Applying POMDPs to Dialog Systems in the Troubleshooting Domain

Jason D. Williams
AT&T Labs – Research
180 Park Ave, Building 103
Florham Park, NJ 07932
jdw@research.att.com

Abstract

This paper reports on progress applying partially observable Markov decision processes (POMDPs) to a commercial dialog domain: troubleshooting. In the troubleshooting domain, a spoken dialog system helps a user to fix a product such as a failed DSL connection. Past work has argued that a POMDP is a principled approach to building spoken dialog systems in the simpler slot-filling domain; this paper explains how the POMDPs formulation can be extended to the more complex troubleshooting domain. Results from dialog simulation verify that a POMDP outperforms a handcrafted baseline.

1 Introduction

In the *troubleshooting domain*, a spoken dialog system (SDS) helps a user to restore a malfunctioning product such as a DSL connection to a working state. Building dialog systems for this domain presents several new challenges. First, the user may make mistakes such as misinterpreting the meaning of a status light or pressing the wrong button, so even if no speech recognition errors are made, the user's response may be misleading. Next, in addition to the speech recognizer, input is also received from running network tests such as pinging the user's DSL modem. Input from both sources may contain errors, and a dialog system must cope with conflicting information from two channels. In sum, the dialog system never knows the true state of the product nor the user's true actions, yet must still instruct the user to successfully restore the product to a working state.

Dialog models which explicitly model uncertainty have been shown to significantly outperform baseline models which do not, primarily because they cope better with conflicting evidence introduced by speech recognition errors (Roy et al., 2000; Zhang et al., 2001; Williams and Young, 2007). However, past work has been confined to slot-filling tasks and has not tackled the troubleshooting domain. Conversely, dialog systems for troubleshooting in the literature have not attempted to model uncertainty directly (Grosz and Sidner, 1986; Lochbaum, 1998).

The contribution of this paper is to show how to model a troubleshooting spoken dialog system as a partially observable Markov decision process (POMDP). We argue that past work in the general troubleshooting literature represents simplifications or special cases of a POMDP, then we show how a troubleshooting POMDP can be combined with a dialog system POMDP to create a unified framework that admits global optimization. Experiments with simulated users show how the POMDP formulation effectively balances diagnostic actions (such as a network test) with communicative actions (such as giving the user instructions), and how the POMDP formulation outperforms a hand-crafted baseline both in terms of efficiency and task completion.

This paper is organized as follows. Section 2 reviews POMDPs, the general troubleshooting problem, and POMDP-based spoken dialog systems; section 3 explains how these two POMDPs can be combined to model a troubleshooting spoken dialog system; sections 4-5 present results from simulation; and section 6 concludes.

2 Background

A POMDP is a model for control when there is uncertainty in the effects of actions and in the state of the environment. 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 environment with $s \in \mathbb{S}$; $\mathbb{A}$ is a set of actions $a \in \mathbb{A}$ which operate on the environment; $\mathbb{T}$ 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 $o \in \mathbb{O}$ which describe the state of the environment; $\mathbb{Z}$ defines an observation probability $P(o'|s', a)$; γ is a geometric discount factor $0 \leq \gamma \leq 1$; and b_0 is an initial belief state, defined below.

The POMDP operates as follows. At each time-step, the environment is in some unobserved state s. Since s is not known exactly, a *distribution* over possible states called a *belief state* b is maintained where $b(s)$ indicates the probability of being in a particular state s, with b_0 denoting the initial belief state. Based on b, a control algorithm (also called a *policy*) selects an action a, receives a reward r, and the environment transitions to (unobserved) state s', where s' depends only on s and a. The environment then generates an observation o' which is dependent on s' and a. At each time-step, b is updated as

$$b'(s') = \eta \cdot P(o'|s', a) \sum_{s} P(s'|s, a) b(s) \qquad (1)$$

where η is a normalization constant (Kaelbling et al., 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 $V = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$, where s_t and a_t indicate the state of the environment and the action taken at time t, respectively. The goal of the control algorithm is to choose actions that maximize the expected return $E[V]$ given b and the POMDP parameters $\mathfrak{P}$, and the process of searching for such a control algorithm is called *optimization*.

2.1 Troubleshooting as a POMDP

The goal of the general (non-dialog) problem of automated troubleshooting is for a control algorithm to fix a product by taking a sequence of diagnosis and repair actions. Different actions have different reliabilities and different costs, and the aim is to find the sequence that minimizes the total cost. Since the actions are not completely reliable, the true state of the product can't be known with certainty: for example, an instrument may provide a faulty reading.

