Scaling POMDPs for dialog management with composite summary point-based value iteration (CSPBVI)

Jason D. Williams*
AT&T Labs – Research
180 Park Ave, Florham Park, NJ, USA
jdw@research.att.com

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

Abstract

Although partially observable Markov decision processes (POMDPs) have shown great promise as a framework for dialog management in spoken dialog systems, important scalability issues remain. This paper tackles the problem of scaling slot-filling POMDP-based dialog managers to many slots with a novel technique called composite point-based value iteration (CSPBVI). CSPBVI creates a "local" POMDP policy for each slot; at runtime, each slot nominates an action and a heuristic chooses which action to take. Experiments in dialog simulation show that CSPBVI successfully scales POMDP-based dialog managers without compromising performance gains over baseline techniques and preserving robustness to errors in user model estimation.

Introduction

Partially observable Markov decision processes (POMDPs) provide a principled formalism for planning under uncertainty and past work has argued that POMDPs are a suitable framework for spoken dialog management. POMDPs have been shown to outperform simpler techniques such as (fully-observable) Markov decision processes (MDPs) and handcrafted dialog managers, especially in the face of higher speech recognition error rates, and to make better use of confidence score information (Roy, Pineau, & Thrun 2000; Williams, Poupart, & Young 2005b; 2005a; 2005b).

Despite their advantages, POMDPs are notoriously difficult to scale. Within the class of so-called *slot-filling* dialogs, the *Summary POMDP* method has enabled the number of values taken on by *one slot* (such as the number of airports in a travel system) to be scaled, but the problem of how to scale the *number of slots* remains (Williams & Young 2005). This paper proposes a novel POMDP optimization technique tailored to spoken dialog systems called *composite summary point-based value iteration* (CSPBVI) which scales to many slots, each with many slot values. CSPBVI keeps optimization tractable by optimizing a dialog manager for each slot locally and combining these at run-time using a simple heuristic.

This paper first reviews the definition of POMDPs and the SDS-POMDP model. Next, the intuition and details of the CSPBVI method are presented, followed by an example spoken dialog system simulator called MULTISLOT. After illustrating the operation of CSPBVI on MULTISLOT, CSPBVI is compared to a variety of baseline techniques, and finally brief conclusions are made.

Background

Formally, a POMDP $\mathfrak{P}$ is defined as a tuple $\mathfrak{P} = (\mathbb{S}, \mathbb{A}, \mathbb{T}, \mathbb{R}, \mathbb{O}, \mathbb{Z}, \gamma, b_0)$ where $\mathbb{S}$ is a set of states s describing the machine's world with $s \in \mathbb{S}$; $\mathbb{A}$ is a set of actions a that an machine may take $a \in \mathbb{A}$; $\mathbb{T}$ defines a transition probability $P(s'|s,a)$; $\mathbb{R}$ defines the expected (immediate, real-valued) reward $r(s,a) \in \Re$; $\mathbb{O}$ is a set of observations o the machine can receive about the world with $o \in \mathbb{O}$; $\mathbb{Z}$ defines an observation probability $P(o'|s',a)$; γ is a geometric discount factor $0 < \gamma < 1$; and b_0 is an initial belief state, defined below.

The POMDP operates as follows. At each time-step, the machine is in some unobserved state s. Since s is not known exactly, a *distribution* over possible states called a *belief state* b is maintain where $b(s)$ indicates the probability of being in a particular state s. Based on b, 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 time-step, b is updated as $b'(s') = \eta \cdot P(o'|s',a) \sum_s P(s'|s,a)$ where η is a normalization constant (Kaelbling, Littman, & Cassandra 1998). The process of maintaining b at each time step is called *belief monitoring*. The cumulative, infinite-horizon, discounted reward is called the *return* and written $V = \sum_{\tau=1}^{\infty} \gamma^\tau \sum_s b_\tau(s) r(s, a_\tau)$, and the goal of the machine is to choose actions so as to maximize this quantity.

