belief decreases, and they do not increase much after that. This results in the large disparity shown in Fig. 4.

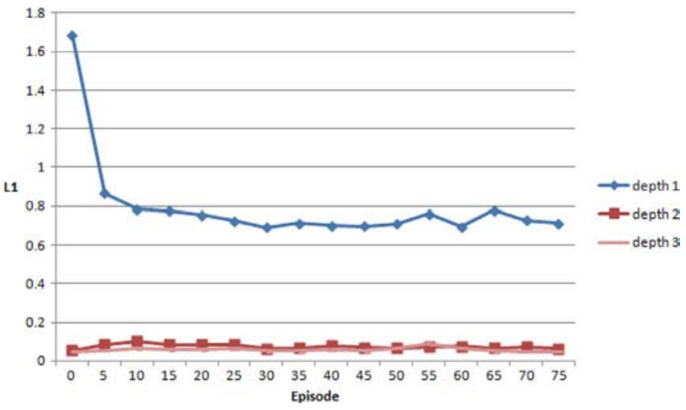

Fig. 4. Sacti1: Belief L1-distance against different depths. The standard error on all lines is too small to be visible on this plot.

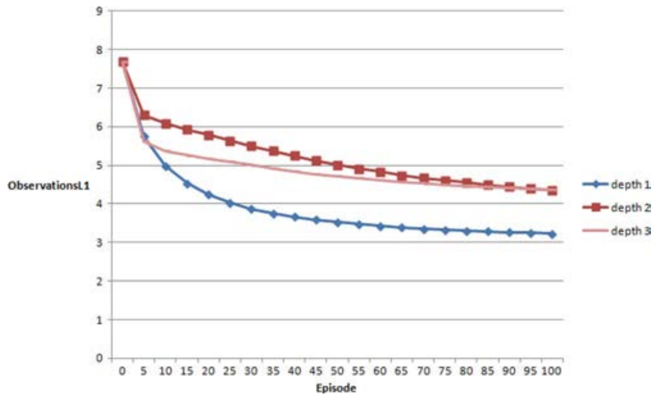

Fig. 5. Sacti1: Observations-L1-distance against different depths.

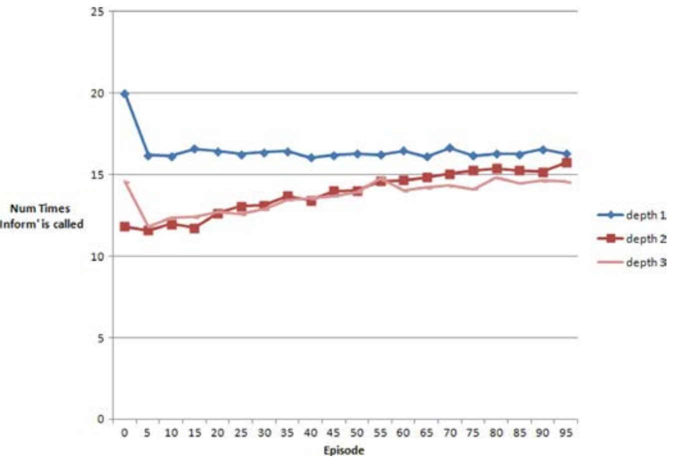

Fig. 6. Sacti1: Number of times "inform" action was performed against different depths.

It is also interesting to note that even though using a search of depth 1 gives us poor returns and L1-distance on the belief, it actually gives better Observations-L1-distance as seen in Fig. 5. Upon further investigations, we realise this is also due to the number of times the *inform* action is performed. As observed in Fig. 6, using an online search of depth 1 leads to performing the *inform* action many more times than using depth 2 and depth 3. These results are consistent because performing the *inform* action more times will lead to better learning of its observation parameters. In Fig. 7, we show how the Observation-L1-distance

