Affective Dialogue Management Using Factored POMDPs

Trung H. Bui, Job Zwiers, Mannes Poel, and Anton Nijholt

Abstract. Partially Observable Markov Decision Processes (POMDPs) have been demonstrated empirically to be good models for robust spoken dialogue design. This chapter shows that such models are also very appropriate for designing affective dialogue systems. We describe how to model affective dialogue systems using POMDPs and propose a novel approach to develop an affective dialogue model using factored POMDPs. We apply this model for a single-slot route navigation dialogue problem as a proof of concept. The experimental results demonstrate that integrating user's affect into a POMDP-based dialogue manager is not only a nice idea but is also helpful for improving the dialogue manager performance given that the user's affect influences their behavior. Further, our practical findings and experiments on the model tractability are expected to be helpful for designers and researchers who are interested in practical implementation of dialogue systems using the state-of-the-art POMDP techniques.

1 Introduction

The HAL 9000 computer character is popular in the speech and language technology research field since his capabilities can be linked to different research topics of the field such as speech recognition, natural language understanding, lip reading, natural language generation, and speech synthesis [20, chap. 1]. This artificial agent is often referred to as a dialogue system, a computer system that is able to talk with humans in a way more or less similar to the way in which humans converse with each other.

Trung H. Bui
Center for the Study of Language and Information, Stanford University,
210 Panama St, Stanford, CA 94305, USA
e-mail: `thbui@stanford.edu`

Job Zwiers, Mannes Poel, and Anton Nijholt
Human Media Interaction Group, University of Twente, Postbus 217,
7500 AE Enschede, The Netherlands
e-mail: `zwiers,mpoel,anijholt@cs.utwente.nl`

Furthermore, HAL is affective[1]. He is able to recognize the affective states of the crew members through their voice and facial expressions and to adapt his behavior accordingly. HAL can also express emotions, which is explained by Dave Bowman, a crewman in the movie:

> Well, he acts like he has genuine emotions. Of course, he's programmed that way to make it easier for us to talk with him.

Precisely, HAL is an "ideally" Affective Dialogue System (ADS), a dialogue system that has specific abilities relating to, arising from, and deliberately influencing people's emotions [30].

Designing and developing ADSs has recently received much interest from the dialogue research community [2]. A distinctive feature of these systems is affect modeling. Previous work mainly focused on showing the system's emotions to the user in order to achieve the designer's goal such as helping the student to practice nursing tasks [17] or persuading the user to change their dietary behavior [33]. A different and more challenging problem is to infer the interlocutor's affective state (hereafter called "user") and to adapt the system's behavior accordingly*.

Solving this problem could enhance the adaptivity of a dialogue system in many application domains. For example, in the information seeking dialogue domain, if a dialogue system is able to detect the critical phase of the conversation which is indicated by the user's vocal expressions of anger or irritation, it could determine whether it is better to keep the dialogue or to pass it over to a human operator [5]. Similarly, many communicative breakdowns in a training system and a telephone-based information system could be avoided if the computer was able to recognize the affective state of the user and to respond to it appropriately [25]. In the intelligent spoken tutoring dialogue domain, the ability to detect and adapt to student emotions is expected to narrow the performance gap between human and computer tutors [6].

This chapter addresses this problem (see *, this page) by introducing a dialogue management system which is able to act appropriately by taking into account some aspects of the user's affective state. The computational model used to implement this system is called the *affective dialogue model*. Concretely, our system processes two main inputs, namely the observation of the user's action (e.g., dialogue act) and the observation of the user's affective state. It then selects the most appropriate action based on these inputs and the context. In human–computer dialogue, building this sort of system is difficult because the recognition results of the user's action and affective state are ambiguous and uncertain. Furthermore, the user's affective state cannot be directly observed and usually changes over time. Therefore, an affective dialogue model should take into account basic dialogue principles, such as turn-taking and grounding, as well as dynamic aspects of the user's affect.

An intuitive solution is to extend conventional methods for developing spoken dialogue systems such as the Rapid Dialogue Prototyping Methodology (RDPM) [12] by integrating an Affect Recognition (AR) module and define a set of rules for the

[1] We use the terms "emotional" and "affective" interchangeably as adjectives describing either physical or cognitive components of the interlocutor's emotion [30, p. 24].

system to act given the observation of the user's affective state (e.g., using the rules defined by Ortony et al. [28]). However, it is nontrivial to handle uncertainty using a pure rule-based approach such as the RDPM.

From recent literature [11, 49] it follows that Partially Observable Markov Decision Processes (POMDPs) are suitable for use in designing these affective dialogue models for three main reasons. First, the POMDP model allows for realistic modeling of the user's affective state, the user's intention, and other user's hidden state components by incorporating them into the state space. Second, recent results in dialogue management research [49] show that the POMDP-based dialogue manager is able to cope well with *uncertainty* that can occur at many levels inside a dialogue system from the automatic speech recognition (ASR) and natural language understanding (NLU) to the dialogue management. Third, the POMDP environment can be used to create a *simulated* user which is useful for learning and evaluation of competing dialogue strategies [37].

In this chapter, we first introduce the basic components of an ADS. Second, we give an overview of the Partially Observable Markov Decision Process (POMDP) techniques and their applications to the dialogue management problem. Third, we describe our factored POMDP approach to affective dialogue modeling. Finally, we address various technical issues when using a state-of-the-art approximate POMDP solver to compute a near-optimal policy for a single slot route navigation application.

2 Components of an Affective Dialogue System

An ADS is a multimodal dialogue system where the user's affective state might be recognized from speech, facial expression, physiological sensors, or combined multimodal input channels. The system expresses emotions through multimodal output channels such as a talking head or a virtual human. Figure 1 shows an architecture of a speech-based ADS. The speech input is first processed by the Automatic Speech Recognition (ASR) module and the semantic meaning is then derived by the Natural Language Understanding (NLU) module. Similarly, the user's affect is interpreted by the AR module and the result is sent to the Dialogue Manager (DM). The DM processes both inputs from the NLU module and the AR module and produces the system action and the system's affective state which are processed by the natural language generation module and the speech synthesis module before sending to the user.

Based on a general statistical Dialogue System (DS) framework presented in [50] and our prototype DSs [12, 14], the interaction between the system and the user is described by the following cycle. The user has a goal g_u in mind before starting a dialogue session with the system. The user's goal g_u might change during the system–user interaction process. At the beginning, the DM sends an action a_s (e.g., informing that the system is ready or greeting the user) and optionally an affective state e_s (whether it is appropriate to show the system's affective state depends on each particular application). Action a_s usually represents the system's intent and is

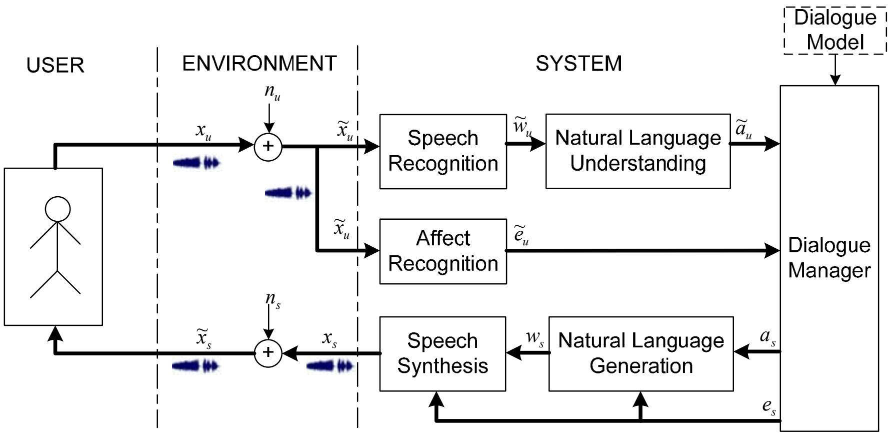

Fig. 1 Components of an affective speech-based dialogue system. Bold arrows show the main flow of the interaction process.

formulated as a sequence of dialogue acts and their associated semantic content. The natural language generation module processes the tuple $< a_s, e_s >$ and realizes a sequence of utterances w_s. The tuple $< w_s, e_s >$ is then processed by the speech synthesis module and the outcome is an audio signal x_s. The signal x_s might be corrupted by the environment noise n_s and the user perceives an audio signal $\tilde{x}_s$. Based on the perceived signal $\tilde{x}_s$, the user infers the system's intent $\tilde{a}_s$ and the affective state $\tilde{e}_s$. The tuple $< \tilde{a}_s, \tilde{e}_s >$ might be different from its counterpart $< a_s, e_s >$ due to a misunderstanding by the user or the corrupted noise n_s from the environment or both. Based on the user's goal g_u, the user forms a communicative action (i.e., intent) a_u. Action a_u might also be influenced by the user's affective state e_u. The user then formulates a sequence of words w_u and articulates a speech signal x_u (see Levelt [22] for further details of the transition from intention to articulation performed by the user). The acoustic signal x_u is processed by both the ASR module and the AR module (the actual input of these modules is, $\tilde{x}_u$, a corrupted signal of x_u caused by the environment noise n_u). The output of the ASR module is a string of words $\tilde{w}_s$. This is then processed by the NLU module and the result is the observation of the user's action $\tilde{a}_u$. The output from the AR module ($\tilde{e}_u$) and NLU module ($\tilde{a}_u$) is sent to the DM. The DM selects a next system's action based on these inputs and the current system's belief b_s. The process is then repeated until either the system or the user terminates the interaction (e.g., the user hangs up the call in a telephone-based dialogue system or the system exits after providing a solution to the user).