Previous work has cast a spoken dialog system (SDS) as a POMDP to produce a model called the SDS-POMDP (Williams, Poupart, & Young 2005a; 2005b). In the SDS-POMDP, the POMDP state variable s is separated into three components, $s = (s_u, a_u, s_d)$. The component $s_u \in \mathbb{S}_u$ gives the *user's goal*, such as a complete travel itinerary. This paper is concerned with so-called slot-filling dialogs

*Work performed while at Cambridge University Engineering Department. Supported in part by EU FP6 Talk Project. The authors would like to thank Pascal Poupart for helpful comments.

in which the user's goal s_u is composed of W *slots*, $s_u = (s_u^1, \ldots, s_u^W)$, where $s_u^w \in \mathcal{S}_u^w$. For example, in the air travel domain, a user goal s_u might be composed of $s_u = (s_u^{\text{FROM}}, s_u^{\text{TO}}, s_u^{\text{CLASS}}, s_u^{\text{AIRLINE}}, s_u^{\text{TIME}}, s_u^{\text{DATE}})$. The component $a_u \in \mathbb{A}_u$ gives the most recent *user action* at the concept level, such as stating a place the user would like to travel to, responding to a yes/no question, or a "null" response indicating the user was silent. Finally the component $s_d \in \mathbb{S}_d$ records relevant *dialogue history*, such as the grounding status of a slot, or how many times a slot has been queried. None of these components is observable directly by the machine and the SDS-POMDP belief state is formed of a distribution over these components $b(s_u, a_u, s_d)$. The POMDP action a corresponds to the machine action in the dialog, such as greeting the user, asking the user where they want to go "to", or confirming a user goal. Finally, the POMDP observation o is separated into two components $o = (\tilde{a}_u, c)$, where $\tilde{a}_u \in \mathbb{A}_u$ gives the hypothesis of the user's action provided by the speech recognition process, and c is a confidence score.

By substitution and making reasonable conditional independence assumptions, the POMDP transition function $P(s'|s,a)$ and observation function $P(o'|s',a)$ can be re-written in SDS-POMDP terms as $P(s'|s,a) = P(s'_u|s_u,a)P(a'_u|s'_u,a)P(s'_d|a'_u,s_d,a)$ and $P(o'|s',a) = P(\tilde{a}'_u,c'|a'_u)$. These individual probability functions correspond to intuitive *models* which can either be estimated from data or handcrafted. For example, $P(a'_u|s'_u,a)$ provides a model of user behavior which can be estimated from dialog data, and $P(s'_d|a'_u,s_d,a)$ could be handcrafted following e.g., the Information State Update approach (Larsson & Traum 2000). The design of the reward function $r(s_u,a_u,s_d,a)$ is left to the application designer as it implements the design objectives of a given system. In general r encodes trade-offs between speed, appropriateness, and accuracy, and one would expect these to be different in (for example) the banking and entertainment domains.

Optimization of an SDS-POMDP faces severe scalability issues. The set $\mathbb{S}_u$ contains all possible user goals and as a result the cardinality of the SDS-POMDP state space grows as more user goals are added. For example, if the size of each slot is $|\mathcal{S}_u^w| = 1000$, then there are a total of $|\mathbb{S}_u| = \prod_w |\mathcal{S}_u^w| = 1000^W$ distinct user goals. Because the set of user actions $\mathbb{A}_u$ and machine actions $\mathbb{A}$ often refer to specific user goals, the SDS-POMDP action and observation sets all grow with the number of user goals. As a result, straightforward optimization of the SDS-POMDP model using general-purpose techniques is only possible with an unrealistically small number of user goals.

One approach to scaling POMDPs in the dialog domain is the *Summary POMDP* method (Williams & Young 2005). The intuition here is to constrain machine actions which refer to a user goal (such as *confirm-to-london*) to always refer to the *most likely* user goal. This simplifies the planning task considerably by reducing the size of the action and observation sets, and by allowing the planner to consider just the *ratio* of belief mass held by the most likely user goal in each slot. That is, rather than planning over $|\mathbb{S}_u| = \prod_w |\mathcal{S}_u^w|$ user goals, the Summary POMDP method plans over $\prod_w 2 = 2^W$ possible user goal summarizations. This represents a significant improvement and results have shown that the summary POMDP method can effectively scale the SDS-POMDP model to problems with two slots and an arbitrary number of values for each slot (Williams & Young 2005). However, 2^W is of course still exponential in the number of slots and the summary POMDP method cannot handle many slots.

CSPBVI method description

Composite summary point-based value iteration (CSPBVI) extends the Summary POMDP method to handle many slots. In a recent data collection (Williams & Young 2004), it was noticed that when users are asked about a certain slot, they most often provide a value for just that slot, and only sometimes provide values for other slots. CSPBVI capitalizes on this insight by assuming that cross-slot effects are unimportant for planning: it first estimates system dynamics locally for each slot, then uses these estimates to produce a distinct dialog manager (i.e., POMDP policy) for each slot. At runtime, each dialog manager nominates an action appropriate for its slot and a handcrafted heuristic chooses which one of these to take. CSPBVI still performs belief monitoring over all user goals, so when a user *does* provide extra information it is properly incorporated into the belief state – the key idea is that actions are nominated by each slot based on the expectation that user responses will not provide information about other slots.