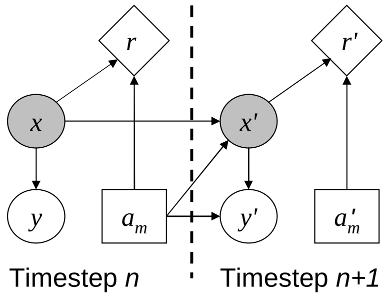

Figure 1: Influence diagram depiction of automated troubleshooting. Round nodes represent random variables, shaded nodes are unobservable and clear nodes are observable. Arcs show conditional dependence. Squares indicate actions, selected by the policy. Diamonds indicate real-valued rewards.

Formalizing this, a product has some hidden state x, which is usually decomposed into components $x = (x_1, x_2, \ldots, x_n)$. A control algorithm takes action a_m, which changes the state of x according to $P(x'|x, a_m)$. The product then produces an observation y according to $P(y'|x', a_m)$. Replacing cost with reward, the control algorithm receives reward $r(x, a_m)$ and the goal is to find the sequence of actions which maximizes the cumulative sum of reward. When viewed in this way, automated troubleshooting can be readily viewed as a POMDP (Shakeri et al., 1997). Figure 1 shows the automated troubleshooting task as an influence diagram.

Although POMDPs are an elegant model for troubleshooting, they are also notoriously difficult to optimize and much of the troubleshooting literature seeks appropriate constraints which render the optimization tractable, such as assuming that each action affects at most one product state component, that actions have deterministic effects, and that there is only fault present (Heckerman et al., 1995). More recently, advances in the POMDP literature have radically increased the scalability of optimization algorithms: for example, Poupart optimizes a substantial network troubleshooting problem cast as a generic POMDP (Poupart and Boutilier, 2004). Viewing troubleshooting as a generic POMDP increases the scope of admissible troubleshooting tasks, and as will be discussed in section 3, this view also allows the uncertainty in the product state to be explicitly modelled in a spoken dialog system.

2.2 Spoken dialog as a POMDP

Past work has argued that POMDPs represent a principled approach to modelling (non-troubleshooting) spoken dialog systems (Roy et al., 2000; Zhang et al., 2001; Williams and Young, 2007). The intuition is that a user's goals and actions form the unobserved state and the (possibly erroneous) ASR result forms the observation. The SDS-POMDP model (Williams and Young, 2007) formalizes this by decomposing the POMDP state variable s into three components, $s = (s_u, a_u, d)$. The component s_u gives the *user's goal*, such as a complete travel itinerary in a travel reservation task. The component a_u gives the most recent *user action* (communicative intent), such as stating a place the user would like to travel to. Finally the component d records relevant *dialog history*, such as the grounding status of a slot. None of these components is observable directly by the dialog system and the SDS-POMDP belief state is formed of a distribution over these components $b(s_u, a_u, d)$. The POMDP action a corresponds to the dialog system action a_m, such as asking the user where they want to go to. Finally, the POMDP observation o is set to $(\tilde{a}_u, c)$, where $\tilde{a}_u$ is the hypothesis of the user's action (communicative intent) provided by the speech recognition and understanding process, and c is the confidence score. Figure 2 shows the SDS-POMDP model as an influence diagram, and also shows the conditional dependencies assumed in the SDS-POMDP model.

3 Troubleshooting SDS-POMDP model

In this section, we develop a statistical model of a troubleshooting dialog system. The formulation begins by taking the union of the state spaces of the dialog POMDP and the troubleshooting POMDP, (s_u, a_u, d, x), and making two modifications. First, it is assumed that the user's goal s_u is known and constant (i.e., to fix the product), and as such does not need to be included. Second, the user's action a_u is decomposed into two components: a_u^{ts} denotes *troubleshooting* actions that are directed toward the product, such as turning a modem on or off, entering a user name or just observing the status lights; and a_u^{com} denotes *communicative* actions to the dialog system such as saying "green" or "yes". Reordering, the combined POMDP state has components:

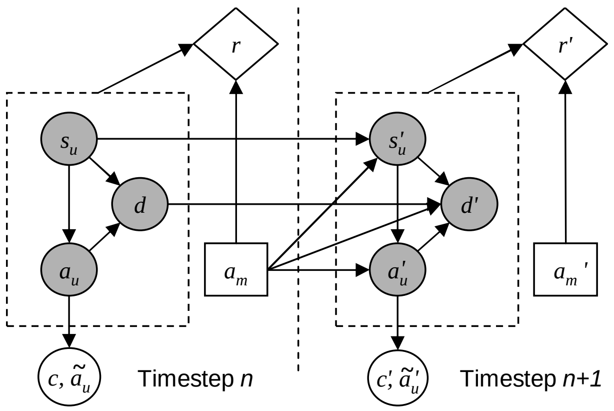

Figure 2: SDS-POMDP model shown as an influence diagram. The dotted box refers to all of the (hidden) POMDP state components.

$$s = (a_u^{ts}, x, a_u^{com}, d). \tag{2}$$

Next, the combined observation is formed of the union of the observations from the dialog and troubleshooting POMDPs:

$$o = (\tilde{a}_u^{com}, c, y). \tag{3}$$

Finally, since the POMDP may choose only one action at each time-step, the POMDP action is simply a_m.

Substituting eq. 2 into the POMDP transition function $P(s'|s, a)$ yields $P(a_u^{ts\prime}, x', a_u^{com\prime}, d'|a_u^{ts}, x, a_u^{com}, d, a_m)$ and is decomposed as follows. First, it is assumed that the user's troubleshooting action $a_u^{ts\prime}$ depends only on the system's action a_m, the previous product state x and the dialog history d. Next, it is assumed that the product state x' depends only on the previous product state x, and the most recent user's and dialog system's troubleshooting actions $a_u^{ts\prime}$ and a_m. Further, the user's communicative action $a_u^{com\prime}$ depends only on the most recent user's troubleshooting action $a_u^{ts\prime}$, product state x', dialog history d and system action a_m. Finally, the dialog history component d' is a function of the previous dialog history d and the most recent user and dialog system actions $a_u^{ts\prime}$, $a_u^{com\prime}$, and a_m. With these assumptions, the combined transition function is:

$$\begin{aligned}&P(a_u^{ts\prime}, x', a_u^{com\prime}, d'|a_u^{ts}, x, a_u^{com}, d, a_m) \approx \\ &\quad P(a_u^{ts\prime}|x, d, a_m) \cdot P(x'|x, a_m, a_u^{ts\prime}) \cdot \\ &\quad P(a_u^{com\prime}|d, a_m, a_u^{ts\prime}, x') \cdot \\ &\quad P(d'|d, a_m, a_u^{ts\prime}, x', a_u^{com\prime})\end{aligned} \tag{4}$$

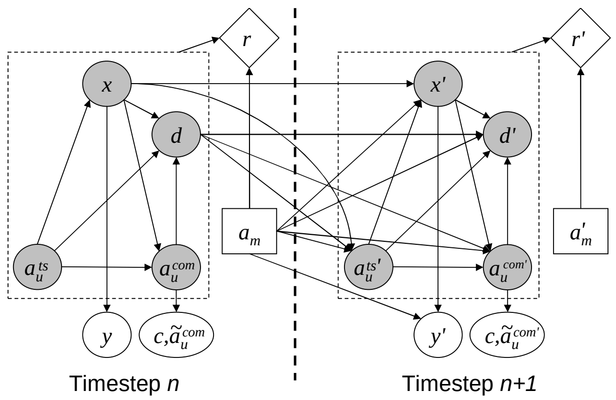

Figure 3: Influence diagram of a troubleshooting spoken dialog system.

Substituting eq. 3 into the POMDP observation function $P(o'|s',a)$ yields $P(\tilde{a}_u^{com\prime}, c', y' | a_u^{ts\prime}, x', a_u^{com\prime}, d', a_m)$. It is assumed that the ASR hypothesis $\tilde{a}_u^{com\prime}$ and confidence score c' depend only on the user's speech in $a_u^{com\prime}$ and that the result of the troubleshooting test (conducted by the dialog system) y' depends only on the state of the product x' and the dialog system's action a_m:

$$\begin{aligned} &P(\tilde{a}_u^{com\prime}, c', y' | a_u^{ts\prime}, x', a_u^{com\prime}, d', a_m) \approx \\ &P(\tilde{a}_u^{com\prime}, c' | a_u^{com\prime}) \cdot P(y | a_m, x') \end{aligned} \tag{5}$$

An influence diagram of the model is shown in Figure 3.