The DM is a critical component in the dialogue system, and recent research has shown the advantages of modeling the DM as a POMDP (see Section 1). In the following, we will describe in detail the theory of POMDPs in the dialogue management context.

3 Theory of POMDPs

A POMDP is a generalization of a Markov Decision Process (MDP) which permits uncertainty regarding the state of the environment. Howard [19] described a transition in an MDP as a frog in a pond jumping from lily pad to lily pad. In a POMDP environment, the lily pond is covered by the mist, therefore the frog is no longer certain about the pad it is currently on [27]. Before jumping, the frog can observe information about its current location. This intuitive view is very appropriate to model the affective dialogue management problem as illustrated in Section 3.2.

In a dialogue management context, the agent is the dialogue manager, loosely called the system (Fig. 2). A part of the POMDP environment represents the user's state and user's action. Depending on the design for a particular dialogue application, the rest of the POMDP environment might be used to represent other modules such as the speech recognition and the emotion recognition [10, chap. 1]. Because the user's state cannot be directly observed, the agent uses a state estimator to compute its internal belief (called *belief state*) about the user's current state and an action selector where the policy, called π, is implemented to select actions. The state estimator takes as its input the previous belief state, the most recent system action and the most recent observation, and returns an updated belief state. The action selector takes as its input the agent's current belief state and returns an action that will be sent to the user.

In the following sections, we first describe a basic framework of POMDPs. Then, we present a simple empathic dialogue agent example for the tutoring domain. Two main tasks of the agent, *belief updating* and *finding an optimal policy*, are briefly described. Further details can be found in [10, chap. 2].

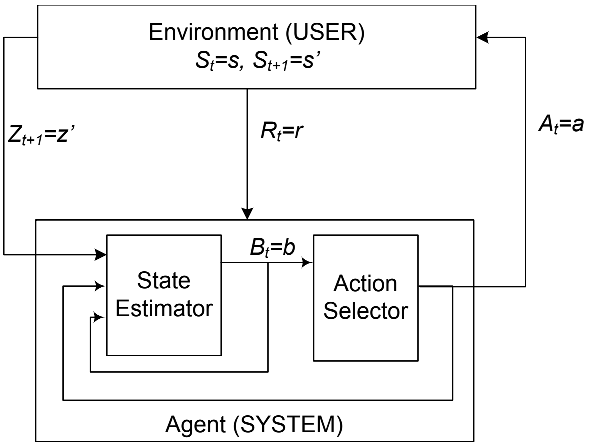

Fig. 2 Modularized view of the interaction between the dialogue manager and the user in a dialogue management context

3.1 Basic Framework

A POMDP [21] is defined as a tuple $\langle S, A, Z, T, O, R\rangle$, where S is a set of states of the environment (usually called *state space*), A is a set of the agent's actions, Z is a set of observations the agent can experience of its environment, T is a *transition function*, O is an *observation function*, and R is a *reward function*. We assume that S, A, Z are finite and that the interaction between the agent and environment follows a sequence of discrete time steps.

Let S_t, A_t, Z_{t+1}, and R_t be random variables taking their values from the sets S, A, Z, and $\mathbb{R}$ (the set of real numbers), respectively. At each time step t, the environment's state is S_t. The agent selects an action A_t and sends it to the environment. The environment's state changes to S_{t+1}. The agent receives an observation Z_{t+1} and a reward value R_t. Following this interaction description, the transition function T, observation function O, and reward function R are formally defined as follows.

- The transition function is defined as $T : S \times A \times S \to [0,1]$. Given any state s and action a, the probability of the next possible state s' is
$$\mathscr{P}^a_{ss'} = T(s,a,s') = P\{S_{t+1} = s' | S_t = s, A_t = a\}, \text{ for all } t. \tag{1}$$
These quantities are called *transition probabilities*. Transition function T is time-invariant and the sum of transition probabilities over the state space $\sum_{s' \in S} \mathscr{P}^a_{ss'} = 1$, for all (s,a).
- The observation function is defined as $O : S \times A \times Z \to [0,1]$. Given any action a and next state s', the probability of the next observation z' is
$$\mathscr{P}^a_{s'z'} = O(s',a,z') = P\{Z_{t+1} = z' | A_t = a, S_{t+1} = s'\}, \text{ for all } t. \tag{2}$$
These quantities are called *observation probabilities*. Observation function O is also time-invariant and the sum of observation probabilities over the observation space $\sum_{z' \in Z} \mathscr{P}^a_{s'z'} = 1$, for all (a,s').
- The reward function[2] is defined as $R : S \times A \to \mathbb{R}$. Given any current state s and action a, the *expected immediate reward* that the agent receives from the environment is
$$R^a_s = R(s,a) \tag{3}$$

Given the POMDP model, we want to design a framework in which the agent's goal is to maximize the *expected cumulative reward* over time

$$V = E\left(\sum_{t=0}^{\mathscr{T}-1} \gamma^t R_t\right) \tag{4}$$

[2] We can also define the reward function as $(R : S \to \mathbb{R})$ or $(R : S \times A \times S \to \mathbb{R})$ or $(R : S \times A \times S \times Z \to \mathbb{R})$. However, the first definition is sufficient and does not change the fundamental properties of the framework.

where $E(.)$ is the mathematical expectation, $\mathscr{T}$ is the *planning horizon* ($\mathscr{T} \geq 1$), γ is a *discount factor* ($0 \leq \gamma \leq 1$). The closer γ to 1, the more effect future rewards have on current agent action selection. This framework is called *finite-horizon* optimality. When $\mathscr{T} \to \infty$ and $\gamma < 1$, the framework is called *infinite-horizon* optimality. It is necessary to set the value of discount factor γ smaller than one in the infinite-horizon case to guarantee that the expected cumulative reward is bounded.

The state space might also contain some special *absorbing state* that only transitions to itself and the reward gained when the agent takes any action is 0. The absorbing state is useful for modeling finite-horizon POMDPs where the number of horizons cannot be determined in advance. Suppose s is an absorbing state, the transition function from s is as follows:

$$T(s,a,s') = \begin{cases} 1 \text{ if } s' = s \\ 0 \text{ otherwise} \end{cases} \text{ and } R(s,a) = 0, \text{ for all } a.$$

Let R_{min} and R_{max} be the lower bound and upper bound of the reward function, that is to say

$$R_{min} < R(s,a) < R_{max}, \text{ for all } (s,a). \tag{5}$$

From (4) and (5), we have for $\gamma < 1$ that:

$$E\left(\sum_{t=0}^{\infty} \gamma^t R_{min}\right) < V < E\left(\sum_{t=0}^{\infty} \gamma^t R_{max}\right) \Rightarrow \frac{R_{min}}{1-\gamma} < V < \frac{R_{max}}{1-\gamma}. \tag{6}$$

3.2 Empathic Dialogue Agent Example

To illustrate the main principle of a POMDP-based dialogue management, we use a simple empathic dialogue agent example for the tutoring domain, which is described as follows. A student ("the user") is interacting with the agent to learn a subject matter. The agent tries to infer the user's affective state to give an appropriate empathic feedback which aims to enhance the student's learning. Suppose that the user's affective state is either $s_1 = pos$ (positive) or $s_2 = neg$ (negative). The agent's goal is to select the most appropriate action from the following three actions: $a_1 = comfort, a_2 = check$, and $a_3 = encourage$. The *comfort* action is expressed by saying "I am sorry that you feel bad about the last question". The *check* action is the agent action to infer the user's affective state from outcome of an affect recognition module assumed to be available. The *encourage* action is expressed in the verbal form such as "Very good!" or "Well done!". If the agent knows exactly the user's true affective state, the action selection problem is trivial. The agent just selects the *encourage* action when the user's affective state is positive and the *comfort* action otherwise. Unfortunately, the user's affective state cannot be directly observed. Therefore, the agent must sometimes execute the *check* action to infer the user's affective state.

A POMDP model for this problem is represented by: (i) $S = \{s_1,s_2\} = \{pos,neg\}$, (ii) $A = \{a_1,a_2,a_3\} = \{comfort, check, encourage\}$, and (iii) $Z = \{z_1,z_2\} = \{\tilde{pos},\tilde{neg}\}$. The transition function, observation function, and reward function are shown in Table 1. All transition and observation probabilities are handcrafted based on the common sense knowledge from the tutoring domain and affect recognition literature.

Table 1 Transition function, observation function, and reward function for the empathic dialogue agent

	$P(s'\|a,s)$			$P(z'\|a,s')$			$R(s,a)$	
a	s	$s' = pos$	$s' = neg$	s'	$z' = \tilde{pos}$	$z' = \tilde{neg}$	s	r
comfort	*pos*	0.80	0.20	*pos*	0.5	0.5	*pos*	-10
	neg	0.30	0.70	*neg*	0.5	0.5	*neg*	10
check	*pos*	0.90	0.10	*pos*	0.9	0.1	*pos*	-1
	neg	0.10	0.90	*neg*	0.1	0.9	*neg*	-1
encourage	*pos*	0.95	0.05	*pos*	0.5	0.5	*pos*	5
	neg	0.05	0.95	*neg*	0.5	0.5	*neg*	-10

The transition probability distribution in Table 1 is based on the following intuition. The user's affective state is *dynamic* and might change even without direct intervention from the agent. Therefore, when the agent selects the *check* action, which does not directly influence the user, the user's affective state might change from positive to negative or vice versa with probability 0.1. If the user is in a *pos* state and the agent selects the *encourage* action which is the right one, the user, therefore, remains in a positive state with a high probability (0.95). Vice versa, if the user is in a *neg* state, the *encourage* is not appropriate and therefore the chance that the negative state remains is high (0.95). Similarly, if the user's affective state is negative, the *comfort* action is appropriate and therefore the probability of changing to the positive state is higher than the *check* action.