CSPBVI consists of four phases: construction, sampling, optimization, and execution. In the *construction* phase, first the *master* POMDP is created, which is an SDS-POMDP with several constraints and additions. The user's goal $s_u \in \mathbb{S}_u$ is decomposed into W slots, $s_u = (s_u^1, \ldots, s_u^W)$ where $s_u^w \in \mathcal{S}_u^w$ and where $\mathcal{S}_u^w$ refers to the set of values for slot w. The dialog history $s_d \in \mathbb{S}_d$ is similarly decomposed into W slots, $s_d = (s_d^1, \ldots, s_d^W)$ where $s_d^w \in \mathcal{S}_d^w$ and where $\mathcal{S}_d^w$ refers to the set of possible dialog histories for w. Machine actions are formed of predicates which take arguments that encode the slot w and the value (or values) s_u^w which the action that refers to. Machine actions are written *predicate*$[w](x)$, where *predicate* refers to the illocutionary force of the action, w refers to a slot index, and x refers to the slot value(s) referred to by the action, if any. For example, the SDS-POMDP machine actions *ask-from* and *confirm-to-london* would be restated as *ask*[*from*]() and *confirm*[*to*](*london*). A special meta-slot $w =$ *all* denotes an action refers to all slots, such as *submit*[*all*]($s_u^{\text{from}} =$ *london*, $s_u^{\text{to}} =$ *paris*) and *greet*[*all*]$()$. Finally, in the master POMDP a modified reward function is created $r_w(s,a)$ which removes conditioning on all but the slot w. For example, if the reward r for incorrectly/correctly submitting a user's *complete* goal is $-25/+25$, then r_w would assess $-25/+25$ for incorrectly/correctly submitting *only* slot w, ignoring all others. Also, belief monitoring must be tractable in the master POMDP, and this may require approximations in the observation function; an example of this is shown in the next section.

After the master POMDP is formed, W summary belief

Markov decision processes (BMDPs) are constructed.[1] Each of these has a state space with two components, $\hat{\mathcal{S}}_u^w$ and $\hat{\mathcal{S}}_d^w$, where $\hat{\mathcal{S}}_u^w = \{\textit{best}, \textit{rest}\}$ and $\hat{\mathcal{S}}_d^w = \mathcal{S}_d^w$. The action set of each of these summary BMDPs consists of the predicates of $\mathbb{A}$ and take one argument, $\hat{w} \in \{\textit{this}, \textit{other}\}$, where $\hat{w} = \textit{this}$ indicates that an action in master space a refers to *this* slot and $\hat{w} = \textit{other}$ indicates that a refers to *some other* slot. (If the action a operates on *all* slots, $\hat{w}$ is set to *this*.) For example, in a slot-filling SDS-POMDP with two slots *from* and *to*, a master POMDP action $a = \textit{confirm}[\textit{from}](\textit{london})$ would be mapped to $\hat{a}^{\text{from}} = \textit{confirm}[\textit{this}]$ in the summary BMDP for the *from* slot, and $\hat{a}^{\text{to}} = \textit{confirm}[\textit{other}]$ in the summary BMDP for the *to* slot.

The *sampling* phase of CSPBVI consists of two stages. The first stage iterates over each slot $w = 1 \ldots W$. For each w the machine takes actions randomly to sample N points in summary space, written as the set $\hat{\mathcal{B}}_w = \{\hat{b}_{w,n}\}$. Initially $\hat{\mathcal{B}}_w$ is empty and at each step $n = 1 \ldots N$, the current belief point b is mapped into summary space for slot w to produce $\hat{b}$ by setting $\hat{b}(\hat{s}_u^w = \textit{best}) \leftarrow \max_{s_u^w} b(s_u^w)$, $\hat{b}(\hat{s}_u^w = \textit{rest}) \leftarrow 1 - \hat{b}(\hat{s}_u^w = \textit{best})$, and $\hat{b}(s_d^w) \leftarrow b(s_d^w), \forall s_d^w \in \mathcal{S}_d^w$. If $\hat{b}$ is not already contained in $\hat{\mathcal{B}}_w$, then it is added and two other quantities are sampled. From b, the machine takes each summary action K times, $k = 1 \ldots K$, resetting to b after each, and recording the resulting reward in $\hat{r}_{w,n}^{\hat{a},k}$ and successor point in summary space in $\hat{b}_{w,n}^{\hat{a},k}$. After N points are sampled in this way for slot w, this first stage is repeated for the *corners* of summary space for each slot to help ensure coverage of summary space.