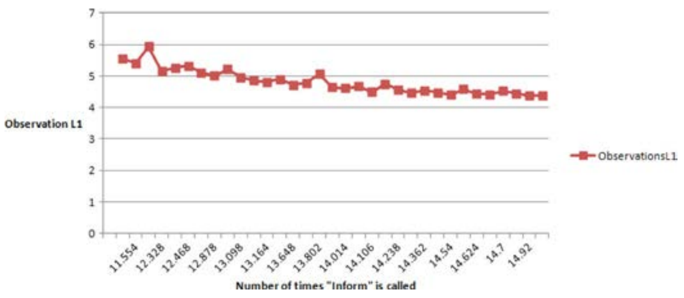

Fig. 7. Sacti1: Observation-L1-distance against the number of times the "inform" action was performed.

varies according to the number of times the *inform* action was performed. These results suggest that using the Belief L1-distance is a better way to judge how well we learn a model than using the Observations-L1-distance. The improvements in Belief L1-distance correspond to the improvements in the returns, as the results indicate that depth 2 and depth 3 give better returns than depth 1. Improvements in Observations-L1-distance, however, are mainly due to the action (with observation parameters we want to learn) being chosen more frequently.

D. Handwashing Dialogue Domain

This problem was introduced in [46] and consists of assisting a user suffering from dementia through tasks of daily living via the help of a dialogue manager. The POMDP parameterization (hand-coded by experts) can be found online[3]. The domain has 180 states, 6 observations, and 6 actions. The reward function is complicated and detailed in the link.

The goal of the dialogue manager is to offer assistance to the user in the form of task guidance such as prompts or reminders when he attempts a HandWashing task. There are 6 actions, which correspond to instructions to be provided to the user: {*nothing, wet hand, turn on water, turn off water, use soap, dry hands*}. The user's state can be factored into state variables, including the plan's step, the user's responsiveness and awareness levels, and what the user has just done. The 6 observations correspond to the position of the user's hand (*away, sink, water, tap, soap, towel*). There are a total of $6 \times 180 = 1080$ observation parameters to learn. However even though there are a larger number of observation parameters, many of them are actually similar. This number has been reduced to 16 parameters, using symmetry (see [47] for more detail on the structure of the symmetry constraints).

In this domain we assume no knowledge of the initial observation parameters in this domain. We generate numbers from a random number generator for each of the 1080 observation parameters. For each row of observation parameters, we then normalise them so that they sum up to 1. Then, we compare the effects of using symmetry to update the counts. We fixed the search depth to $d = 2$, we also capped the planning time at 6 seconds per action. We allowed 20 actions per episode.

The first result, presented in Table I, is a rather unexpected one: when we measure the return of the learnt model, we find that the return does not improve with more training data, whether or not we use parameter tying (symmetry) to accelerate learning. Upon further investigation, we discovered that solving the correct model as a POMDP does not give us higher returns than solving the same model but replacing the original observation parameters with randomly generated probabilities. This suggests that the observation parameters do not change the optimal policy, and so the improvement in return is not an useful metric for our learning algorithm.

Considering instead the Observations-L1-Distance, we see in Fig. 8 that error decreases somewhat when using parameter tying (symmetry), but not without. This suggests two things. First, parameter tying, as expected, can improve learning in terms of parameter error rate. Second, since there's no benefit to be gained in terms of the return, the learning algorithm does not need to expend resources to learn a better model. As such, this is an interesting case for showcasing the advantages of decision-theoretical/reinforcement learning approaches for parameter learning in dialogue managers.

TABLE I
COMPARING RETURNS OF THE ACTUAL MODEL WITH DIFFERENT OBSERVATION PARAMETERS. C.I., MEANS CONFIDENTIAL INTERVAL

Type	Returns	95% C.I.
MDP	12.42	(10.97,13.87)
BAPOMDP with symmetry	15.55	(13.99, 17.11)
POMDP with known parameters	16.45	(14.85,18.05)
POMDP w/randomly generated obs. params	16.87	(15.27,18.46)

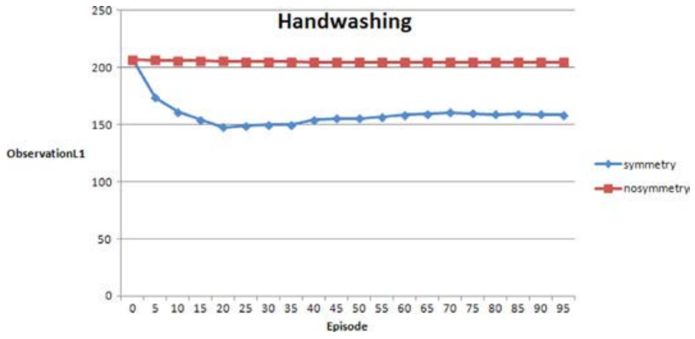

Fig. 8. HandWashing: Observations-L1-distance against models with and without symmetry.