When the agent selects the *check* action, it can infer the user's affective state with a 90% correctness (Table 1). The correctness rate here is interpreted as the classification accuracy of the affect recognition module. When the agent selects *encourage* or *comfort* action, the observation probabilities are equally distributed.

Reward values are specified by the designer. They represent *what* the designer wants the agent to achieve, not *how* the designer wants it achieved [42, pg. 57]. In this example, we want the agent to select an appropriate action given uncertainty about the user's affective state. When the user's affective state is positive, the *encourage* action is appropriate and therefore receives a positive reward. When the user's state is negative, the *comfort* action is appropriate and a positive reward is assigned for this action. When the agent selects *check* action, it incurs a small negative reward and in return the agent is more certain about the user's current affective state.

3.3 Computing Belief States

The state of the user cannot be directly observed. Therefore, in order to select good actions, the agent needs to maintain a complete trace of all the observations and actions that have happened so far. This trace is known as a *history*[3]. It is formally defined as:

$$H_{t+1} := \{A_0, Z_1, ..., Z_t, A_t, Z_{t+1}\}, \tag{7}$$

Astrom [3] showed that history H_{t+1} can be summarized via a *belief distribution*. A belief distribution is exactly the belief state of the agent.

$$B_{t+1}(s') = P\{S_{t+1} = s' | B_0, H_{t+1}\}, \tag{8}$$

Assuming the Markov property and using Bayes' rule, Equation 8 is transformed to the following equation (see the proof in [39, Appendix A]):

$$B_{t+1}(s') = P\{S_{t+1} = s' | Z_{t+1} = z', Z_t = a, B_t = b\} \tag{9}$$

Formally, let the belief space B be an infinite set of belief states. A belief state $b \in B$ is encoded as a $|S|$-dimensional column vector $(b_1, b_2, ..., b_{|S|})^T$, where each element $b_i = b(s_i)$ is the probability that the current state of the environment is s_i. Geometrically, a belief state is a point (called *belief point*) in a $(|S|-1)$-dimensional belief simplex.

Concretely, the agent starts with an initial belief state $B_0 = b_0$. At time t, the agent's belief is $B_t = b$, it selects action $A_t = a$ and sends this to the environment. The state changes to $S_{t+1} = s'$. State S_{t+1} cannot be directly observed and the agent only gets observation $Z_{t+1} = z$. The agent also receives a reward $R_t = r$, the value of which depends on the actual values of state s and agent's action a. At this moment, the agent needs to update its belief state $B_{t+1} = b'$ given known values for b, a, z. Starting from Equation 9, $b'(s')$ is computed using the basic laws from the probability theory as follows (see [10, chap. 2] for further details):

$$b'(s') = P(s'|z, a, b) = \eta \mathscr{P}^a_{s'z} T^a_{s'} b, \tag{10}$$

where $T^a_{s'}$ is a $|S|$-dimensional row vector:

$$T^a_{s'} = \left(\mathscr{P}^a_{s_1 s'}, ..., \mathscr{P}^a_{s_{|S|} s'} \right), |S| \text{ is the number of elements of } S.$$

$\eta = 1/P(z|a,b)$ is a normalizing constant, independent of state s'.

The belief state b' is represented as

$$b' = \eta W^a_z b \tag{11}$$

where W^a_z is a $|S| \times |S|$ matrix,

[3] In a dialogue management context, this trace is the dialogue history.

$$W_z^a = \begin{bmatrix} \mathscr{P}^a_{s_1 z}\mathscr{P}^a_{s_1 s_1} & \mathscr{P}^a_{s_1 z}\mathscr{P}^a_{s_2 s_1} & \dots & \mathscr{P}^a_{s_1 z}\mathscr{P}^a_{s_{|S|} s_1} \\ \mathscr{P}^a_{s_2 z}\mathscr{P}^a_{s_1 s_2} & \mathscr{P}^a_{s_2 z}\mathscr{P}^a_{s_2 s_2} & \dots & \mathscr{P}^a_{s_2 z}\mathscr{P}^a_{s_{|S|} s_2} \\ \dots & \dots & \dots & \dots \\ \mathscr{P}^a_{s_{|S|} z}\mathscr{P}^a_{s_1 s_{|S|}} & \mathscr{P}^a_{s_{|S|} z}\mathscr{P}^a_{s_2 s_{|S|}} & \dots & \mathscr{P}^a_{s_{|S|} z}\mathscr{P}^a_{s_{|S|} s_{|S|}} \end{bmatrix} \tag{12}$$

Given the current belief state the agent has to execute the optimal action, i.e. determining a policy that optimizes the rewards received. How an optimal policy is computed will be described in the next section.

3.4 Finding an Optimal Policy

A policy is a function:

$$\pi(b) \longrightarrow a, \tag{13}$$

where b is a belief state and a is the action chosen by the policy π.

An optimal policy π^* is a policy that maximizes the expected cumulative reward:

$$\pi^* = argmax_\pi E\left[\sum_{t=0}^{\infty} \gamma^t R_t\right], \tag{14}$$

where R_t is the reward when the agent follows policy π.

We define value functions $V_i : B \longrightarrow \mathbb{R}$. $V_n(b)$ is the maximum expected cumulative reward when the agent has n remaining steps to go. Its associated policy is denoted by π_n. When the agent has only one remaining step to go (i.e. $n = 1$), all it can do is to select an action and send it to the environment, we have:

$$\begin{aligned} V_1(b) &= \max_{a\in A} \sum_{s\in S} R(s,a)b(s) \\ &= \max_{a\in A} r_a b, \end{aligned} \tag{15}$$

where r_a is a row vector, $r_a = \left(R^a_{s_1},\dots,R^a_{s_{|S|}}\right)$.

When the agent has n remaining steps to go ($n > 1$), the value function V_n is defined inductively as [39]:

$$V_n(b) = \max_{a\in A}\left[r_a b + \gamma \sum_{z\in Z} P(z|a,b) V_{n-1}(b_a^z)\right] \tag{16}$$

where b_a^z is the belief state of the agent after selecting action a, and the observation of the environment changes to z.

When $n \longrightarrow \infty$, the optimal value function for the infinite-horizon case is denoted by V^*. Puterman [32, Theorem 6.9] proved that V_n converges to V^* when n goes to infinity. Therefore, from Equation 16 we have:

$$V^*(b) = \max_{a \in A} \left[r_a b + \gamma \sum_{z \in Z} P(z|a,b) V^*(b_a^z) \right] \tag{17}$$

For any positive number ε, the policy π_n is *ε-optimal* if

$$V^*(b) - V_n(b) \leq \varepsilon \text{ for all } b \in B. \tag{18}$$

Equation 16 is used to develop an important type of algorithm called Value Iteration (VI), which is an algorithm for finding *ε-optimal* policies. The approximation terminates when:

$$\sup_b |V_n(b) - V_{n-1}(b)| \leq \frac{\varepsilon(1-\gamma)}{2\gamma}, \tag{19}$$

where $\sup|X|$ stands for supremum norm of set X [32]. The left part of Equation 19 is called the *Bellman residual*.

Because there are an infinite number of belief states, we cannot compute V_{n-1} directly for each belief state b. Sondik [40] proved that V_{n-1} can be represented through a finite set of α-vectors $\Gamma_{n-1} = \{\alpha_1,...,\alpha_{|\Gamma_{n-1}|}\}$, where each vector $\alpha \in \Gamma_{n-1}$ is a $|S|$-dimensional row vector (also called a *hyperplane*, hereafter it is called an *α-vector*), and

$$V_{n-1}(b) = \max_{\alpha \in \Gamma_{n-1}} \alpha b \tag{20}$$

Therefore, from Equations 11 and 20 we can rewrite Equation 16 as

$$\begin{aligned}
V_n(b) &= \max_{a \in A} \left[r_a b + \gamma \sum_{z \in Z} P(z|a,b) \max_{\alpha \in \Gamma_{n-1}} \alpha b_a^z \right] \\
&= \max_{a \in A} \left[r_a b + \gamma \sum_{z \in Z} P(z|a,b) \max_{\alpha \in \Gamma_{n-1}} \alpha \frac{W_z^a b}{P(z|a,b)} \right] \\
&= \max_{a \in A} \left[r_a b + \gamma \sum_{z \in Z} \max_{\alpha \in \Gamma_{n-1}} \alpha W_z^a b \right] \\
&= \max_{a \in A} \left[r_a b + \gamma (\max_{l_1} \alpha_{l_1} . W_{z_1}^a b + ... + \max_{l_{|Z|}} \alpha_{l_{|Z|}} W_{z_{|Z|}}^a b) \right] \\
&= \max_{a \in A} \left[r_a b + \gamma \max_{l_1} ... \max_{l_{|Z|}} (\alpha_{l_1} W_{z_1}^a b + ... + \alpha_{l_{|Z|}} W_{z_{|Z|}}^a b) \right] \\
&= \max_{a \in A} \left[r_a b + \gamma \max_{l_1} ... \max_{l_{|Z|}} \sum_{k=1}^{|Z|} \alpha_{l_k} W_{z_k}^a b \right] \\
&= \max_{a \in A} \max_{l_1} ... \max_{l_{|Z|}} \left[r_a + \gamma \sum_{k=1}^{|Z|} \alpha_{l_k} W_{z_k}^a \right] b,
\end{aligned} \tag{21}$$

where $l_1, l_2, ..., l_{|Z|} \in [1, |\Gamma_{n-1}|]$.