In the second stage of the sampling phase, for each point $\hat{b}_{w,n}^{\hat{a},k}$, the closest point in $\hat{b}_{w,n}$ is located and its index is recorded in $l(w, n, \hat{a}, k)$.

CSPBVI *optimization* is run W times, once for each slot w using that slot's dynamics and reward. Back-ups are performed on the belief MDP estimated for each slot w. The backups run from $t = 1 \ldots T$, where T is the plan horizon. Each back-up first computes $\hat{q}^{\hat{a},n}$, which estimates the value of taking action $\hat{a}$ from point $\hat{b}_{w,n}$, then from this compute $\hat{a}_t^{w,n}$ (the optimal t-step action at $\hat{b}_{w,n}$) and $\hat{v}_t^{w,n}$ (the expected value of the optimal t-step policy starting from $\hat{b}_{w,n}$). $\hat{v}_0^{w,n}$ is initialized to 0 for all w and n.

$$
\begin{aligned}
\hat{q}^{\hat{a},n} &\leftarrow \frac{1}{K}\sum_k \hat{r}_{w,n}^{\hat{a},k} + \frac{\gamma}{K}\sum_k \hat{v}_{t-1}^{l(w,n,\hat{a},k)} \\
\hat{a}^* &\leftarrow \arg\max_{\hat{a}} \hat{q}^{\hat{a},n} \\
\hat{a}_t^{w,n} &\leftarrow \hat{a}^* \\
\hat{v}_t^{w,n} &\leftarrow \hat{q}^{\hat{a}^*,n}
\end{aligned}
$$

Summary actions selected in each iteration are *restricted* to $\hat{w} = \textit{this}$: that is, only actions which operate on *this* slot (or all slots) are incorporated into conditional plans. Optimization ultimately produces an optimal summary action $\hat{a}^{w,n}$ for each point $\hat{b}_{w,n}$.

[1] A *Belief MDP* is a Markov decision process with a continuous state corresponding to a POMDP belief state (Kaelbling, Littman, & Cassandra 1998).

To *execute* a policy, belief monitoring is performed in the master POMDP. For a given belief point b, the corresponding set of summary belief points $\hat{b}_w$ is computed for all slots w. For each belief point $\hat{b}_w$ the index of the closest point n^* in the set $\{\hat{b}_{w,n}\}$ is found, and its summary action ($\hat{a}^{w,n^*}$) is mapped to a master action a^w. This process is repeated for each slot w and produces a vector of nominated master actions, a^w. Finally, a handcrafted heuristic, which must be created for each application, selects an action from this vector to take. Because the number of summary actions and summary states are constant with respect to the number of slots (and the number of values for each slot), CSPBVI scales to handle many slots. The quality of the solution produced is a function of the optimization parameters T, N, and K, and of the quality of the handcrafted action selection heuristic.

Example spoken dialog system

A POMDP-based dialog manager called MULTISLOT was created. MULTISLOT is an SDS-POMDP with W slots, where each slot contains 100 values and where W can be varied. To keep belief monitoring tractable, some independence assumptions between slots are made. User actions are decomposed by slot into $a_u = (a_u^1, \ldots, a_u^W)$, and each per-slot user action element a_u^w is decomposed into three components $a_u^w = (a_{\text{state}}^w, a_{\text{stateSlot}}^w, a_{\text{yesNo}}^w)$, where $a_{\text{state}}^w \in \mathcal{A}_{\text{state}}^w$, $a_{\text{stateSlot}}^w \in \mathcal{A}_{\text{stateSlot}}^w$, and $a_{\text{yesNo}}^w \in \mathcal{A}_{\text{yesNo}}^w$. $\mathcal{A}_{\text{state}}^w$ consists of $\textit{state}[w](s_u^w)$ and indicates the user said their goal *without identifying* which slot it corresponds to – for example, "London" or "10:00 AM". $\mathcal{A}_{\text{stateSlot}}^w$ consists of $\textit{stateSlot}[w](s_u^w)$ and indicates the user said their goal *and identified* which slot it corresponds to – for example, "to London", "from London", "leaving at 10:00 AM", or "arriving at 10:00". Finally $\mathcal{A}_{\text{yesNo}}^w$ includes actions *yes* and *no*. The sets $\mathcal{A}_{\text{state}}^w$, $\mathcal{A}_{\text{stateSlot}}^w$, and $\mathcal{A}_{\text{yesNo}}^w$ each also contain *null*.