E. SmartWheeler Dialogue Domain

The SmartWheeler dialogue domain is a POMDP model used onboard a smart wheelchair for dialogue management between the user and the onboard computer [48]. Input voice commands, acquired via a commercial speech recognition system, are processed through basic semantic parsing, and then handled as observations by the dialogue manager. Responses from the dialogue manager are presented through either a visual interface or text-to-speech system.

The domain considered here is a modification of the POMDP model described in [49]. The precise problem description can be downloaded online[4]. In this domain, the user has an unknown intent, and the robot has to execute an action based on its guess of the user's intent. When it has identified the user's intent, the

[3]*http://www.cs.uwaterloo.ca/~ppoupart/software/symbolicPerseus/problems/handwashing/cppo3.txt*

[4]*http://www.cs.mcgill.ca/~smartwheeler/data/wheelchairdialogue25.POMDP*

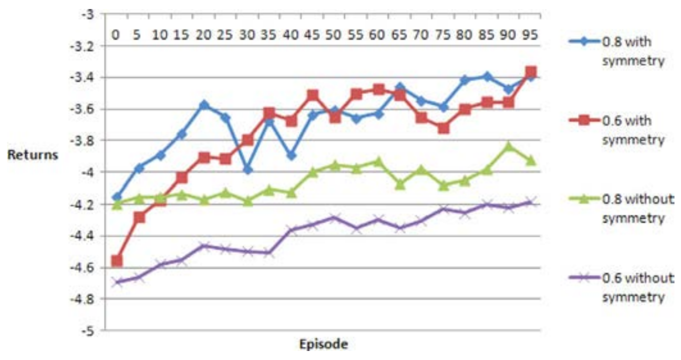

Fig. 9. SmartWheeler: Returns against different priors with and without symmetry. The empirical return for planning with the correct parameters is -0.25. The empirical return for planning with the prior at 0. 6 is -4.553. The empirical return for planning with the prior at 0. 8 is -4.195.

robot can execute a command action, receiving a positive reward if correct, and zero reward otherwise. There are 25 states, 29 actions, and 25 observations. Each state corresponds to a possible user intent, such as *"drive one meter forward"* or *"set speed to fast"*. There is one observation corresponding to each state, however observations are not fully accurate, and are at best an indication of the user's intent. Observations are handled dynamically when received. If the phrase was parsed successfully by the semantic parser, the belief state is updated using statistics based on the semantic assignment. If parsing failed, the phrase is treated as a bag-of-words [50]. The observation parameters for the simulator were obtained through manual labelling of trials with real users.

In a typical dialogue system, many parameters are likely to be similar. For instance, the phrase *"drive slowly backward"* is similar to *"drive slowly forward"*, but is very different from *"avoid obstacle"*. Using the knowledge that certain parameters have similar values, we can learn the parameters in a faster manner. There are $25 \times 25 = 625$ unknown observation parameters to learn. After reducing this number via symmetry, we focused on estimating only 6 hyper-parameters [47].

For each state, there is a corresponding "correct" action that can be executed by the wheelchair. Additionally, there are four query actions. There is a general query action that requests a phrase to be repeated, and three other action-specific queries which request clarification for a particular set of actions. The reward is 0. 1 when executing the correct action, -0.3 for selecting a clarification question, and -0.9 for picking another action that does not match the user's intent.

Our experiment aims to evaluate the performance of the BAPOMDP approach under different conditions in the context of SmartWheeler. To this end, we fixed the search depth to $d = 2$, capped the planning time at 3 seconds per action, and the horizon at 20 actions per episode. First, we investigate the effects of having different priors for the observation parameters. Then, we measure the impact of using symmetry to update the counts. More specifically, we consider different priors ranging from 0.5 to 0.9, knowing that the observation parameter has the true value $x = 0.97$.