The set Γ_n can now be generated from set Γ_{n-1} by the following update:

$$\Gamma_n \leftarrow \alpha' = r_a + \gamma \sum_{k=1}^{|Z|} \alpha_{l_k}.W_k^a, \forall a \in A, \alpha_{l_k} \in \Gamma_{n-1}, \tag{22}$$

where $n \geq 1$ and $\Gamma_1 = \{r_{a_1}, r_{a_2}, ..., r_{a_{|A|}}\}$.

Finding the optimal policy (for planning horizon $\mathscr{T} = n$) is now considered as solving a set of $|A||\Gamma_{n-1}|^{|Z|}$ linear constraints derived from Equation 21. To gain the computational tractability, it is necessary to keep only the vectors that contribute to the optimal value function because the number of α-vectors generated from Equation 22 is very large. We distinguish two types of α-vectors: *useful* vectors and *extraneous* vectors [54]. A vector $\alpha \in \Gamma_n$ is useful[4] if:

$$\exists b \in B : \alpha b > \alpha' b, \text{for all } \alpha' \in \Gamma_n - \alpha \tag{23}$$

A vector $\alpha' \in \Gamma_n$ that does not satisfy Equation 23 is an extraneous vector. A set Γ_n that is composed of useful vectors is called a *parsimonious* set [53]. From Equation 20 it is obvious that we can safely remove all the extraneous vectors from the set Γ_n. Monahan [27] proposed a procedure to remove extraneous vectors by solving the following linear program for each $\alpha \in \Gamma_n$:

$$\begin{aligned} &\textit{variables: } x, b_i, \forall i \in [1, |S|] \\ &\textit{maximize } x \\ &\textit{subject to constraints: } b(\alpha - \alpha') \geq x; \forall \alpha' \in \Gamma_n \ \& \ \sum_{i=1}^{|S|} b_i = 1 \end{aligned} \tag{24}$$

If $x < 0$, remove α from Γ_n.

When the set of useful α-vectors Γ_n is found. The agent's action $\hat{a}$ is determined as $\hat{a} \leftarrow \hat{\alpha}$[5], where

$$\hat{\alpha} = \arg\max_{\alpha \in \Gamma_n} \alpha b \tag{25}$$

4 Review of the POMDP-Based Dialogue Management

In this section, we focus on describing the POMDP-based dialogue management approaches. Reviews of other dialogue management approaches for spoken and multimodal dialogue systems are reviewed in McTear [26] and Bui [9], respectively.

Section 3 explained the basic activity of a POMDP-based dialogue management system. Young et al. [51] have argued that nearly all existing dialogue management systems, especially those based upon the information state approach [43], can be considered as direct implementations of the POMDP-based model with a *deterministic* (i.e., handcrafted) dialogue policy. These systems have a number of "severe

[4] Assume that the identical vectors in Γ_n are merged.

[5] Because each α-vector is associated with only one action, see Equation 22.

weaknesses" such as using unreliable confidence measures, having difficulty coping with the dynamic changing of the user's goal and intention. Moreover, tuning the dialogue policy is labor extensive, based on off-line analysis of the system logs [51].

The first work that applies the POMDP for the dialogue management problem was proposed by Roy et al. [34] for building a nursing home robot application. In this application, a flat POMDP model is used where the states represent the *user's intentions*; the observations are the *user's speech utterances*; and the actions are the *system responses*. They showed that the POMDP-based DM copes well with noisy speech utterances, for example their POMDP-based DM makes fewer mistakes than an MDP-based DM and it automatically adjusts the dialogue policy when the quality of the speech recognition degrades. Zhang et al. [52] extended the Roy model in several dimensions: (1) a factored POMDP [7] is deployed for the state and observation sets, (2) the states are composed of the *user's intentions* and *"hidden system states"*, (3) the observations are the *user's utterances* and other observations being inferred from *lower-level information of the speech recognizer, robust parser, and from other input modalities*. Williams et al. [47, 48] further extended the Zhang model by adding the state of the dialogue from the perspective of the user which is hidden from the system's view (called *user's dialogue state*) to the state set and adding the confidence score into the observation set. All these approaches have shown that POMDP-based dialogue strategies outperform MDP counterparts (e.g., Pietquin [31]). Furthermore, these strategies cope well with different types of errors in a Spoken Dialogue System (SDS), especially with ASR errors. Table 2 describes characteristics of these POMDP-based DMs.

Table 2 Characteristics of some POMDP-based dialogue managers (n is the number of slots)

Application	$n,\|S\|,\|A\|,\|Z\|$	Algorithm	Reward function
Nursing home robot [34]	4, 13, 20, 16	AMDP [35]	If the system action is labeled as *correct*: 100, *ok*: -1, *wrong*: -100.
Tour guide [52]	3, 40, 18, 25	QMDP [24], FIB [16]	If the answer matches user's request, the reward is positive. Otherwise, the reward is negative.
Travel booking [45]	2, 36, 5, 5	Perseus [41]	If the system action & dialogue state is *ask & not stated*: -1, *ask & stated*: -2, *ask & confirmed*: -3, *confirmed & not stated*: -3, *confirmed & stated*: -1, *confirm & confirmed*: -2. If the user's goal is determined correctly: 50, incorrectly: -50.

5 The Factored POMDP Approach

Extending the Williams model [47], we represent our affective dialogue model as a factored POMDP [7]. Factored representation of the state set and observation set allows for a natural and intuitive way to encode the information state of the dialogue domain. For example, a goal-oriented dialogue system needs to encode at least variables such as the user's actions and the user's goals. In our model, the state set is composed of the user's goal (Gu), the user's affective state (Eu), the user's action (Au), and the user's grounding state (Du) (similar to the user's dialogue state described in [47]). The observation set is composed of the observations of the user's action (OAu) and the observations of the user's affective state (OEu). Depending on the complexity of the application's domain, these features can be represented by more specific features. For example, the user's affective state can be encoded by continuous variables such as *valence* and *arousal* and can be represented using a continuous-state POMDP [8]. The observation of the user's affective state might be represented by a set of observable features such as response speech, speech pitch, speech volume, posture, and gesture [4]. Similarly, a continuous-observation POMDP [44] can be used to incorporate the continuous-valued features into the observation space.

Figure 3b shows the structure of our affective dialogue model. The features of the state set, action set, observation set, and their dependencies form a two time-slice Dynamic Decision Network (2TDN). Technically, a factored POMDP is equivalent to a 2TDN. Implicitly, some assumptions are made in this model: the user's goal only depends on the user's goal in the previous slice and the system action from the previous slice only influences the user's emotion, the user's action, and the grounding state. We can easily modify this model for representing other dependencies, for example the dependency between the user's emotion and the observation of the user's action. Parameters $p_{gc}, p_{ae}, p_{ec}, p_e, p_{oa}$, and p_{oe} are used to produce handcrafted transition and observation models in case no real data is available (e.g.,

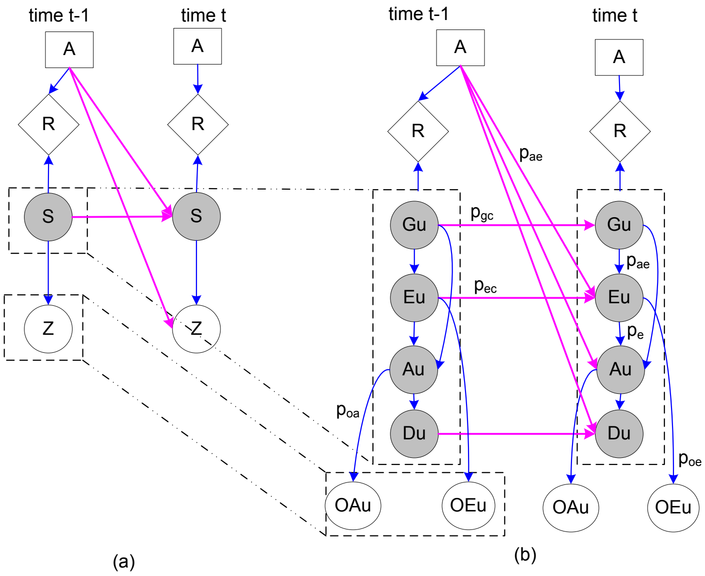

Fig. 3 (a) Standard POMDP and (b) Two time-slice of factored POMDP for the dialogue manager

at the initial phase of the system development), where p_{gc} is the probability of the user goal change; p_{ae} is the probability of the user's emotional change because of the influence of the system action such as when the system confirms an incorrect user's goal (represented by two causal links from the system action and user's goal to the user's affective state); p_{ec} is the probability that the user emotion change is due to the emotion decay and other causes; p_e is the probability of an error in the user's action being induced by emotion; p_{oa} and p_{oe} are the probabilities of the observation error of the user's action and the observation error of the user's affective state, respectively. The reward function is in principle different for each particular application. Therefore, it is not specified in our general affective dialogue model.