Next, the user action model $p(a'_u|s'_u, a)$ was extended to support this formulation. Each slot contains a slot-specific user model, conditioned on whether the machine is asking about this slot or another (i.e., *any* other) slot. To make the user action model as realistic as possible, real dialog data from the SACTI-1 corpus was employed (Williams & Young 2004). The SACTI-1 corpus contains 144 human-human dialogs in the travel/tourist information domain using a "simulated ASR channel" (Stuttle, Williams, & Young 2004). 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 (Williams & Young 2004). 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, called the *training* user model and the *testing* user model.

A key property of spoken dialog systems is that speech recognition errors may be made both *within* and *between* slots. To model this as closely as possible, the observation model was separated into a *generation* model and an *infer-*

ence model. An important goal in this work is to allow the user to say anything at any point, and so we assume that the same recognition grammar is active throughout. To model this, the generation model makes concept confusions with a constant probability p_{err}, where a confusion substitutes a non-null user action component to any other component *in any slot*. For example, if one concept error is made, the user action "Yes, London" might be changed to "Frankfurt London" or even "Yes No". Since *null* is one type of user action, the generation model also simulates deletion errors – for example, "Yes, London" could be changed to "Yes", "London" or *null*. The model does not simulate insertion errors. Also, each observation component (such as "London" or "To Edinburgh" or "Yes") carries with it a *per-concept* confidence score. Confidence scores for correct recognitions are sampled from $p_h(c)$ and incorrect recognitions are sampled from $p_h(1-c)$, where $p_h(c) = (he^{hc})/(e^h - 1)$. When $h = 0$, the confidence score is random noise, and as h increases the confidence score becomes more reliable. Ideally the observation inference model should express the probability of the entire observation given the entire user action $P(\tilde{a}'_u, c'|a_u)$, but this formulation would complicate belief monitoring significantly. Instead, the observation model estimates $P(\tilde{a}'_u, c'|a_u^{w\prime})$ *separately* for each slot: if a_u^w exists in the observation $\tilde{a}'_u$, then $P(\tilde{a}'_u, c'|a_u^{w\prime}) = p_h(c'_i) \cdot (1 - p_{err})$; otherwise $P(\tilde{a}'_u, c'|a_u^{w\prime}) = p_{err}/(|\mathcal{A}_u^w| - 1)$.

The reward function provided a large positive reward ($+12.5 \cdot W$) for taking a correct submit action; a large penalty ($-12.5 \cdot W$) for taking an incorrect submit action; and a host of smaller penalties ranging from -1 to -3 depending on the appropriateness of information gathering actions and the grounding state given in s_d.

A CSPBVI-based dialog manager requires a heuristic which chooses among actions nominated by each slot. For the MULTISLOT application, this heuristic first looks for an *ask* action by considering the slots in order; if it doesn't find one, it then looks for a *confirm* action again considering the slots in order; and if it doesn't find one then all slots must have nominated the *submit* action, which is selected.

The CSPBVI optimization procedure takes a set of parameters K, N, and T. Experimentation found that no gains in performance were achieved for values beyond $N = 100$, $K = 50$, and $T = 50$, and these values were used for all experiments, below. Figure 6 shows a conversation with a 2-slot version of MULTISLOT with a typical concept error rate ($p_{err} = 0.30$) and a somewhat informative confidence score ($h = 2$).

Results and discussion

First the performance of CSPBVI on the 2-slot MULTISLOT dialog problem was compared to the Summary POMDP method. Whereas CSPBVI constructs local policies, a Summary POMDP constructs a plan over all possible summary states, and in this respect a Summary POMDP should construct better policies. Results are shown in Figure 1. Error bars (here and throughout) show 95% confidence interval for the true average return for 10,000 simulated dialogs. Overall the two methods perform similarly, indicating that the assumptions made by CSPBVI do not prevent it from attaining good policies. The Summary POMDP method was also applied to a 3-slot MULTISLOT but was unable to find a good policy.

Next, CSPBVI was compared to two MDP baselines *without* confidence score information (i.e., with h = 0). Both MDPs used a state estimator which received the speech recognition hypothesis $\tilde{a}_u$ as input and tracked whether each slot was *not-stated*, *unconfirmed*, or *confirmed* using basic grounding rules. In all other respects the simulation environment for the MDPs and CSPBVI were identical (e.g., the MDP action set included the same actions as in the CSPBVI action set). The first MDP baseline, "MDP-Full", formed its state space as the cross-product of all MDP slot-states and the second MDP baseline, "MDP-Composite", estimated a separate MDP policy for each slot and used the same heuristic to choose actions at runtime as CSPBVI. Both MDP baselines were trained using Q-learning (Watkins 1989). A variety of learning parameters were explored and the best-performing set were selected: 100,000 training dialogs, initial Q values set to 0, exploration parameter $\epsilon = 0.2$, and learning rate $\alpha = 1/m$, where m is the number of visits to the $Q(s, a)$ being updated. This experiment was repeated for $w = 1 \ldots 5$ at a variety of error rates using the same optimization parameters. Results are shown in Figure 2. When no recognition errors are made (i.e., $p_{err} = 0.00$), the POMDP and MDPs perform identically but where concept recognition errors are made (i.e., $p_{err} > 0$), the POMDP outperforms the MDP. As the number of slots increases, average return declines slightly for all techniques, because eliciting values for more slots results in longer dialogs.

