Scaling Up POMDPs for Dialog Management: The “Summary POMDP” Method

Jason D. Williams and Steve Young

Cambridge University Engineering Department
Trumpington Street, Cambridge CB2 1PZ, UK
jdw30@cam.ac.uk sjy@eng.cam.ac.uk

ABSTRACT

Partially Observable Markov Decision Processes (POMDPs) have been shown to be a promising framework for dialog management in spoken dialog systems. However, to date, POMDPs have been limited to artificially small tasks. In this work, we present a novel method called a “Summary POMDP” for scaling slot-filling POMDP-based dialog managers to cope with tasks of a realistic size. An example dialog problem incorporating a user model built from real dialog data is presented. A dialog manager is created using this method and evaluated using a second user model created from held-out dialog data. Results confirm that summary POMDP policies scale well, and also show that summary POMDP policies are reasonably robust to variations in user behavior.

1. INTRODUCTION

Dialog management for spoken dialog systems can be approached as planning under uncertainty. (Fully-observable) Markov decision processes ((FO)MDPs) and partially observable Markov decision processes (POMDPs) offer a principled framework in this pursuit [5]. MDPs offer an accessible body of optimization techniques and have been successfully used to create dialog managers for problems of realistic sizes [4, 6, 9, 10]. However, MDPs do not take proper account of the corruption introduced by the speech recognition channel. The POMDP framework, which is designed to cope with noisy input, has been found to consistently outperform MDP-based systems [8, 14, 15, 16]. Despite their promise, POMDPs to date have been limited to artificially small “toy” problems. Systems in the literature handle fewer than 10 distinct user goals [8, 14, 15, 15]. In this paper, we present a method for scaling up POMDPs for dialog management – the “Summary POMDP method” – and demonstrate its ability to create a robust slot-based dialog manager capable of handling a realistically sized dialog problem.

This paper is organized as follows: Section 2 briefly reviews background on POMDPs. Section 3 presents the summary POMDP method. Section 4 briefly demonstrates the method, Section 5 provides an evaluation, and Section 6 concludes.

2. OVERVIEW OF POMDPS

Formally, a POMDP is defined as a tuple *{S, A, T, R, O, Z}*, where *S* is a set of states; *A* is a set of actions that an agent may take; *T* defines a transition probability $P(s' \mid s,a)$; *R* defines the expected (immediate, real-valued) reward $r(s,a)$; *O* is a set of observations; and *Z* defines an observation probability, $P(o' \mid s',a)$.

The POMDP operates as follows. At each time-step, the machine is in some unobserved state s. Since s is not known exactly, we maintain a distribution over states called a “belief state,” *b*. We write $b(s)$ to indicate the probability of being in a particular state *s*. Based on *b(s),* the machine selects an action a, receives a reward r, and transitions to (unobserved) state s', where s' depends only on s and a. The machine then receives an observation o' which is dependent on s' and a. At each timestep, we update *b* as follows:

$$b'(s') = k \cdot P(o' \mid s',a) \sum_{s \in S} P(s' \mid a,s) b(s) , \tag{1}$$

where *k* is a normalization constant [2]. We refer to maintaining the value of *b* at each time-step as “belief monitoring.”

The variables *a, o,* and *s* may be factored into components. For example, *s* could be decomposed into $s = (s_1, s_2, \ldots, s_N)$ such that $s_1 \in S_1, s_2 \in S_2, \ldots, s_N \in S_N$. In this case, *b(s)* is written $b(s) = b(s_1, s_2, \ldots, s_N)$, $P(o' \mid s',a)$ is written $P(o' \mid s_1', s_2', \ldots, s_N', a)$, etc. This factoring may allow conditional independence to be exploited, reducing the parameters required to express *T*, *R*, and *Z*.

The cumulative, infinite-horizon, discounted reward is called the *return*:

$$\sum_{t=0}^{\infty} \lambda^t \sum_{s \in S} b_t(s) r(s, a_t) \tag{2}$$

where b_t indicates the distribution over all states at time *t*; $b_t(s)$ indicates the probability of being in state *s* at timestep *t*; a_t is the action *a* taken at time *t*; and λ is a geometric discount factor, $0 \le \lambda \le 1$. A *policy* $\pi(b) \in A$ specifies an action to take given a belief state.[1] The goal of the planner is to find a policy that maximizes the return.