First we consider the empirical return, shown in Fig. 9, using a prior of 0.60 vs 0.80. We do notice that using the more accurate prior $(x_0 = 0.80)$ gives better returns (initially when using symmetry, and through the experiment when not using symmetry). As expected, using symmetry leads to faster convergence no matter whether we use a prior of 0.60 or 0.80. Using an inaccurate prior of 0.60 with symmetry actually leads to a faster convergence than using a more accurate prior of 0.80 without symmetry. We note however that it seems that we have not achieved full learning, since the empirical return for the known parameters is much better than with learning, and lines in Fig. 9 have not converged.

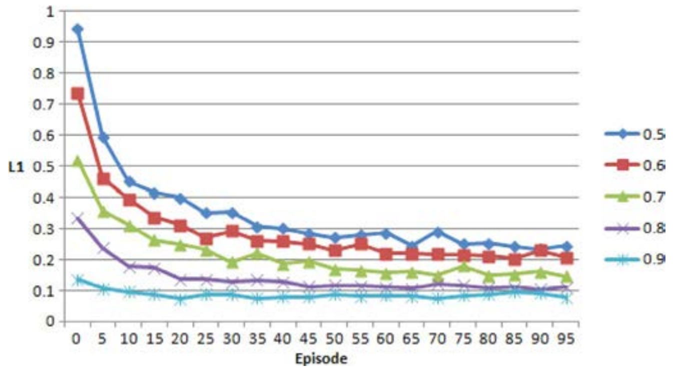

Fig. 10. SmartWheeler: L1-distance against different priors with symmetry.

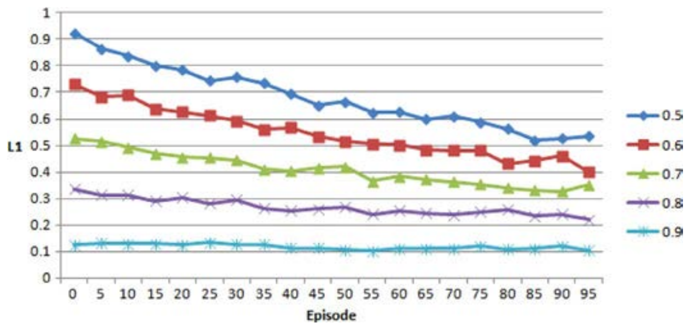

Fig. 11. SmartWheeler: L1-distance against different priors without symmetry.

Using symmetry to update the observation counts also results in a faster convergence to the correct model as shown in Figs. 10–12. In Figs. 10 and 11, the initial L1-distance is smaller as the prior is closer to the true value of 0.97, but the L1-distance converges faster if we use symmetry in the model. In Fig. 12, we observe that using symmetry leads to a faster convergence no matter which priors are used. These empirical results suggest that prior information on the *structure* of the observation parameters is more important than having accurate information about the specific values. Nonetheless we note that evidence of learning is observed even without symmetry, when there are 625 parameters to estimate.

V. Discussion

In this paper, we described a Bayes-adaptive POMDP framework for simultaneous learning and planning for robust dialogue management. The framework is mathematically sound, and the algorithms are tractable on realistic domains, especially when leveraging structural information about the domain, and using fast online planning techniques. Even though knowledge engineering in terms of defining the states, actions, priors and rewards, is still a challenge, this may be applicable

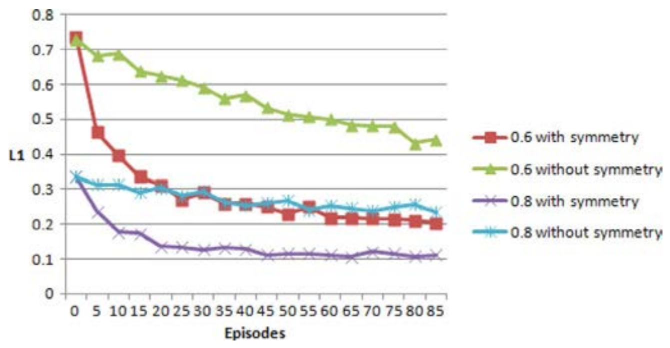

Fig. 12. SmartWheeler: Belief L1-distance against different priors with and without symmetry.

in many practical domains. We demonstrated the benefits of our approach on four different simulated dialogue systems, including two human-robot interaction tasks.