Suppose the set of user's goals has m values which are represented by $Gu = \{v_1, v_2, ..., v_m\}$. The features of S and Z and the action set A are formulated as follows:

- $Eu = \{neutral, stress, frustration, anger, happiness, ...\}$.
  Note that we can extend the representation of the user's emotion by adding more relevant features into the state space. For example, if the user's emotion is described by two dimensions *valence* and *arousal*. E_u then becomes a sub-network with two continuous variables.
- $Au = \{answer(v), yes, no, ...\}$, where $v \in Gu$.
  The abstract format of Au is $userSpeechAct(v)$, where *userSpeechAct* is an element of the set of the user's speech acts.
- $Du = \{notstated, stated, confirmed, ...\}$.
- $OAu = \{answer(v), yes, no, ...\}$, where $v \in Gu$.
  The value $y \in OAu$ depends on the level of abstraction of the observation of the user's action. For example, if the observation of the user's action is sent by the ASR module, y is the word-graph or N-best hypotheses of the user's utterance. In our model, we assume a high level of abstraction for the observation of the user's action such as the output from a dialogue act recognition module or the intention level in the simulated user model [15]. In the latter case, *OAu* has the same set of values as Au.
- $OEu = \{neutral, stress, frustration, anger, happiness, ...\}$.
  Similar to the observation of the user's action, the observation of the user's affective state can be represented by a set of observable effects such as *response speed*, *speech pitch*, *speech volume*, *posture*, and *gesture* features [4]. In our current model, we assume that the observations of the user's affective states are the output of an AR module and therefore *OEu* has the same set of values as Eu.
- $A = \{ask, confirm(v), ...\}$, where $v \in Gu$.
  The abstract format of A is $systemSpeechAct(v)$, where $systemSpeechAct$ is an element of the set of the system speech acts.

For a random variable X, we denote x and x' as the values of X at time $t-1$ and t, respectively. Based on the network structure shown in Figure 3b, the transition function is represented compactly as follows:

$$\mathscr{P}^a_{ss'} = P(g'_u|g_u)P(e'_u|a, e_u, g'_u)P(a'_u|a, g'_u, e'_u)P(d'_u|a, d_u, a'_u). \tag{26}$$

$P(g'_u|g_u)$ is called the *user's goal model*, $P(e'_u|a,e_u,g'_u)$ is called the *user's emotion model*, $P(a'_u|a,g'_u,e'_u)$ is called the *user's action model*, and $P(d'_u|a,d_u,a'_u)$ is called the *user's grounding state model*. The observation function is as follows:

$$\mathscr{P}^a_{s'z'} = P(\tilde{a}'_u|a'_u)P(\tilde{e}'_u|e'_u), \tag{27}$$

where $\tilde{a}'_u \in OAu$ and $\tilde{e}'_u \in OEu$. $P(\tilde{a}'_u|a'_u)$ is called the *observation model of the user's actions* and $P(\tilde{e}'_u|e'_u)$ is called the *observation model of the user's emotions*.

6 User Simulation

The DM that we have described is a statistical DM. Dialogue corpora are usually used to train this type of DMs. However, the (PO)MDP-based DM has a huge number of states, therefore it is almost impossible to learn an optimal policy directly from a fixed corpus, regardless of its size [37]. To solve this problem, user simulation techniques have been used [15, 23, 31, 36, 38]. The main idea is a two-phase

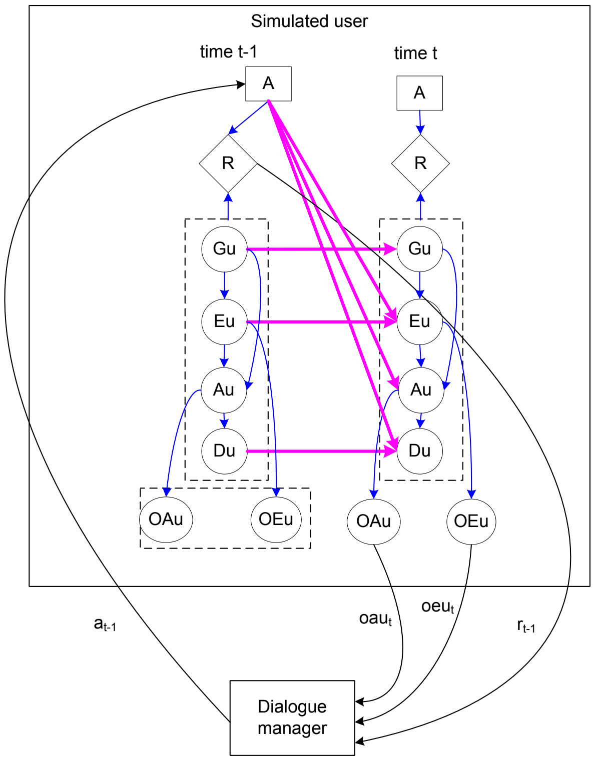

Fig. 4 Simulated user model using the Dynamic Decision Network (DDN). The user's state, action at each time-step are generated from the DDN. Only the observation of the user's action, affective state, and the reward are sent to the dialogue manager.

approach. A simulated user is first trained on a small human–computer dialogue corpus to learn responses of a real user given the dialogue context. The learning DM then interacts with this simulated user in a trial and error manner to learn an optimal dialogue strategy. Experimental results show that a competitive dialogue strategy can be learnt even with handcrafted user model parameters [51]. Recent work also demonstrated that user simulation can be used for testing dialogue systems in early phases of the iterative development cycle [1].

Our simulated user model, constructed based on the POMDP environment, is shown in Figure 4. The structure of this model is similar to the structure of the POMDP model (Fig. 3b), except that the state feature nodes (i.e., Gu, Eu, Au, and Du) in the simulated user model are observable from the user's perspective. If a corpus is available, we can use it to train the model structure and the parameters.

The process to generate observations of the user's actions and of the user's affective states is as follows: First, the value a_{t-1} from the DM is updated on node A of the time-slice $t-1$, the reward r_{t-1} is identified from node A of time-slice $t-1$. Second, the user's goal, affective state, action, and dialogue state are randomly generated based on the probability distributions of the nodes Gu, Eu, Au, and Du (of time-slice t), respectively. Third, the network is updated and the observation of the user's action oau_t and affective state oeu_t are randomly selected based on the probability distribution of nodes OAu and OEu. The tuple $< r_{t-1}, oau_t, oeu_t >$ is sent back to the DM.

7 Example: Single-Slot Route Navigation Example

We illustrate our affective dialogue model described in Section 5 by a simulated toy route navigation example: "A rescue worker (denoted by "the user") needs to get a route description to evacuate victims from an unsafe tunnel. To achieve this goal, he communicates his current location (one of m locations) to the system. The system can infer the user's stressed state and uses this information to adapt its dialogue policy."

In this simple example, the system can *ask* the user about their current location, *confirm* a location provided by the user, show the route description (*ok*) of a given location, and stop the dialogue (i.e., execute the *fail* action) by connecting the user to a human operator. The factored POMDP model for this example is represented by:

- $S = \langle Gu \times Au \times Eu \times Du \rangle \cup end$, where *end* is an absorbing state.
    1. $Gu = \{v_1, v_2, ..., v_m\}$
    2. $Au = \{answer(v_1), answer(v_2), ..., answer(v_m), yes, no\}$
    3. $Eu = \{e_1, e_2\} = \{nostress, stress\}$
    4. $Du = \{d_1, d_2\} = \{notstated, stated\}$
- $A = \{ask, confirm(v_1), ..., confirm(v_m), ok(v_1), ok(v_2), ..., ok(v_m), fail\}$,
- $Z = \langle OAu \times OEu \rangle$.

1. $OAu = \{answer(v_1), answer(v_2), ..., answer(v_m), yes, no\}$
2. $OEu = \{nostress, stress\}$

The full flat-POMDP model is composed of $4m^2 + 8m + 1$ states, $2m + 2$ actions, and $2m + 4$ observations, where m is the number of locations in the tunnel.

The reward model is specified in such a way that an (near) optimal policy helps the user obtain the correct route description as soon as possible and maintains the dialogue appropriateness [47]. Concretely, if the system *confirms* when the user's grounding state is *notstated*, the reward is -2, the reward is -3 for action *fail*, the reward is 10 when the system gives a correct solution (i.e., the system action is $ok(x)$ where x is the user's goal), otherwise the reward is -10. The reward for any action taken in the absorbing *end* state is 0. The reward for any other actions is -1. Designing a reward model that leads to a good dialogue policy is a challenging task. It requires both expert knowledge and practical debugging [13].

The probability distributions of the transition function and observation function are generated using the parameters $p_{gc}, p_{ac}, p_{ec}, p_e, p_{oa}, p_{oe}$ defined in Section 5 and two other parameters K_{ask} and $K_{confirm}$, where K_{ask} and $K_{confirm}$ are the coefficients associated with the *ask* and *confirm* actions. We assume that when the user is stressful, he will make more errors in response to the system *ask* action than the system *confirm* action because in our current model, the number of possible user actions in response to *ask* (`m` possible actions: $answer(v_1), answer(v_2), ..., answer(v_m)$) is greater than to *confirm* (2 actions: yes, no).