The domain *b* of a policy $\pi(b)$ is a point in *S*-dimensional space, called *belief space.* In general a policy is a partitioning of belief space, where each partition corresponds to an action. In fact, the size of the policy space grows exponentially with the size of the observation set and doubly exponentially with the

[1] We will assume the planning horizon for a policy is infinite unless otherwise stated.

distance (in time-steps) from the horizon [2]. Nevertheless, real-world problems often possess small policies of high quality.

In this work, POMDP optimization is performed with *Perseus* [11]. *Perseus* heuristically selects a small set of representative belief *points*, and then iteratively applies value updates to just those points instead of all of belief space, achieving a significant speed-up over exact method. Increasing the number of belief points enables the algorithm more complex policies at the expense of more computation.

3. THE "SUMMARY POMDP" METHOD

This paper is concerned with so-called "slot-filling" dialogs, which are common in the spoken dialog system literature. In slot-filling dialogs, there exist N slots, where slot *i* takes on one of N_i values. The user enters the dialog with a *goal* – i.e., a desired value for each slot – and the aim of the machine is to correctly *submit* the user's goal.

Traditional POMDP-based dialog managers scale poorly because their state space, action set, and observation set grow as $\Pi\ N_i$. For example, consider a problem with two slots in which each slot takes on one of $N_1 = N_2 = M = 1000$ values. In this case, the POMDP is performing planning on a distribution over $M^2 = 10^6$ states with more than 10^6 actions and observations. Even with recent techniques, creating policies on POMDPs of this size is intractable.

By contrast, the summary POMDP method considers only the most likely hypothesis for each slot, effectively reducing the size of each slot to 2 values. The action and observation sets are similarly reduced. Overall this approach reduces the growth factor from M^{N} to 2^{N}.

The summary POMDP method consists of three phases: construction, sampling & optimization, and execution. As the method is explained, an example dialog problem is presented for illustration.

3.1. Construction of the "master POMDP"

First, a POMDP called the *master* POMDP is constructed. The state variable of the master POMDP is factored into three groups of variables which express the user's goal, the user's action, and the state of the dialog *from the perspective of the user*. Note that, from the machine's perspective, all of these variables are hidden. This factoring facilitates estimating the transition and observation functions and specifying the reward function [14].

The method requires that the state space component for the user's goal is further subdivided into N slots, with one component for each slot, $S_1^{slot} \ldots S_N^{slot}$. The components for the user's action and dialog state may, if desired, be further decomposed.

An illustrative master POMDP in the travel domain is shown as an influence diagram in Figure 1, with components described in Table 1. In this dialog problem, the user is trying to travel from a *from* location to a *to* location in a world with 1000 places. Thus the user's goal includes 2 components representing 2 slots, S_{from}^{slot} and S_{to}^{slot}, each of which takes on 1 of 1000 values. The model of user's action has been decomposed into the components S_i^A, S_i^{eq}, S_i^{WH}, S_i^{YN}, S_i^{WH2}, and S_i^{WH1}. Two conditional probability tables form the core of the user model. The distribution $P(s_i^{WH} \mid s_i^A)$ gives the probability that the user provides various *wh*-responses (i.e., ways of stating a slot value) such as *london* or *to(london)* for a given system action. The distribution $P(s_i^{YN} \mid s_i^A, s_i^{eq})$ gives the probability that the user includes a *yn*-response (*yes* or *no*) in their utterance for a given system action. The other user action components (S_i^A, S_i^{eq}, S_i^{WH2}, and S_i^{WH1}) are deterministic functions and simply enable S_i^{WH} and S_i^{YN} to be expressed succinctly. Finally, the node S_i^{dlg} expresses dialog state, i.e., whether slot *i* is, from the standpoint of the user, *not-stated, stated,* or *confirmed.*

In the summary POMDP method, the action set is also factored, into $a = (a^{force}, a_1^{slot}, \ldots, a_N^{slot})$, such that $a_i^{slot} \in S_i^{slot}$ and $a^{force} \in A^{force}$. A^{force} represents the set of locutionary forces available to the system. In Figure 1, the system action is formed $a = (a^{force}, a_{from}^{slot}, a_{to}^{slot})$. A^{force} contains 5 values: *ask-from, confirm-from, ask-to, confirm-to, submit.* Composite system actions express include, for example, *confirm-to(london), submit(boston,london)*, or *ask-from.*

The observation set *O* represents the output of the speech recognition and parsing process, and includes all possible user utterances at the concept level. In the example dialog problem, the observation set includes all combinations of user actions (as observed by the parser). For example, one observation might be the parse {*yes, place(london), to(place(boston))*}. In general and as in the example, the observation set will be too large to estimate a conditional probability table. Instead, we create a function $f(o, p_{err}) \rightarrow P(o \mid s)$ which estimates $P(o \mid s)$ for a given *o* and per-concept error rate p_{err}. In the example problem, the error rate p_{err} specifies the likelihood that a single concept is misrecognized (i.e., substituted or deleted).