In Section IV.C, we noted that the depth of the search may not affect the returns. We should further investigate this point and see what is really the impact of the depth d (of RTBSS) on the returns. We also found, in some domains, that using the Belief L1-distance is a better way to judge how well we learn a model than using the Observations-L1-distance.

In addition, in Sections IV.C and IV.E, we showed that using symmetry in the model is very efficient in terms of learning the model parameters. Even when initial priors are more noisy, using parameters tying leads: (i) to a faster convergence of the model parameters and, (ii) to a faster convergence in both the returns and the L1-distance. For learning in a dialogue system, it might be more effective to look for symmetry in the model than to come up with more accurate initial priors. It is worth noting that symmetry can in fact be interpreted as a structural prior on the domain. There has been some work showing that learning the structural prior can be an effective way of accelerating learning [51]. Extension of these results to problems of dialogue management could lead to interesting solutions for large structured dialogues.

One limitation of our analysis is the lack of comparison to other learning frameworks for dialogue systems (e.g., those referenced in Section II.D). Comparing to algorithms that make different assumptions (e.g., EM, supervised learning) would yield an unfair comparison, which is why we did not provide this analysis. In terms of the online planning component, an extensive evaluation of the performance and computation time of such algorithms is provided in [7].

Another limitation of our analysis is the fact that all the results presented above applied the BAPOMDP framework on simulated models, instead of real deployed dialogue systems. There are practical issues which need to be further investigated. For example, the planning time in our experiments is on the order of a few seconds per action, which is possibly too slow for most real systems. But with current development of technology, we expect the BAPOMDP model to be computationally fast enough for deployment on realistic dialogue systems in the very near future. In the mean time, user studies with small to mid-size domains should be pursued.