Concretely, the handcrafted models of the transition, observation and reward function are described as follows. The user's goal model is represented by:

$$P(g'_u|g_u) = \begin{cases} 1 - p_{gc} & \text{if } g'_u = g_u, \\ \frac{p_{gc}}{|G_u|-1} & \text{otherwise,} \end{cases} \tag{28}$$

where g_u is the user's goal at time $t-1$, g'_u is the user's goal at time t, $|G_u|$ is the number of the user's goals. This model assumes that the user does not change their goal at the next time step with the probability $1 - p_{gc}$.

The user's stress model:

$$P(e'_u|a, e_u, g'_u) = \begin{cases} 1 - p_{ec} & \text{if } e'_u = e_u \text{ and } a \in X, \\ p_{ec} & \text{if } e'_u \neq e_u \text{ and } a \in X, \\ 1 - p_{ec} - p_{ae} & \text{if } e_u = e'_u = e_1 \text{ and } a \notin X, \\ p_{ec} + p_{ae} & \text{if } e_u = e_1 \text{ and } e'_u = e_2 \text{ and } a \notin X, \\ p_{ec} - p_{ae} & \text{if } e_u = e_2 \text{ and } e'_u = e_1 \text{ and } a \notin X, \\ 1 - p_{ec} + p_{ae} & \text{if } e_u = e'_u = e_2 \text{ and } a \notin X, \end{cases} \tag{29}$$

where $p_{ec} \geq p_{ae} \geq 0$, $(p_{ec} + p_{ae}) \leq 1$ and $X = \{ask, confirm(g'_u), ok(g'_u)\}$. This model assumes that system mistakes (such as confirming the wrong item) would elevate the user's stress level.

The user's action model:

$$P(a'_u|a,g'_u,e'_u) = \begin{cases} 1 & \text{if } a = a_1,\, e'_u = e_1, \text{ and } a'_u = a_2, \\ 1 - p_1 & \text{if } a = a_1,\, e'_u = e_2, \text{ and } a'_u = a_2, \\ \frac{p_1}{|A_u|-1} & \text{if } a = a_1,\, e'_u = e_2, \text{ and } a'_u \neq a_2, \\ 1 & \text{if } a = a_3,\, e'_u = e_1, \text{ and } a'_u = a_4, \\ 1 - p_2 & \text{if } a = a_3,\, e'_u = e_2, \text{ and } a'_u = a_4, \\ \frac{p_2}{|A_u|-1} & \text{if } a = a_3,\, e'_u = e_2, \text{ and } a'_u \neq a_4, \\ 1 & \text{if } a = a_5,\, e'_u = e_1, \text{ and } a'_u = a_6, \\ 1 - p_2 & \text{if } a = a_5,\, e'_u = e_2, \text{ and } a'_u = a_6, \\ \frac{p_2}{|A_u|-1} & \text{if } a = a_5,\, e'_u = e_2, \text{ and } a'_u \neq a_6, \\ \frac{1}{|A_u|} & \text{if } a = ok(y) \text{ or } a = fail, \\ 0 & \text{otherwise,} \end{cases} \tag{30}$$

where $a_1 = ask$, $a_2 = answer(g'_u)$, $a_3 = confirm(g'_u)$, $a_4 = yes$, $a_5 = confirm(x)$, $a_6 = no$, $p_1 = p_e/K_{ask}$, $p_2 = p_e/K_{confirm}$, x & $y \in Gu$, and $x \neq g'_u$.

The main idea behind this handcrafted user's action model is explained as follows. When the user is not stressed, no communicative errors are made. When the user is under stress, they might make errors. The probability that the user makes no communicative errors when the system asks is $1 - p_1$ and when the system confirms is $1 - p_2$.

The user's grounding state model is handcrafted. It is represented by

$$P(d'_u|a,d_u,a'_u) = \begin{cases} 1 & \text{if } a = ask,\, d_u = d_1,\, a'_u = answer(x), \text{ and } d'_u = d_2, \\ 1 & \text{if } a = ask,\, d_u = d_1,\, a'_u \in \{yes,no\}, \text{ and } d'_u = d_1, \\ 1 & \text{if } a = confirm(x),\, d_u = d_1,\, a'_u = no, \text{ and } d'_u = d_1, \\ 1 & \text{if } a = confirm(x),\, d_u = d_1,\, a'_u \neq no, \text{ and } d'_u = d_2, \\ 1 & \text{if } a \in \{ask,confirm(x)\},\, d_u = d_2 \text{ and } d'_u = d_2, \\ 1 & \text{if } a \in \{ok(x),fail\} \text{ and } d'_u = d_1, \\ 0 & \text{otherwise.} \end{cases} \tag{31}$$

The observation model of the user's actions:

$$P(\tilde{a}_u|a_u) = \begin{cases} 1 - p_{oa} & \text{if } \tilde{a}'_u = a_u, \\ \frac{p_{oa}}{|A_u|-1} & \text{otherwise,} \end{cases} \tag{32}$$

where a_u and $\tilde{a}'_u$ are the user's action and the observation of the user's action, respectively. $|A_u|$ is the number of the user's actions. Parameter p_{oa} can be considered as the recognition error rate of the NLU module (see Section 2).

The observation model of the user's stress:

$$P(\tilde{e}_u|e_u) = \begin{cases} 1 - p_{oe} & \text{if } \tilde{e}_u = e_u, \\ \frac{p_{oe}}{|E_u|-1} & \text{otherwise,} \end{cases} \tag{33}$$

where e_u and $\tilde{e}_u$ are the user's stress state and the observation of the user's stress state, respectively. $|E_u|$ is the number of the user's stress states. Parameter p_{oe} can be considered as the recognition error rate of an affect recognition module used for the user's stress detection.

The reward function:

$$R(s,a) = \begin{cases} 0 & \text{if } s = \mathit{end}, \\ -2 & \text{if } a = \mathit{ask} \text{ and } g_u = \mathit{stated}, \\ -2 & \text{if } a = \mathit{confirm}(x) \text{ and } g_u = \mathit{notstated}, \\ 10 & \text{if } a = \mathit{ok}(x) \text{ and } g_u = x, \\ -10 & \text{if } a = \mathit{ok}(x) \text{ and } g_u \neq x, \\ -3 & \text{if } a = \mathit{fail}, \\ -1 & \text{otherwise.} \end{cases} \tag{34}$$

8 Evaluation

To compute a near-optimal policy for the route navigation example presented in Section 7, we use the Perseus solver[6] which is one of the state-of-the-art approximate POMDP solvers[7]. All experiments were conducted on a Linux server using a 3 GHz Intel Xeon CPU and a 24 GB RAM.

The performance of the computed policy is then evaluated using the simulated user presented in Section 6. The simulated user is implemented using the SMILE library[8]. The discount factor is only used for the planning phase. In the evaluation phase, the total reward for each dialogue session is the sum of all the rewards the system receives at each turn during the system–user interaction process (i.e., $\gamma = 1$). There are two reasons for this decision. First, in the dialogue domain, the number of turns in a dialogue session between the system and the user is finite. Second, the intuitive meaning of the discount factor ($\gamma < 1$) is reasonable for positive reward values but is not appropriate for negative reward values. For example, it is less likely that the second *confirm* action bears a smaller cost than the first one on a given piece of information provided by the user.

Note that when the discount factor γ is not used in the evaluation, it would be rational to set $\gamma = 1$ during the planning phase (our problem is guaranteed to terminate in a finite number of steps). However, Perseus requires that γ is smaller than 1. The planning time also increases when γ approaches 1. We argue that it does make sense to hand-tune the discount factor because it is not just a part of the problem description. As γ approaches 1, the agent becomes more farsighted [42].

[6] http://staff.science.uva.nl/ mtjspaan/pomdp/[accessed 2009-06-22]

[7] Although Perseus is made for flat POMDPs, this solver does exploit the factor representation of the state and observation space for computing near optimal policies. Alternatively, we can use the Symbolic Perseus solver. The advantage of this solver is mentioned in Section 8.4.

[8] http://genie.sis.pitt.edu[accessed 2009-06-24]

A dialogue session between the system and the user is defined as an *episode*. Each episode in the route navigation application starts with the system's action. Following the turn-taking mechanism, the system and the user exchange turns[9] until the system selects an *ok* or *fail* action (Table 3). The episode is then terminated[10].

The interaction between the DM and the simulated user is as follows. Both the dialogue manager (that uses the POMDP policy) and the simulated user exchange text messages through the iROS middleware[11]. Every time the dialogue manager performs *ok* or $fail$ action, the current episode is finished and a new episode is started. The interaction ends when it reaches the number of episodes predefined by the experimenter.

Formally, the reward of each episode is calculated from the user side as follows:

$$R_e = \sum_{t=0}^{n} R_t(S_t, A_t), \tag{35}$$

where n is the number of turns of the episode, the reward at turn t $R_t(S_t, A_t)$ is equal to R_s^a (see Section 3) if the user's state and action at turn t is $S_t = s$ and the system action at turn t is a. Note that each turn is composed of a pair of system and user's actions except the last turn. Table 3 shows an example of a 3-turns episode of the single-slot route navigation application.