The reward function is specified by the system designer. In the example dialog problem, the reward function encourages the system to correctly identify the user's goal as quickly as possible while observing conversational norms. For actions which do not end the dialog, per-turn penalties are assigned which seek to reward "appropriate" behavior, as listed in Table 2. For example, a higher per-turn penalty is given for confirming a slot that hasn't been stated than for confirming one that has. The *submit* action, which is the only action that ends the dialog, assigns +50 if the user's goal is correctly identified and -50 if not.

The process of belief monitoring involves inferring *b(s)* based on the observation *o* as in Eq. 1. Initially, *b(s)* is a flat distribution but as the dialog proceeds, it sharpens around the most likely values of s_i^{slot}. The provision of an accurate user action model and observation function greatly facilitates this process.

3.2. Construction of the summary POMDP

Whereas traditional methods would attempt to optimize the master POMDP directly (e.g., [14]), here we form a second POMDP called the *summary* POMDP in which optimization will be conducted.

The state space of the summary POMDP, $\tilde{S}$, contains a component $\tilde{S}_i^{slot}$ corresponding to each slot component S_i^{slot} in

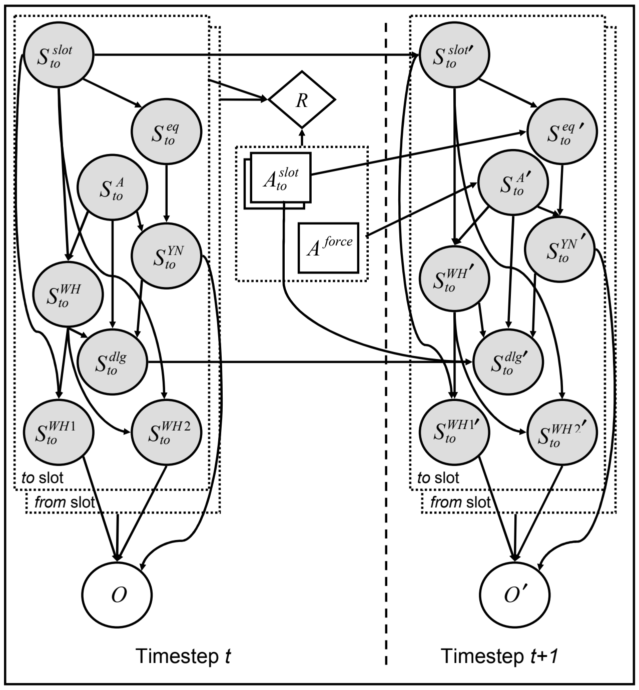


Figure 1: Influence diagram showing the master POMDP for the sample dialog problem, following [1]. Unshaded nodes are observable, and shaded nodes are unobservable. Circles are chance nodes; squares are decision nodes; and diamonds are utility nodes. Arrows show causal influence. For clarity, only the *to* slot is shown, and the action and utility nodes for the *t+1* time-step have been omitted. The dotted boxes show composite state and action variables.

Set	Meaning
S_i^{slot}	User's goal for slot *i*
S_i^{eq}	Indicates whether A_i^{slot} equals S_i^{slot}
S_i^{A}	System's action as it relates to this slot
S_i^{WH}	WH-portion of user's action, e.g., *to(x)* or *x*
S_i^{YN}	YN-portion of user's action (*yes, no)*
S_i^{WH1}	WH action with slot value only, e.g., *london*
S_i^{WH2}	WH act w/ slot name & value, e.g., *to(london)*
S_i^{dlg}	Dialog state: *{not stated, stated,* or *confirmed}*
O	Full concept string (both slots)
A^{force}	Sys Act: *ask-to/from, confirm-to/from,* or *submit*
A_i^{slot}	Content of system action (same set as S_i^{slot})
R	Reward (see Table 2 and text)

Table 1: Definition of node labels in Figure 1.