At runtime, a belief state (i.e., distribution) is maintained over the POMDP state variables, $b(a_u^{ts}, x, a_u^{com}, d)$. Based on this belief state the policy chooses an action a_m and receives observation $(\tilde{a}_u^{com\prime}, c', y')$. The belief state is updated by applying Eq 1, and the cycle repeats.

The user action models $P(a_u^{ts\prime}|x, d, a_m)$ and $P(a_u^{com\prime}|d, a_m, a_u^{ts\prime}, x')$ indicate how users are likely to respond in troubleshooting dialogs and can be estimated from annotated dialog data. The product models $P(x'|x, a_m, a_u^{ts\prime})$ and $P(y'|a_m, x')$ indicate how user and dialog system actions change the state of the product and the reliability of tests, and these can be estimated by interviewing domain experts or by examining logs of product performance. As in the SDS-POMDP model, the dialog history model $P(d'|d, a_m, a_u^{com\prime}, x', a_u^{ts\prime})$ can be handcrafted so as to incorporate features from the dialog history which the dialog designer believes are important, such as appropriateness or notions of grounding. The ASR confusion model $P(\tilde{a}_u^{com\prime}, c'|a_u^{com\prime})$ can be estimated from speech recognition data or derived analytically. The reward function can include distinct costs for different diagnostic tests, dialog actions, and for successful/unsuccessful task completion. It is not specified explicitly here since it depends on the application.

4 Illustration: DSL-1

To illustrate the general framework, we first created a very simple troubleshooting spoken dialog system called DSL-1. Table 1 shows the values for all of the variables. In DSL-1, the are just 2 possible problems: *no-power* and *no-network*.

The conditional probability tables composing the model were handcrafted based on conversations with troubleshooting experts and past experience with spoken dialog systems. For example, the model of user's troubleshooting action assumes that the user performs the correct action with $p = 0.9$, doesn't understand with $p = 0.05$, and performs an incorrect action with $p = 0.05$. The model of the user's communicative action assumes that the user provides correct (but possibly incomplete) information with $p = 0.9$, and remains silent with $p = 0.1$.

The model of the product was designed such that the user's *check-power* and *check-network* actions are always effective, but if power is restored there may still be *no-network* with $p = 0.2$.

The model of the speech recognition and understanding process uses a concept error rate of 30%, where errors are uniformly distributed, and no confidence scores are used. For example, when the user expresses the concept *all-ok*, it will be recognized correctly 70% of the time, and will be misrecognized as *no-power* 5% of the time, as *no-network* 5% of the time, etc. The model for y indicates how reliable the *ping* action is, set with a parameter p_{err}: for example if $p_{err} = 0.1$, the result of a ping test will be incorrect 10% of the time. In the experiments below, the value of p_{err} is varied to explore how the POMDP policy trades off between the *ping* action and communicative actions.

The reward function provides $+100$ for taking the *end-call* action when the connection is working, -100 for taking the *done* action when the connection isn't working, and -1 for any communicative or test action. The dialog continues until the dialog

	Variable	Values
State Components	a_u^{ts}	$\{$check-power, check-network, observe, do-nothing, dont-understand$\}$
	x	$\{$all-ok, no-power, no-network$\}$
	d	$\{$start, not-done, done$\}$
	a_u^{com}	$\{$no-power, no-network, power-ok, all-ok, silent, didnt-understand$\}$
Observation Components	$\tilde{a}_u^{com}$	(same set as a_u^{com})
	y	$\{$ping-ok, no-response$\}$
Action	a_m	$\{$ping, ask-working-ok, req-check-power, req-check-network, end-call$\}$

Table 1: Variable values in the DSL-1 simple troubleshooting example.

system takes the *done* action, at which point the dialog is over.

4.1 Results

The POMDP was optimized using a standard algorithm from the literature (Spaan and Vlassis, 2005). This algorithm optimizes the policy at a discrete set of belief points; as more points are added, the quality of the resulting policy improves at the expense of more computation. We found that 300 belief points achieved asymptotic performance. A model was constructed for values of p_{err} ranging from 0.0 to 0.5; each model was optimized and then evaluated using 5000 simulated dialogs.

Results are shown in Figures 4 and 5. In each figure the x-axis is the accuracy of the *ping* action: $p_{err} = 0\%$ indicates that the *ping* action is entirely reliable and $p_{err} = 50\%$ indicates that the *ping* action returns useless noise. In Figure 4, the y-axis shows average return, and in Figure 5, the solid line shows the task completion rate and the dotted line shows the average dialog length. The error bars indicate the 95% confidence interval.