Table 3 An episode of the interaction between the system and the user

Turn	Utterance	Action	Reward
1	S_1 : Please provide the location of the victims?	*ask*	-1
	U_1 : Building 1.	$answer(v_1)$	
2	S_2 : The victims are in the Building 1. Is that correct?	$confirm(v_1)$	-1
	U_2 : Yes.	*yes*	
3	S_3 : Ok, the route description is shown on your PDA.	$ok(v_1)$	10
		Reward of the episode:	8

The average return of N episodes is defined as follows:

$$R_N = \frac{1}{N} \sum_{e=1}^{N} R_e \tag{36}$$

In the following sections, we first present the experiments for tuning three important parameters: the discount factor, the number of belief points (i.e., belief states) and the planning time. Second, we show the influence of the stress to the performance of computed policies. Third, we compare the performance of the approximate POMDP policies versus three handcrafted policies and the greedy action selection policy. We then conduct experiments to address tractable issues.

[9] Each time step in a formal POMDP definition is now considered as a turn.

[10] A telephone-based dialogue example in [13] shows that the episode ends when the user hangs up.

[11] http://sourceforge.net/projects/iros/[accessed 2009-06-24]

8.1 Parameter Tuning

Figures 5, 6, and 7 show the average returns of the simplest instance of the route navigation problem (three locations) where the near-optimal policy is computed based on different values of the discount factor, the number of belief points, and the run-time for the planning phase.

Based on the result shown in Figure 5, the discount factor value $\gamma = 0.99$ is selected for subsequent experiments that will be presented in this chapter. Interestingly, it turns out that the *de facto* discount factor value ($\gamma = 0.95$) that is usually used for POMDP-based dialogue problems from the literature (e.g., Williams et al. [49]) is not an optimal solution, at least for this example. It is worth noting that the off-line planning time to get an ε-optimal policy increases monotonically with the value of the discount factor especially when the convergence threshold ε is small. For example, the time to get a similar policy for γ is equal to 0.9, 0.95, and 0.99 is 7, 17, and 98 seconds, respectively. However, the time-bounded solution (Fig. 5) performs well when we conduct the test with our simulated user.

Figure 6 shows that a reasonable solution is achieved with the number of belief points starting from 200. Given a fixed threshold of the planning time, the number of iterations decreases when the number of belief points increases. That is why the average return of the case of 10000 belief points is low when the planning

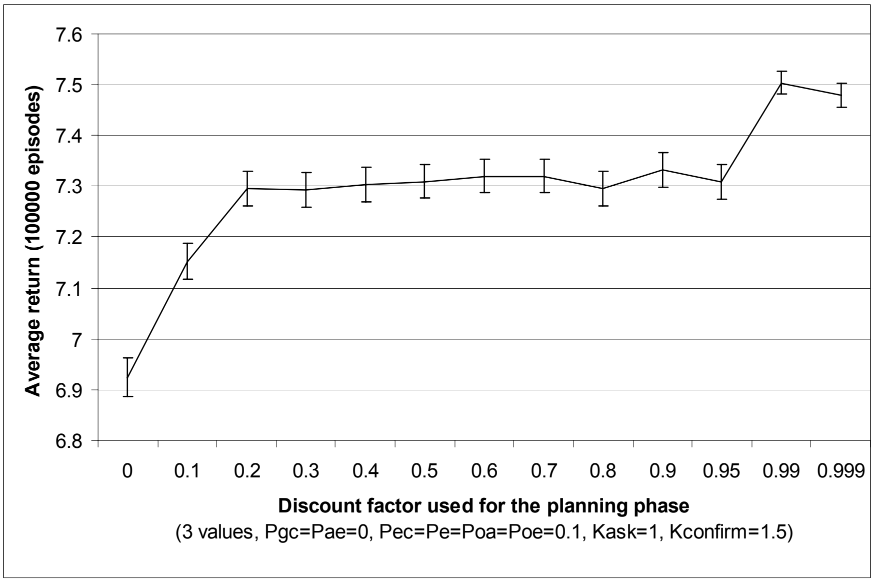

Fig. 5 Average return vs. the discount factor used for the planning phase. Error bars show the 95% confidence level. The threshold of the planning time is 60 seconds. Policies with $\gamma \leq 0.95$ converge ($\varepsilon = 0.001$) before this threshold.

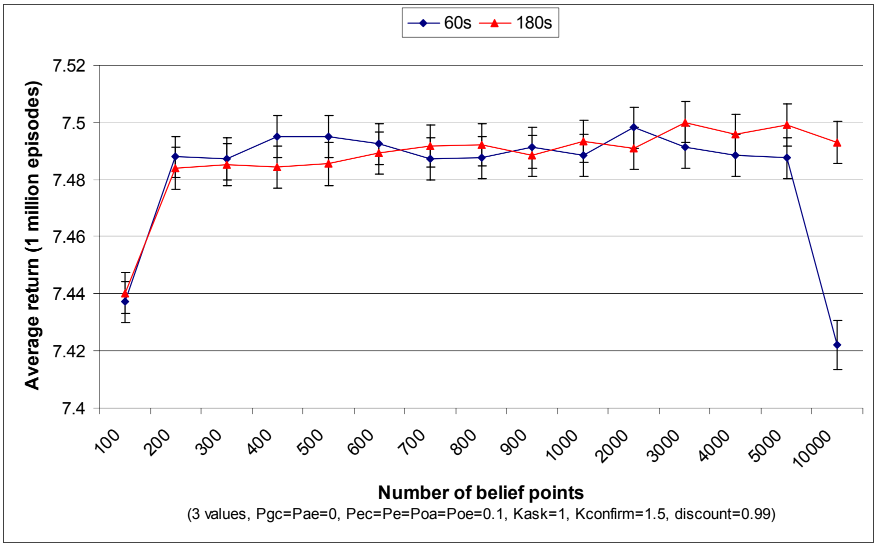

Fig. 6 Average return vs. number of belief points. Error bars show the 95% confidence level.

time is limited to 60 seconds. A default number of belief points for all subsequent experiments is set to 1000 (this value is a reasonable choice for the case that the number of locations is greater than 3).

Figure 7 shows that a stable solution is achieved when the planning time threshold is 60 seconds. For all the subsequent experiments for the 3-locations case, the default threshold for the planning time is set to 60 seconds. When the number of locations increases, the solver need a longer time to get a good solution. For example, the minimum time for the 10 locations case is 30 minutes.

8.2 Influence of Stress to the Performance

Figure 8 shows the influence of stress to the performance of two distinctive policies: the non-affective policy (SDS-POMDP) and the affective policy (ADS-POMDP). The SDS-POMDP does not incorporate the stress variable in the state and the observations set[12] (similar to the SDS-POMDP policy described in [49]). The ADS-POMDP is our affective dialogue model described in Section 5. The probability of the user's action error being induced by stress p_e changes from 0 (stress has no influence to the user's action selection) to 0.8 (the user is highly stressed and acts almost randomly). The average returns of both policies decreases when p_e increases. When stress has no influence on the user's action error, the average returns of the

[12] The SDS-POMDP policy is equivalent to the ADS-POMDP policy computed for the case $p_e = 0$.

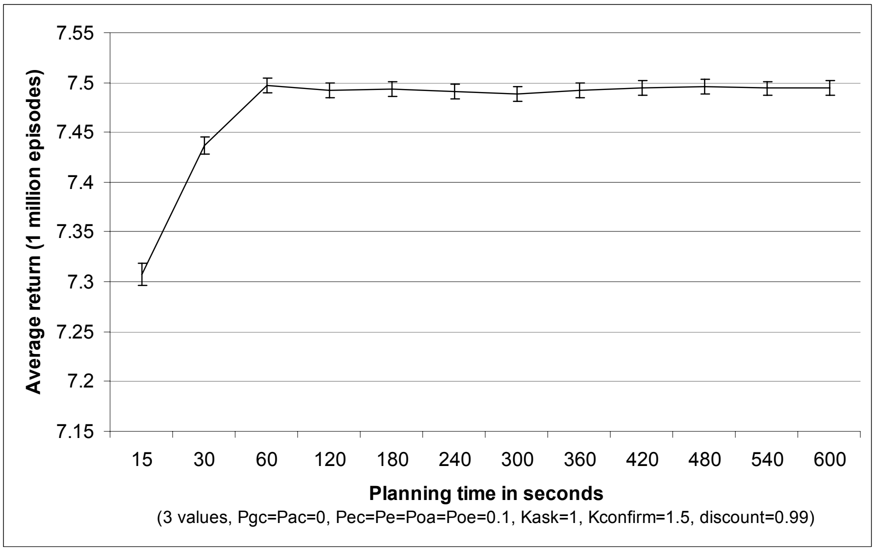

Fig. 7 Average return vs. planning time in seconds. Error bars show the 95% confidence level.

two policies are equal. When $p_e \geq 0.1$, the ADS-POMDP policy outperforms the SDS-POMDP counterpart[13].

Note that the affective POMDP policy performs better than its non-affective counterpart because it exploits the user's stress model. In reality, it is very challenging to obtain a reliable model of the user's stress. Therefore, it would be interesting to compare these two policies in an experiment with real users.

8.3 Comparison with Other Techniques

In this section, we evaluate the performance of the POMDP DM by comparing the performance of the approximate POMDP policy (ADS-POMDP) and four other dialogue strategies: HC1, HC2, HC3 (Fig. 9), and the greedy action selection strategy. HC1 is the optimal dialogue policy when $p_{gc} = p_e = p_{oa} = 0$ (the user's goal does not change; stress has no influence on the user's action and there is no error in observing the user's action, i.e., the speech recognition and spoken language understanding errors are equal to 0). HC1 and HC2 are considered as the non-affective dialogue strategies since they ignore the user's stress state. HC3 uses commonsense rules to generate the system behavior. The greedy policy is a special case of the POMDP-based dialogue with the discount factor $\gamma = 0$ (this strategy is similar to the one used in two real-world dialogue systems [29, 51]).