Next, the effect of confidence score was investigated by varying h. For the MDP-Composite baseline, a "confidence bucket" feature was added to the MDP state space representing "hi" and "low" observed confidence scores. A variety of confidence thresholds were explored and it was found that using a threshold of 0.5 produced optimal results for the MDP. Where recognition is not perfect (i.e., $p_{err} > 0$), CSPBVI outperforms the MDP-2 baseline, and as the confidence score becomes more informative (i.e., as h increases), performance at a given concept error rate p_{err} increases for both the CSPBVI and MDP policies.

CSPBVI optimization was then compared to two hand-

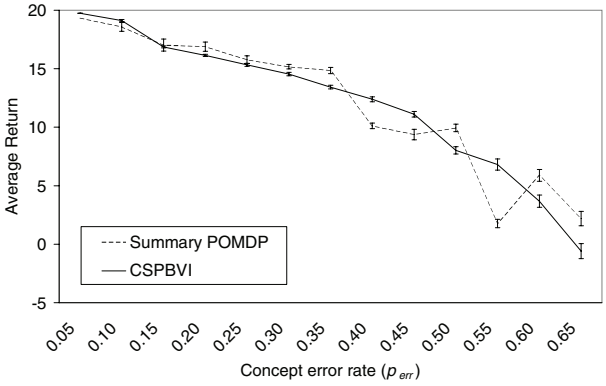

Figure 1: Concept error rate vs. average return for CSPBVI and Summary POMDP baseline.

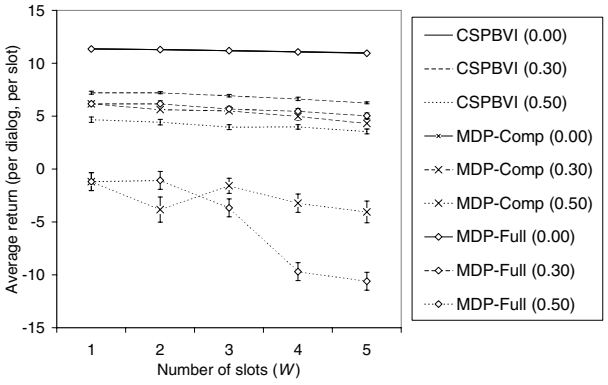

Figure 2: Number of slots (W) vs. average return for CSPBVI and two MDP baselines at various concept error rates.

crafted dialog managers, HC1 and HC2. HC1 and HC2 both use the same state estimator as the "MDP-Composite" baseline. Both HC1 and HC2 took the *ask* action for *not-stated* slots, and the *submit* action for *confirmed* slots. For *unconfirmed* slots, HC1 took the *confirm* action and HC2 took the *ask* action. HC1 and HC2 were evaluated by running 10,000 simulated dialogs for various number of slots and error rates. Results are shown in Figure 4. CSPBVI outperforms both handcrafted controllers at all error rates. As the number of slots increases, the reward gained per slot decreases, but at higher error rates (i.e., $p_{err} = 0.50$) this decline is precipitous for the handcrafted controllers but gradual for the POMDP. One reason for this is that the POMDP is making use of a user model and taking proper account of less likely observations, but the handcrafted policies place equal trust in all observations. As dialogs become longer, the simulated user provides less-reliable information about *other* slots more times in each dialog, causing the performance of handcrafted policies to degrade.

Finally, the effects of mis-estimating the user model were explored. A 5-slot MULTISLOT was trained using the *training* user model and evaluated using the *testing* user model, estimated from the SACTI data as described above. Results are shown in Figure 5. As speech recognition errors

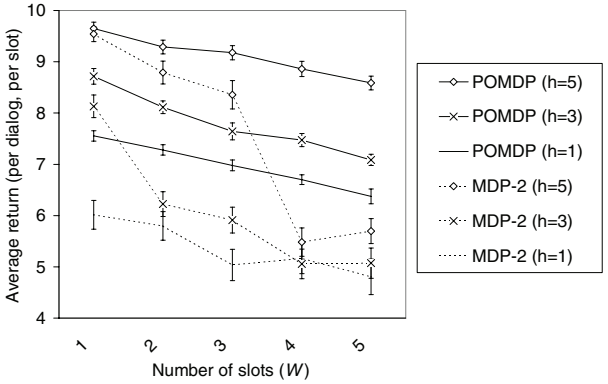

Figure 3: Number of slots (W) vs. average return for CSPBVI and MDP-Composite-2 baseline at various levels of confidence score reliability.