Dialog state	System action	Reward
Not stated	*ask*	–1
Not stated	*confirm*	–3
Stated	*ask*	–2
Stated	*confirm*	–1
Confirmed	*ask*	–3
Confirmed	*confirm*	–2

Table 2: Per-turn rewards for the sample problem.

the master POMDP. Each $\tilde{s}_i^{slot}$ takes on just 2 values, $\tilde{s}_i^{slot} \in \{best, rest\}$, calculated as follows:

$$\tilde{s}_i^{slot} = \begin{cases} best, & if \arg\max(b_t(s_i^{slot})) = S_i^{slot}(t) \\ rest, & if \arg\max(b_t(s_i^{slot})) \neq S_i^{slot}(t). \end{cases}$$

Essentially, for a given slot, the state of the summary POMDP expresses whether the most likely hypothesis in the master POMDP is correct.

Other state space components from the master POMDP may be included in the summary POMDP state space if desired. In the example dialog problem, the state of the summary POMDP $\tilde{S}$ includes a component $\tilde{S}_i^{dlg}$ which is a copy of the dialog state in the master POMDP, S_i^{dlg}. The example summary POMDP includes a total of 36 states. Figure 2 shows an influence diagram of the example summary POMDP, illustrating how a summary POMDP contains fewer and more compact components than the master POMDP, facilitating optimization.

The action space of the summary POMDP includes only A^{force}. Thus, in the example dialog problem, the summary action $\tilde{A}$ includes the component A^{force} but not A_{from}^{slot} or A_{to}^{slot}, yielding a total of five actions in the example summary POMDP.

The observation set of the summary POMDP, $\tilde{O}$, includes two components $\tilde{O}_i^{S\leftrightarrow O}$ and $\tilde{O}_i^{S\leftrightarrow S}$ for each slot *i*. The first observation component is calculated as follows:

$$\tilde{O}_i^{S\leftrightarrow O} = \begin{cases} c,\ if\ o \text{ is consistent with argmax } (b(s_i^{slot})) \\ ic,\ if\ o \text{ is inconsistent with argmax}(b(s_i^{slot})) \\ ni,\ if\ o \text{ provides no information about } b(s_i^{slot}) \end{cases}$$

This "consistency" relationship is specified by the system designer. In the example dialog problem, if the observation contains *london* (without an indication of *to* or *from*), it is consistent with $\arg\max(b(s_i^{slot})) = london$ and inconsistent with any other value for both *i = to* and *i = from*. If the observation contains *to(london),* it is consistent with $\arg\max(b(s_{to}^{slot}))$ = *london* and provides no information about s_{from}^{slot}. The

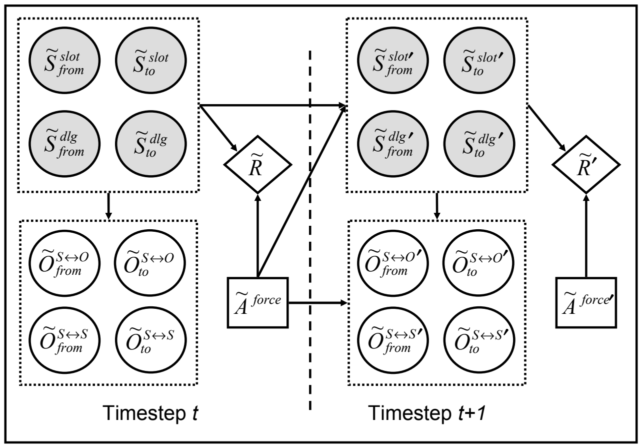

$\tilde{S}_{from}^{slot}$
$\tilde{S}_{to}^{slot}$
$\tilde{S}_{from}^{dlg}$
$\tilde{S}_{to}^{dlg}$
$\tilde{R}$
$\tilde{O}_{from}^{S\leftrightarrow O}$
$\tilde{O}_{to}^{S\leftrightarrow O}$
$\tilde{O}_{from}^{S\leftrightarrow S}$
$\tilde{O}_{to}^{S\leftrightarrow S}$
$\tilde{A}^{force}$
$\tilde{S}_{from}^{slot\prime}$
$\tilde{S}_{to}^{slot\prime}$
$\tilde{S}_{from}^{dlg\prime}$
$\tilde{S}_{to}^{dlg\prime}$
$\tilde{R}'$
$\tilde{O}_{from}^{S\leftrightarrow O\prime}$
$\tilde{O}_{to}^{S\leftrightarrow O\prime}$
$\tilde{O}_{from}^{S\leftrightarrow S\prime}$
$\tilde{O}_{to}^{S\leftrightarrow S\prime}$
$\tilde{A}^{force\prime}$
Timestep *t*
Timestep *t+1*

Figure 2: Influence diagram showing the summary POMDP for the sample dialog problem

observations *yes* and *no* alone provide no information about any slot via $\tilde{O}_i^{S\leftrightarrow O}$ – i.e., $\tilde{O}_i^{S\leftrightarrow O} = ni$.