As the error rate for the *ping* action increases from 0% to 20%, the average dialog length increases from 5.1 turns to 6.5 turns, and the successful task completion rate falls from 100.0% to 98.9%. These figures then remain broadly constant from 20% to 50%. In other words, as errors in the ping action increase, dialogs become longer and occasionally the system fails to fix the connection. Inspecting the dialog transcripts showed that at $p_{err} = 0\%$, the policy relies on the *ping* action to judge whether the connection is working. As p_{err} increases, the policy decreasingly employs the *ping* diagnostic action in favor of the *ask-working-ok* communicative action until $p_{err} = 20\%$, at which point the ping action is

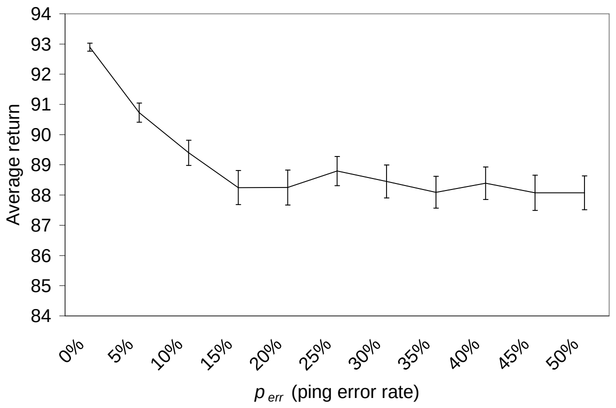

Figure 4: Error rate of the *ping* action vs. reward gained per dialog. As the error rate of the *ping* action is increased, performance declines until the error rate reaches 20%, at which point the system no longer uses the *ping* action.

not used at all. At this point the planning process has determined that the ping action doesn't help produce better dialogs than just interacting with the caller, and the performance from 20% to 50% is constant.[1]

These experiments confirm that, for a very simple troubleshooting dialog system in simulation, the POMDP approach is able to synthesize noisy information gained from communicative and test actions into one unified belief while the underlying, hidden product state is changing. This is an important result because past work that has applied POMDPs to dialog systems has employed a single modality (communicative actions), and have largely had fixed persistent state. Even so, this illustration is much too small to be of practical use, and relies entirely on hand-crafted models of the dynamics. In the next section a model of realistic scale is presented with transition dynamics estimated from real conversa-

[1]The variations in performance between 20% and 50% are due to sampling in the optimization algorithm.

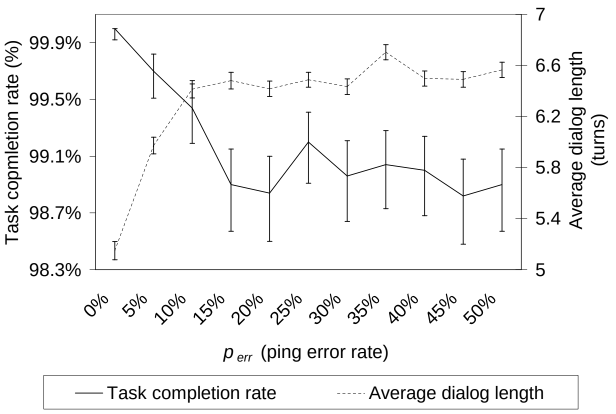

Figure 5: Error rate of the *ping* action vs. successful task completion rate and average dialog length. The left y axis and the solid line show the task completion rate, and the right y axis and the dotted line show the average dialog length in number of turns.

tional data.

5 Illustration: DSL-2

In this section we present a second POMDP-based troubleshooting dialog system called DSL-2 which captures many of the properties of a real-world DSL troubleshooting task. Approximately 100 telephone calls between (human) DSL support agents and customers were monitored, and the observations of these conversations guided creation of the dialog system, including typical problems, agent instructions, and user responses. The product state X was decomposed into 19 components which track, for example, whether there are any outages reported, whether the DSL modem is switched on, and whether the username has been entered correctly in the DSL configuration. Seven of these components can cause the connection to fail: (1) router powered off or crashed, (2) an upstream network crash, (3) a service outage, (4-6) a wrong username, password, or connection type entered in the DSL modem configuration, and (7) an unknown root cause which can't be fixed by the dialog system. Some of the problems can only be identified or fixed by the dialog system (such as a service outage or an upstream network crash), and the rest only by the user (such as a router being off or wrong username entered). The problems may occur in any combination: for example, there may be a service outage while the user's password is entered incorrectly. The system action set (A_m) consisted of 18 actions such as asking the user to turn the modem on, providing the correct username, checking whether any outages have been reported, and rebooting the upstream network interface. The user's troubleshooting action set A_u^{ts} consisted of 12 actions such as turning the modem on or off, opening the DSL configuration screen, entering a password, and attempting to surf to a website. The user's communicative action set A_u^{com} consisted of 11 actions such as saying the color of a light (e.g., "red" or "green"), yes and no, back-channel, silence, and an "out-of-grammar" action which accounts for user speech which cannot be recognized.