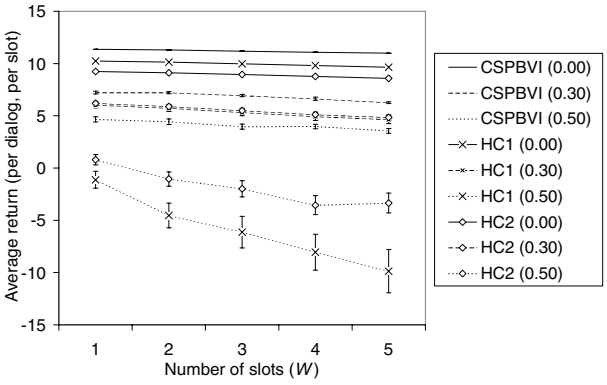

Figure 4: Number of slots vs. average return for CSPBVI and handcrafted baselines at various concept error rates.

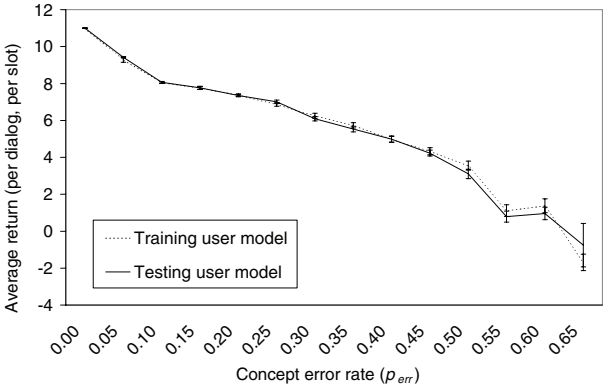

Figure 5: Concept error rate vs. average return for training and testing user models.

increase, the average reward per turn decreases as expected, and in general performance on the test user model is less than but very close to the training user model, implying that the method is reasonably robust to variations in patterns of user behavior or estimation errors in the user model.

Conclusions

CSPBVI enables slot-filling dialog systems cast as SDS-POMDPs to be scaled to handle many slots. In dialog simulation, the scalability gained with localized planning maintains performance gains over baseline techniques while tolerating errors in user model estimation. Future work will conduct trials with real users.

References

Kaelbling, L.; Littman, M.; and Cassandra, A. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101.

Larsson, S., and Traum, D. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering* 5(3/4):323–340.

Roy, N.; Pineau, J.; and Thrun, S. 2000. Spoken dialog management for robots. In *Proc ACL, Hong Kong*.

Stuttle, M.; Williams, J.; and Young, S. 2004. A framework for Wizard-of-Oz experiments with a simulated ASR channel. In *Proc ICSLP, Korea*.

Watkins, C. 1989. *Learning from delayed rewards*. Ph.D. Dissertation, Cambridge University.

Williams, J., and Young, S. 2004. Characterizing task-oriented dialog using a simulated ASR channel. In *Proc ICSLP, Korea*.

Williams, J., and Young, S. 2005. Scaling up POMDPs for dialog management: The "summary POMDP" method. In *Proc ASRU, Puerto Rico*.

Williams, J.; Poupart, P.; and Young, S. 2005a. Factored partially observable Markov decision processes for dialogue management. In *Proc IJCAI Workshop on Knowledge and Reasoning in Practical Dialog Systems, Edinburgh*.

Williams, J.; Poupart, P.; and Young, S. 2005b. Partially observable Markov decision processes with continuous observations for dialogue management. In *Proc SIGDial, Lisbon*.