The second observation component is calculated as follows:

$$\tilde{O}_i^{S\leftrightarrow S} = \begin{cases} eq, \text{ } if \arg\max(b_{t-1}(s_i^{slot})) = \arg\max(b_t(s_i^{slot})) \\ ne, \text{ } if \arg\max(b_{t-1}(s_i^{slot})) \neq \arg\max(b_t(s_i^{slot})) \end{cases}$$

In other words, $\tilde{O}_i^{S\leftrightarrow S}$ indicates whether the most likely hypothesis for s_i^{slot} has changed. For example, in the sample problem, suppose that $\arg\max(b_t(s_{from}^{slot})) = london$, the action *confirm-from(london)* is taken, and *no* is observed. This would result in a significant reduction in $b_{t+1}(london)$, and a new most likely hypothesis, for example $\arg\max(b_{t+1}(s_{from}^{slot})) = leeds$. In this example, $london \neq leeds$, so $\tilde{O}_{from}^{S\leftrightarrow S} = ne$.

3.3. Sampling & Optimization

To estimate the system dynamics of the summary POMDP, we sample from the master POMDP using a random policy. At each timestep, an action $\tilde{a} \in \tilde{A}$ in the summary POMDP is randomly selected. The action $a \in A$ in the master POMDP is then formed by setting $a_i^{slot} = \arg\max(b(s_i^{slot}))$ and combining $\tilde{a}$ and $a_i^{slot}, \forall i$ to form *a*. Finally, the value of the reward *r* is calculated.

Next, a value for the next (unobserved) state s' and observation o' of the master POMDP are then sampled. Then the belief state $b'(s')$ of the master POMDP is calculated as in Eq. 1. Finally, the state $\tilde{s}'$ and observations $\tilde{o}'$ of the summary POMDP are calculated as described in 3.2. By sampling repeatedly in this way, the transition function $P(\tilde{s}' | \tilde{s}, \tilde{a})$, observation function $P(\tilde{o}' | \tilde{s}', \tilde{a})$, and reward function $r(\tilde{s}, \tilde{a})$ of the summary POMDP are estimated. After sampling, optimization is performed on the summary POMDP to compute a policy, $\tilde{\pi}(\tilde{b}(\tilde{s})) \in \tilde{a}$, using for example [11].

3.4. Execution

To execute the controller, belief monitoring is performed in the master POMDP. The belief state of the summary POMDP is calculated as

$$\tilde{b}(\tilde{s}_i^{slot} = best) = \max_{s_i^{slot}}(b(s_i^{slot})) \qquad (3)$$

and $\tilde{b}(\tilde{s}_i^{slot} = rest) = 1 - \tilde{b}(\tilde{s}_i^{slot} = best)$. The action $\tilde{a}$ is selected based on the policy calculated above, i.e., $\tilde{a} = \pi(\tilde{b}(\tilde{s}))$. The action $a \in A$ in the master POMDP is then formed by setting $a_i^{slot} = \arg\max(b(s_i^{slot}))$ for all slots *i* and combining $\tilde{a}$ and $a_i^{slot}, \forall i$ to form *a*. After *a* is taken, a reward *r* and an observation o' are received, and the next belief state $b'(s')$ is calculated (Eq. 1.)

4. DEMONSTRATION

Figure 5 shows an example conversation between the test user model and the POMDP. The small graphs show the distribution of "belief mass" in $b(s_i^{slot})$ and $\tilde{b}(\tilde{s}_i^{slot})$ over the course of the dialog. Only 3 of the 1000 values of s_i^{slot} are shown.

At the beginning of the dialog, the belief mass in $b(s_i^{slot})$ is spread evenly over all goals, and $\tilde{b}(\tilde{s}_i^{slot} = best)$ is low. As the dialog progresses, probability mass sharpens around observed slot values. As explained in the description of the method above, at each time-step, $\tilde{b}(\tilde{s}_i^{slot} = best) = \max(b(s_i^{slot}))$. For example, after turn *U1*, $\arg\max(b(s_{to}^{slot})) = leeds$ and $\tilde{b}(\tilde{s}_i^{slot} = best) = b(leeds)$. $\tilde{b}(\tilde{s}_i^{slot} = rest)$ is equal to $\sum b$ for *all other* values of s_i^{slot} .