The conditional probability tables for each of the product components were handcrafted based on interviews with DSL technicians and are almost all deterministic. For example, if the DSL modem is powered on, the power light will always be on. Next a subset of the agent/user telephone calls were transcribed and annotated with simple dialog acts, and from these the two user models were estimated. Smoothing was applied so that the models allow for the user to take any action at any point in the dialog. Concept recognition errors were generated with $p = 0.30$, and confidence scores were drawn from an exponential distribution such that (at an equal error rate confidence threshold) about half of the concept errors could be identified. The reward function provides $+100$ for ending the dialog having correctly identified (and if possible resolved) the root causes, -100 for ending the dialog with unidentified or unresolved root causes, and -1 for any other action. If a dialog ran for more than 100 turns, it was considered a failure and terminated.

We created a state-based dialog manager by hand (called HC) which broadly reflects the agents' troubleshooting practices and which serves as our baseline. HC consisted of 19 dialog states, where each state specified an action a_m to take (for example to ask the user to turn the modem on), and observations from the speech recognizer $\tilde{a}_u^{com}$ or troubleshooting tests y may cause transitions between dialog states. HC first asks the user to power cycle the modem, then checks for outages and "resets" the upstream network interface, then verifies that the username, password, and network type are configured correctly on the router. After each step HC checks if the connection is working by asking if the network light is green, pinging the modem, then asking the user

	POMDP	HC	HC(0)
CER	30%	30%	0%
N	500	500	500
TCR	96.1%	78.0%	88.6%
Length	19.9	76.5	48.5
Return	73.3	8.13	48.8

Table 2: Results for the POMDP and hand-crafted dialog managers. CER is concept error rate; TCR is task completion rate; Length is measured in turns.

to open a web browser; if any one of these tests fails, troubleshooting resumes, and if they all succeed then HC ends the dialog. If an outage is detected, HC says this and exits, and if the connection still isn't working at the end of the dialog then HC escalates the call to a (human) technician. In general when HC receives an unexpected answer or confidence score below the equal-error rate threshold, it treats this as a likely speech recognition error and remains in the same dialog state.

Next, optimization was performed as described in (Williams et al., 2005). This technique takes as input a POMDP model and a state-based dialog controller, and produces an improved dialog controller. Space limitations prevent a full description here; the intuition is that the algorithm uses the POMDP belief state at runtime to "rewire" the dialog controller to achieve an improvement in reward. Because this optimization algorithm improves a standard state-based dialog controller (in this case the HC baseline), it provides an indication of the value of adding the POMDP machinery.

5.1 Results and discussion

First, 500 simulated dialogs were run with the POMDP, and then 500 simulated dialogs were run with the HC baseline controller. Finally, as a further comparison, the ASR simulation was changed so that no ASR errors were made, and HC was run for 500 dialogs in this configuration, which we call HC(0). Results are shown in Table 2. All of the observed differences are statistically significant ($p \ll 0.01$).

In the presence of speech recognition errors, the POMDP produces dialogs which are significantly shorter and more successful than HC. Moreover, the POMDP, which faced ASR errors, also outperforms HC(0), which did not. Examination of the dialog transcripts found that the main source of failure for HC(0) was exceeding 100 turns. In other words, quantitatively, the POMDP is both more robust to ASR errors and (independent of ASR errors) more efficient.

The dialog transcripts were inspected to determine qualitatively how the POMDP attained better performance. An example is shown in Table 3. At the start of the conversation, the belief (probability) that the connection is working $p(\text{allOk})$ is 56% and the belief that the power to the DSL modem is on $p(\text{pwrOn})$ is 98.0% (these are 2 of the 19 components in the product state x). As the dialog progresses, belief monitoring updates these to account for the evidence received. For example, the unsuccessful *ping* in S1 causes $p(\text{allOk})$ to drop from 56% to 14%. The belief monitoring process also naturally makes use of indirect evidence – for example, in U14 the user indicates the network light is "red": since the network light will only be on if the power to the DSL modem is on, this causes an increase in the belief that the power is on, from 99.1% to 99.8%.