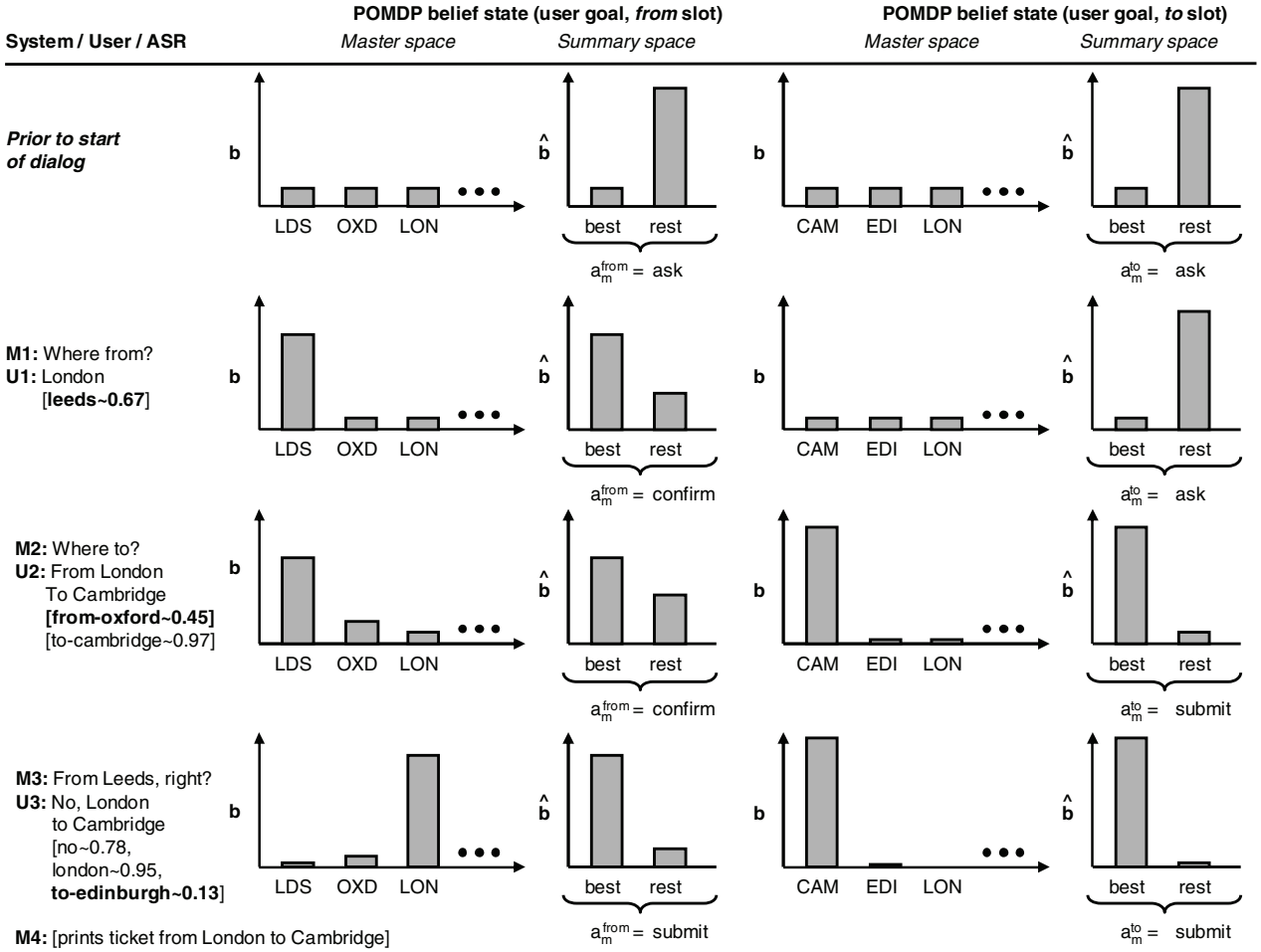

Figure 6: Sample conversation between user and CSPBVI-based dialog manager. LDS=Leeds, OXD=Oxford, LON=London, CAM=Cambridge, and EDI=Edinburgh. a_m^{from} and a_m^{to} indicate actions nominated by each slot. Numbers indicate confidence scores; boldface highlights concept recognition errors. In this example, the user is trying to buy a ticket from London to Cambridge. Initially belief mass is spread over all user goals evenly and both slots nominate the *ask* action. The heuristic examines these two actions and selects the *ask*[*from*]() action (M1). The user's response (U1) of "London" is mis-recognized as "Leeds" with relatively high confidence (0.67), causing a large shift in belief mass toward *leeds* in the *from* slot. For the second machine action (M2), the *from* slot nominates the *confirm* action and the *to* slot nominates the *ask* action, and the heuristic selects *ask*[*to*]() (M2). This process continues until M4, where both slots nominate the *submit* action, the heuristic chooses to submit the form and the dialog ends successfully. Note how POMDP belief monitoring naturally combines the user model with confidence score information – for example, the medium confidence mis-recognition of "from Oxford" in U2 is predicted against by the user model and causes a small shift of belief mass, whereas recognition of "to Cambridge" in U2 is both highly predicted by the user model and receives a high confidence score, causing a massive shift in belief mass.