One strength of the probabilistic approach is that the system never commits to one value for a slot; rather, $\tilde{b}(best)$ is acting as a global confidence score over the course of the dialog. For example, the *no* observed in *U4* redistributes probability mass in $b(s_{to}^{slot})$ from $s_{to}^{slot} =$ *leeds* to all other values, causing a decrease in $\tilde{b}(s_{to}^{slot} = best)$. Alternatively, a second observation providing further support for $s_{from}^{slot} =$ *cambridge* in *U5* causes an increase in $\tilde{b}(s_{from}^{slot} = best)$. A second strength is that the effect on *b* of an observation is scaled by the likelihood of the corresponding user action. For example, in *U4,* the user action *from(sheffield)* is very unlikely, so when the observation *from(sheffield)* is made it increases $b(s_{from}^{slot} = sheffield)$ but only slightly.

5. EVALUATION

We are interested in determining how performance of the summary method varies with error rate, how robust the method is to variations in user behavior, and finally how the summary method compares to traditional methods.

To do this, we employ real dialog data from the *SACTI-1* corpus [13]. The *SACTI-1* corpus contains 144 human-human dialogs in the travel/tourist information domain using a "simulated ASR channel" [12]. The corpus contains a variety of word error rates, and the behaviors observed of the subjects in the corpus are broadly consistent with behaviors observed of a user and a computer using a real speech recognition system [13]. The corpus was segmented into a "training sub-corpus" and a "test sub-corpus," which are each composed of an equal number of dialogs, the same mix of word error rates, and disjoint subject sets. Wizard/User turn pairs were annotated, and one user model was then estimated from each sub-corpus.

The training user model was installed in nodes S_i^{WH} and S_i^{YN} in the master POMDP, giving the *wh* portion of the user's response such as *to(cambridge)*, and the *yn* (*yes/no*) portion, respectively. To create the summary POMDP, 20,000 dialog turns were sampled. Policy optimization was performed with 50 belief points, 50 iterations, and a discount of $\lambda = 0.99$.[2] To form a baseline, 20,000 dialog turns were then run with the

[2] The number of belief points broadly sets an upper bound on the complexity of the resulting policy.

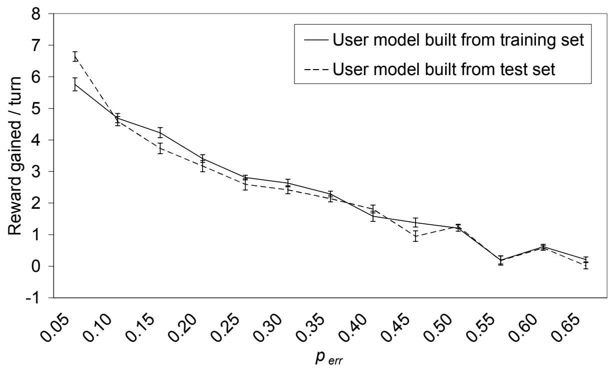


Figure 3: p_{err} vs. reward gained per turn for the sample dialog problem

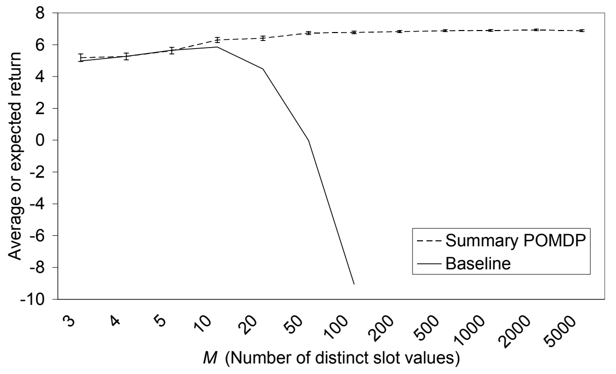


Figure 4: *n* (number of distinct slot values) vs. average or expected return for a simplified 1-slot dialog problem. The baseline is a direct solution of the master POMDP.

resulting policy and the training user model. Next, the test user model was installed into the Master POMDP, and 20,000 dialog turns were run with the policy created from the training user model. This process was repeated for values of p_{err} from 0.05 to 0.60.