References

[1] S. Young, “Cognitive user interfaces,” *IEEE Signal Process. Mag.*, vol. 27, no. 3, pp. 128–140, May 2010.
[2] D. Kim, J. Kim, and K. Kim, “Robust performance evaluation of POMDP-based dialogue systems,” *IEEE Trans. Audio, Speech, Lang. Process.*, no. 99, 2011.
[3] J. Williams and S. Young, “Partially observable Markov decision processes for spoken dialog systems,” *Comput. Speech Lang.*, vol. 21, no. 2, 2007.
[4] J. Williams, P. Poupart, and S. Young, “Partially observable Markov decision processes with continuous observations for dialogue management,” *Recent Trends in Discourse and Dialogue*, pp. 191–217, 2008.
[5] S. Png and J. Pineau, “Bayesian reinforcement learning for POMDP-Based dialogue systems,” in *ICASSP*, 2011.
[6] S. Ross, B. Chaib-draa, and J. Pineau, “Bayes-adaptive POMDPs,” in *Proc. Neural Inf. Process. Syst.*, 2007.
[7] S. Ross, J. Pineau, B. Chaib-Draa, and P. Kreitmann, “A Bayesian approach for learning and planning in partially observable Markov decision processes,” *J. Mach. Learn. Res.*, vol. 12, pp. 1729–1770, 2011.
[8] S. Young, “Probabilistic methods in spoken dialogue systems,” *Philosoph. Trans. R. Soc. London. Ser. A: Math. , Phys. Eng. Sci.*, vol. 358, no. 1769, pp. 1389–1402, 2000.
[9] M. White, OpenCCT: The OpenNLP CCG Library, [Online]. Available: http://openccg.sourceforge.net/
[10] E. Levin and R. Pieraccini, “A stochastic model of computer-human interaction for learning dialogue strategies,” in *Proc. Eur. Conf. Speech Commun. Technol.*, 1997.
[11] E. Levin, R. Pieraccini, and W. Eckert, “A stochastic model of human-machine interaction for learning dialog strategies,” *IEEE Trans. Speech Audio Process.*, vol. 8, no. 1, pp. 11–23, Jan. 2000.
[12] D. Goddeau and J. Pineau, “Fast reinforcement learning of dialog strategies,” in *Proc. ICASSP*, 2000, pp. 1233–1236.
[13] M. Walker, “An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email,” *J. Artif. Intell. Res.*, vol. 12, no. 1, pp. 387–416, 2000.
[14] K. Scheffler and S. Young, “Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning,” in *Proc. Int. Conf. Human Lang. Technol. Res.*, 2002.
[15] S. Singh, D. Litman, M. Kearns, and M. Walker, “Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system,” *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 105–133, 2002.
[16] M. Denecke, K. Dohsaka, and M. Nakano, “Learning dialogue policies using state aggregation in reinforcement learning,” in *Proc. Int. Conf. Spoken Lang. Process.*, 2004.
[17] N. Roy, J. Pineau, and S. Thrun, “Spoken dialogue management using probabilistic reasoning,” in *Assoc. Comput. Linguist.*, 2000.
[18] F. Doshi and N. Roy, “Efficient model learning for dialog management,” in *Proc. Int. Conf. Human-Robot Interact.*, 2007.
[19] K. Astrom, “Optimal control of Markov decision processes with incomplete state estimation,” *J. Math. Anal. Applicat.*, vol. 10, no. 1, pp. 174–205, 1965.
[20] R. Smallwood and E. Sondik, “The optimal control of partially observable Markov processes over a finite horizon,” *Operat. Res.*, vol. 21, no. 5, pp. 1071–1088, 1973.
[21] L. Kaelbling, M. Littman, and A. Cassandra, “Planning and acting in partially observable stochastic domains,” *Artif. Intell.*, vol. 101, no. 1–2, pp. 99–134, 1998.
[22] J. Pineau, G. Gordon, and S. Thrun, “Anytime point-based approximations for large POMDPs,” *J. Artif. Intell. Res.*, vol. 27, pp. 335–380, 2006.
[23] M. Spaan and N. Vlassis, “Perseus: Randomized point-based value iteration for POMDPs,” *J. Artif. Intell. Res.*, vol. 24, no. 1, pp. 195–220, 2005.
[24] D. Hsu, W. Lee, and N. Rong, “A point-based POMDP planner for target tracking,” in *Proc. ICRA*, 2008.
[25] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, “Online planning algorithms for POMDPs,” *J. Artif. Intell. Res.*, vol. 32, no. 1, pp. 663–704, 2008.
[26] G. Shani, R. Brafman, and S. Shimony, “Model-based online learning of POMDPs,” in *Proc. Eur. Conf. Mach. Learn.*, 2005.
[27] D. Bertsekas and D. Castanon, “Rollout algorithms for stochastic scheduling problems,” *J. Heuristics*, vol. 5, no. 1, pp. 89–108, 1999.
[28] S. Paquet, L. Tobin, and B. Chaib-Draa, “Real-time decision making for large POMDPs,” in *Proc. Adv. Artif. Intell.*, 2005.

[29] S. Paquet, "Distributed decision-making and task coordination in dynamic, uncertain and real-time multiagent environments," Ph.D. dissertation, Univ. Laval, Quebec, Canada, 2006.

[30] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Proc. Neural Inf. Process. Syst.*, 2010.

[31] H. Cuayáhuitl, S. Renals, O. Lemon, and H. Shimodaira, "Evaluation of a hierarchical reinforcement learning spoken dialogue system," *Comput. Speech Lang.*, vol. 24, no. 2, 2010.

[32] J. Henderson, O. Lemon, and K. Georgila, "Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets," *Comput. Linguist.*, vol. 34, no. 4, pp. 487–511, 2008.

[33] F. Jurvčíček, B. Thomson, and S. Young, "Natural actor and belief critic: Reinforcement algorithm for learning parameters of dialogue systems modelled as POMDPs," *ACM Trans. Speech Lang. Process.*, vol. 7, no. 3, 2011.

[34] F. Jurvčíček, B. Thomson, and S. Young, "Reinforcement learning for parameter estimation in statistical spoken dialogue systems," *Comput. Speech Lang.*, vol. 26, no. 3, pp. 127–228, 2012.

[35] O. Pietquin, M. Geist, S. Chandramohan, and H. Frezza-Buet, "Sample-efficient batch reinforcement learning for dialogue management optimization," *ACM Trans. Speech Lang. Process.*, vol. 7, no. 3, 2011.