The key benefit of the POMDP approach is that the dialog manager can exploit the belief state to make better progress in the face of low-confidence or even nonsensical replies, without sacrificing overall task completion. For example, in S1 through S9 the POMDP policy differs from the baseline controller: the baseline controller would have ignored the lower-confidence recognitions in U4 and U8, but the POMDP policy moves ahead. When the policy receives a nonsensical reply, for example in U6, it reverts back to an earlier stage of the troubleshooting procedure it had previously skipped. This latter behavior ensures that omitting steps to move faster through the procedure doesn't ultimately sacrifice task completion.

6 Conclusions

This paper has shown how a spoken dialog system for troubleshooting can be cast as a POMDP. The troubleshooting domain has important differences to past applications of the POMDP approach and the two illustrations provided in this paper support our claim that, at least in dialog simulation, the advantages of POMDPs apply to this domain.

After finishing simulation experiments, we installed DSL-2 into a real dialog system, and found that belief monitoring runs slower than real-time. We subsequently developed a method to address this, which we will report on separately in the future, and are now preparing for a pilot study with real users.

References

BJ Grosz and CL Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

D Heckerman, JS Breese, and K Rommelse. 1995. Decision-theoretic troubleshooting. *Communications of the ACM*, 38(3):49–57.

L Kaelbling, ML Littman, and AR Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101.

KE Lochbaum. 1998. A collaborative planning model of intentional structure. *Computational Linguistics*, 24(4):525–572.

P Poupart and C Boutilier. 2004. VDCBPI: an approximate scalable algorithm for large scale POMDPs. In *Proc Advances in Neural Information Processing Systems 17 (NIPS), Vancouver, Canada*, pages 1081–1088.

N Roy, J Pineau, and S Thrun. 2000. Spoken dialog management for robots. In *Proc Association for Computational Linguistics (ACL), Hong Kong*.

M Shakeri, KR Pattipati, V Raghavan, A Patterson-Hine, and DL Iverson. 1997. Multiple fault isolation in redundant systems. Technical Report NCC2-5123, NASA Ames.

MTJ Spaan and N Vlassis. 2005. Perseus: randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220.

JD Williams and SJ Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21(2):393–422.

JD Williams, P Poupart, and SJ Young. 2005. Partially observable Markov decision processes with continuous observations for dialogue management. In *Proc SIGdial Workshop on Discourse and Dialogue, Lisbon*.

B Zhang, Q Cai, J Mao, and B Guo. 2001. Planning and acting under uncertainty: A new model for spoken dialogue system. In *Proc Conf on Uncertainty in Artificial Intelligence (UAI), Seattle, Washington*, pages 572–579.

ID	p(allOk)	p(pwrOn)	Transcript
*S1	56%	98.0%	I'm going to try pinging your DSL modem from here.
S2	-	-	There was no response so it seems your connection isn't working yet.
*S3	14%	96.1%	Please turn the DSL modem on.
U4	-	-	*ok* [ok, 0.004]
*S5	30%	96.4%	What color is the network light on the DSL modem?
U6	-	-	*dontunderstand* [no, 0.052]
*S7	30%	93.8%	Please turn the DSL modem on.
U8	-	-	*dontunderstand* [ok, 0.454]
*S9	31%	99.1%	I'm going to check if there are any outages reported, just a moment.
S10	-	-	There are no outages reported.
S11	39%	99.1%	I'm going to try resetting your connection from here, just a moment.
S12	-	-	Ok, that has been done.
S13	48%	99.1%	What color is the network light on the DSL modem?
U14	-	-	*red* [red, 0.678]
S15	3%	99.8%	Please open the DSL configuration screen.
...	...	...	...

Table 3: Fragment of a conversation with the POMDP dialog manager. Asterisks (*) indicate transitions not in the baseline dialog manager. p(allOk) shows the probability that DSL connectivity is working and p(pwrOn) shows the probability that the power to the DSL modem is on, according to the POMDP belief state. The simulated user's communicative actions are shown in *italics*, followed by the [concept, confidence score] produced by the ASR simulation.