Figure 3 shows results for a range of values of p_{err}. The Y-axis shows average reward gained per dialog turn, and error bars indicate the 95% confidence interval for the true average reward gained per dialog turn. As speech recognition errors increase, the average reward per turn decreases, consistent with traditional methods [8, 14, 15]. For all values of p_{err} except 0.05, performance on the test user model is no less than 0.5 points of the training user model and is generally very close. In other words, the policy created for the training user model is performing similarly on the testing user model, implying that the method is reasonably robust to variations in patterns of user behavior. In some cases, e.g., p_{err} = 0.05, the policy performs better on the test user model than on the training user model. This is possible because different user models can provide different amounts of information. In this example, the test user model provides slightly more information than the training user model, which enables the policy to perform better on the testing user model at certain error rates.

Finally, the system was run in a simpler one slot configuration which allowed the master POMDP to be optimized directly. Direct solutions were computed with 1,000 belief points and 50 iterations. The summary POMDP method was then applied using 20,000 dialog turns, 50 belief points and 50 iterations. This process was repeated for values of *M* (i.e., distinct number of slot values) from 3 to 5000. The concept error rate was set to p_{err} = 0.30, and the discount to $\lambda = 0.99$ for all experiments.

Figure 4 shows *M* vs. average or expected return for the summary method and the baseline. The solution algorithm was not able to optimize the master POMDP directly for *M* > 100. Error bars show 95% confidence intervals for true expected return. For small problems, i.e., lower values of *M*, the summary method performs equivalently to the baseline. For larger problems, the summary method outperforms the baseline by an increasing margin. Moreover, the summary POMDP policy was derived using 95% fewer belief points than the baseline; i.e., the summary method's policies scale to large problems *and* are much more compact.

It is interesting to note that as *M* increases, the performance of the summary POMDP method appears to increase toward an asymptote. This trend is due to the fact that all confusions are equally likely in this model. For a given error rate, the more concepts in the model, the less likely *consistent* confusions are. Thus, having more concepts helps the policy identify spurious evidence over the course of a dialog.

6. CONCLUSION

POMDP-based dialog managers have been shown to outperform MDP-based dialog managers, but to date have been limited to artificially small problems. This paper has demonstrated a method for scaling POMDP-based dialog managers to problems of a realistic size. The method has been demonstrated with a 2-slot problem incorporating a user model estimated from real dialog data, and experiments have shown that the resulting dialog manager copes well with changes in patterns of user behavior. Moreover, summary dialog policies appear to be as good as those derived directly for the full model.

As presented here, the size of the state space and observation space in the summary POMDP still grow exponentially in the number of slots. While many real-world problems use a handful of slots (e.g. 2, 3 or 4), others use many slots. Thus one theoretical issue to address in future work is how to scale to large numbers of slots. The factored nature of the summary POMDP may be of some help here, for example [7]. Finally, future practical work will attempt to validate the method by constructing an end-to-end system.

7. ACKNOWLEDGEMENTS

The authors would like to thank Pascal Poupart for several helpful, insightful discussions. Thanks also to Matt Stuttle and Jost Schatzmann for helpful comments on the presentation of this paper. The work reported in this paper was supported by the EU FP6 Talk Project.

11. REFERENCES

[1] Finn V. Jensen. *Bayesian Networks and Decision Graphs*. New York: Springer Verlang, 2001.

[2] Leslie Pack Kaelbling, Michael L. Littman and Anthony R. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, Vol. 101, 1998.

[3] Staffan Larsson and David Traum. Information state and dialogue management in the trindi dialogue move engine toolkit. Natural Language Engineering, 5(3–4):323–340, 2000.

[4] Esther Levin, Roberto Pieraccini, and Wieland Eckert. A Stochastic Model of Human-Machine Interaction for Learning Dialogue Strategies. *IEEE Transactions on Speech and Audio Processing*, Volume 8, No. 1, 11-23, 2000.

[5] Esther Levin and Roberto Pieraccini. A Stochastic Model of Computer-Human Interaction For Learning Dialogue Strategies. *Eurospeech*, Rhodes, Greece, 1997.

[6] Olivier Pietquin. *A Framework for Unsupervised Learning of Dialogue Strategies*. Ph D thesis, Faculty of Engineering, Mons, Belgium, 2004.

[7] Pascal Poupart and Craig Boutilier. VDCBPI: an Approximate Scalable Algorithm for Large Scale POMDPs. *Advances in Neural Information Processing Systems 17* (NIPS-2004), Vancouver, BC, pp 1081-1088.

[8] Nicholas Roy, Joelle Pineau and Sebastian Thrun. Spoken Dialogue Management Using Probabilistic Reasoning. *Annual meeting of the Association for Computational Linguistics* (ACL-2000).