[36] F. Doshi and N. Roy, "Spoken language interaction with model uncertainty: An adaptive human-robot interaction system," *Connect. Sci.*, pp. 299–318, 2008.

[37] B. Thomson, F. Jurcicek, M. Gasic, S. Keizer, F. Mairesse, K. Yi, and S. Young, "Parameter learning for POMDP spoken dialogue models," in *Proc. SLT*, 2010.

[38] B. Boots, S. Siddiqi, and G. Gordon, "Closing the learning-planning loop with predictive state representations," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 954–966, 2011.

[39] M. Duff, "Computational procedures for optimal learning," Ph.D. dissertation, , Univ. of Mass., Amherst, 2002.

[40] R. Dearden, N. Friedman, and D. Andre, "Model based Bayesian exploration," in *Proc. Uncertainty in Artif. Intell.*, 1999.

[41] P. Poupart, N. Vlassis, J. Hoey, and K. Regan, "An analytic solution to discrete Bayesian reinforcement learning," in *Proc. ICML*, 2006.

[42] M. M. Fard, J. Pineau, and P. Sun, "A variance analysis for POMDP policy evaluation," in *Proc. AAAI*, 2008.

[43] A. Cassandra, L. Kaelbling, and M. Littman, "Acting optimally in partially observable stochastic domains," in *Proc. National Conf. Artif. Intell.*, 1995.

[44] J. Williams and S. Young, The SACTI-1 Corpus: Guide for Research Users, 2005.

[45] A. Boularias, H. Chinaei, and B. Chaib-Draa, "Learning the reward model of dialogue POMDPs from data," in *Proc. NIPS Workshop Mach. Learn. for Assist. Tech.*, 2010.

[46] J. Boger, P. Poupart, J. Hoey, C. Boutilier, G. Fernie, and A. Mihailidis, "A decision-theoretic approach to task assistance for persons with dementia," in *Proc. IJCAI*, 2005.

[47] S. Png, "Bayesian reinforcement learning for POMDP-based dialogue systems," M.S. thesis, , McGill Univ., Montreal, QC, Canada, 2011.

[48] A. Atrash, R. Kaplow, J. Villemure, R. West, H. Yamani, and J. Pineau, "Development and validation of a robust interface for improved human-robot interaction," *Int. J. of Social Robot.*, pp. 345–356, 2009.

[49] A. Atrash and J. Pineau, "A Bayesian reinforcement learning approach for customizing human-robot interfaces," in *Proc. IUI*, 2009.

[50] A. Atrash, "A Bayesian framework for online parameter learning in POMDPs," Ph.D. dissertation, McGill Univ., Montreal, QC, Canada, 2011.

[51] S. Ross and J. Pineau, "Model-based Bayesian reinfrocement learning in large structured domains," in *Proc. UAI*, 2008.

Shaowei Png is a Software Engineer at Google. He holds a Bachelor of Computer Science at the National University of Singapore, and an M.Sc. degree from the School of Computer Science at McGill University (2011).

Joelle Pineau is an Associate Professor at the School of Computer Science at McGill University, where she co-directs the Reasoning and Learning Lab. She received her Ph.D. in robotics from Carnegie Mellon University in 2004. Her research centers on developing efficient algorithms for planning and learning in partially observable stochastic domains. She is on the editorial board of the Journal of Machine Learning Research and the Journal of Artificial Intelligence Research.

Brahim Chaib-draa (SM'03) received the Diploma degree in computer engineering from the Ecole supérieure d'Electricité, Paris, France, in 1978 and the Ph.D. degree in computer science from the Université du Hainaut-Cambrésis, Valenciennes, France, in 1990. In 1990, he joined the Department of Computer Science and Software Engineering, Laval University, Quebec City, QC, Canada, where he is currently a Professor and the Group Leader of the Decision for Agents and Multiagent Systems (DAMAS) Group. He is the author or a coauthor of several technical publications on reasoning under uncertainty, machine learning, and multiagent systems. He is on the Editorial Boards of Computational Intelligence and IEEE-SMC and is a member of the ACM, AAAI and Senior Member of the IEEE Computer Society.