[9] Konrad Scheffler and Steve Young. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. *Proc. Human Language Technology* (HLT-2002), San Diego, pp. 12-18.

[10] Satinder Singh, Diane Litman, Michael Kearns and Marilyn Walker. Optimizing Dialogue Management with Reinforcement Leaning: Experiments with the NJFun System. *Journal of Artificial Intelligence*, Vol. 16, 105-133, 2002.

[11] Matthijs T. J. Spaan and Nikos Vlassis. Perseus: randomized point-based value iteration for POMDPs. Technical Report IAS-UVA-04-02, Informatics Institute, University of Amsterdam, 2004.

[12] Matthew Stuttle, Jason D. Williams, and Steve Young. A Framework for Wizard-of-Oz Experiments with a Simulated ASR-Channel. *International Conferences on Spoken Language Processing (ICSLP-2004)*, Jeju, South Korea, 2004.

[13] Jason D. Williams and Steve Young. Characterizing Task-Oriented Dialog using a Simulated ASR Channel. *International Conference on Spoken Language Processing* (ICSLP), October 2004, Jeju, South Korea.

[14] Jason D. Williams, Pascal Poupart, and Steve Young. Factored Partially Observable Markov Decision Processes for Dialogue Management. *4th Workshop on Knowledge and Reasoning in Practical Dialog Systems, International Joint Conference on Artificial Intelligence (IJCAI)*, August 2005, Edinburgh.

[15] Jason D. Williams, Pascal Poupart, and Steve Young. Partially Observable Markov Decision Processes with Continuous Observations for Dialogue Management. In *Proc. 6th SigDial Workshop on Discourse and Dialogue*, September 2005, Lisbon.

[16] Zhang Bo, Cai Qingsheng, Mao Jianfeng, and Guo Baining. Planning and Acting under Uncertainty: A New Model for Spoken Dialogue System. *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)*. San Francisco, USA, 2001.

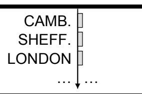
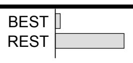
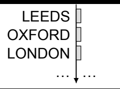
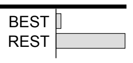
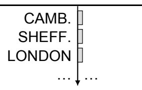
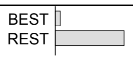
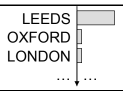
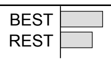
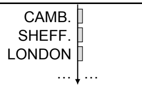
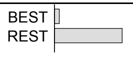
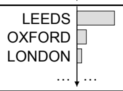
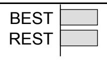
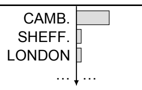
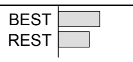
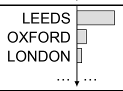
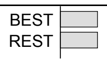
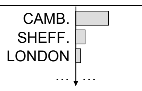
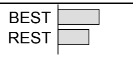
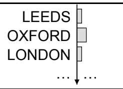
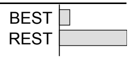
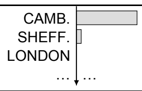
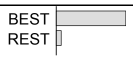
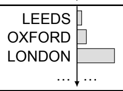
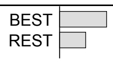
Turn	Notes	$b(s^{slot}_{from})$	$\tilde{b}(\tilde{s}^{slot}_{from})$	$b(s^{slot}_{to})$	$\tilde{b}(\tilde{s}^{slot}_{to})$
[prior to dialog start]	Prior to dialog start, probability mass is spread evenly over all slot values.				
S1: Where are you going to? *U1: London* [misrecognized as *Leeds*]	Top hypothesis for *to* slot is now *leeds*; *from* slot has received no information.				
S2: Where are you leaving from? *U2: From Cambridge* [misrec as *To Oxford*]	User model predicts this obs. is unlikely: *to(oxford)* gets minimal belief mass.				
S3: Where are you leaving from? *U3: From Cambridge* [reco ok]	Top hypothesis for *from* slot is now *cambridge.*				
S4: To Leeds, is that right? *U4: No, to London* [misrec as *No, from Sheffield*]	*to(leeds)* reduced; *from(sheffield)* has small effect b/c of user model.				
S5: Where are you going to? *U5: To London from Cambridge* [reco ok]	Top hypothesis for *to* slot is now *london*; *from* slot is now very certain.				
[System prints ticket from London to Cambridge]					

Figure 5: Example conversation and belief states. The user is trying to travel from Cambridge to London.