[13] We then assume that the ADS-POMDP policy is better than the non-affective MDP policy since Williams [46] demonstrated that the SDS-POMDP policy outperforms its MDP counterpart.

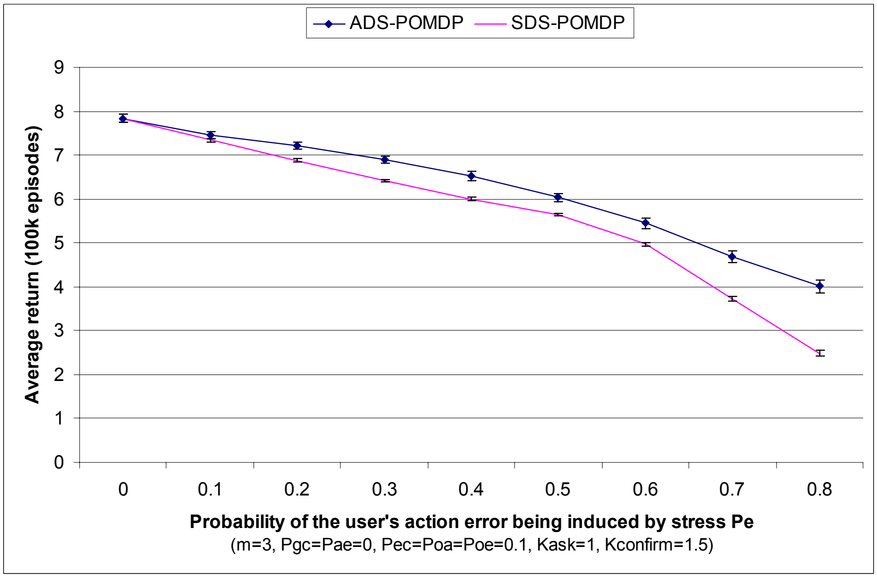

Fig. 8 Average returns of the affective policy and non-affective policy vs. the probability of the user's action error induced by stress p_e

As expected, the ADS-POMDP policy outperforms all other strategies (Fig. 10). HC3 outperforms its handcrafted counterparts.

8.4 Tractability

The ε-optimal policy presented in Sections 8.1, 8.2, and 8.3 is computed for the simplest instance of the single-slot route navigation example which is composed of only three locations ($m = 3$). In reality, even with a single-slot dialogue problem, the number of slot values m is usually large. Section 7 presents a connection between m and the size of the POMDP problem (state, action, and observation sets). For the single-slot route navigation example, the number of states is a quadric function of m. The number of actions and observations is also a linear function of m. In this section, we address the POMDP tractable issues by increasing the number of slot values gradually and trying to compute the ε-optimal policy for each case.

Perseus can only handle problems with $m \leq 15$. This is because although the approximate PBVI algorithms such as Perseus are able to handle the curse of history problem, the curse of dimensionality (i.e., the dimensionality of α-vectors grows exponentially with the number of states) remains (see [10, chap. 2] for further discussions about the POMDP tractability). Another practical issue is that the size of the optimized POMDP parameter file also increases exponentially in the dimension of m. A recently implemented POMDP solver, Symbolic Perseus, allows for a compact representation of the POMDP parameter files. Symbolic Perseus can help to

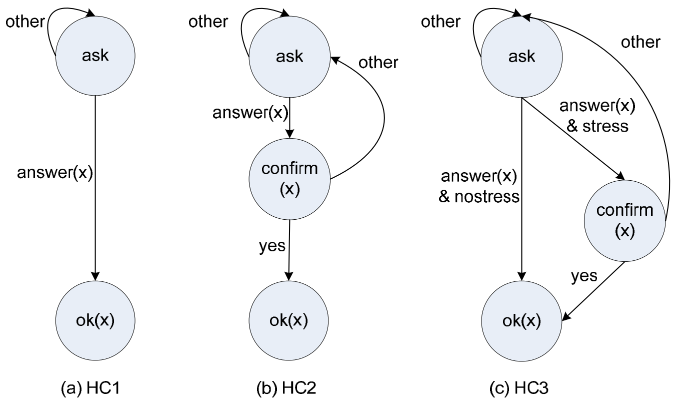

Fig. 9 Three handcrafted dialogue strategies for the single-slot route navigation problem (x is the observed location): (a) first *ask* and then select *ok* action if the observation of the user's action $\tilde{a}_u$ is *answer* (otherwise *ask*), (b) first *ask*, then *confirm* if $\tilde{a}_u = answer$ (otherwise *ask*) and then select *ok* action if $\tilde{a}_u = yes$ (otherwise *ask*), (c) first *ask*, then *confirm* if $\tilde{a}_u = answer$ & $\tilde{e}_u = stress$ and select *ok* action if $\tilde{a}_u = yes$

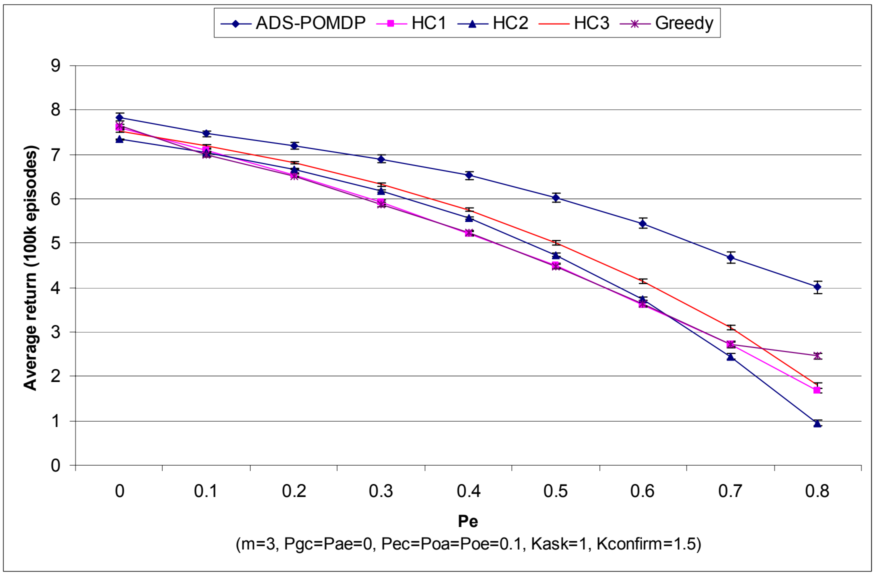

Fig. 10 Average return of the POMDP policy vs. other policies

scale the range of solvable problems to two orders of magnitude compared with the Perseus solver. As described in [18], it was demonstrated to solve a hand-washing problem with the size of 50 million states, 20 actions, and 12 observations. Although this is one of the most complex problems in the POMDP research community, it is only a toy problem compared with real-world dialogue problems. For example, the affective dialogue model for the RestInfo problem presented in [10, chap. 1] is composed of more than 600 million states and its spoken counterpart is composed of more than 300 million states. In [11], we proposed a solution to handle these complex problems by decomposing the dialogue model into two levels: global dialogue manger level and slot level dialogue manager level. The first is modeled using a set of simple rules. The second is first modeled as a factored POMDP similar to the one presented in this paper and then approximated as a set of Dynamic Decision Networks. More detailed information about this solution is presented in [11].

9 Conclusions

This chapter argues that POMDPs are appropriate for affective dialogue management. To support this argument, we have described the interaction process of a general affective dialogue system and illustrated how to specify the POMDP model for a simple empathic dialogue agent.

Extending from the previous POMDP-based dialogue management work, we have presented a factored POMDP approach for affective dialogue model design with a potential use for a wide range of applications. The 2TBN representation allows integration of the features of states, actions, and observations in a flexible way. The approach is illustrated through a route navigation example as a proof of concept. The experimental results showed that the affective POMDP policy outperformed its spoken counterpart and the handcrafted and greedy action selection strategies given the user's stress influences their behavior. We also conducted experiments to determine a best set of parameters (discount factor, number of belief points, and planning time). The results could be a good indicator for computing POMDP policy for other dialogue problems.

A key limitation of the current POMDP-based dialogue models is tractability. Given the mathematical soundness of the POMDP framework and recent efforts to resolve this limitation [11, 51], it would be worth to follow this direction for building robust dialogue systems.

References

1. Ai, H., Weng, F.: User simulation as testing for spoken dialog systems. In: Schlangen, D., Hockey, B.A. (eds.) Proceedings of the 9th SIGDial Workshop on Discourse and Dialogue (SIGdial 2008), Columbus, Ohio, USA, pp. 164–171 (2008)
2. André, E., Dybkjær, L., Minker, W., Heisterkamp, P. (eds.): ADS 2004. LNCS (LNAI), vol. 3068. Springer, Heidelberg (2004)
3. Astrom, K.J.: Optimal control of Markov processes with incomplete state information. Journal of Mathematical Analysis and Applications 10, 174–205 (1965)
4. Ball, E.: A Bayesian heart: Computer recognition and simulation of emotion. In: Robert Trappl, P.P., Payr, S. (eds.) Emotions in Humans and Artifacts, vol. 11, pp. 303–332. The MIT Press, Cambridge (2003)

5. Batliner, A., Fischer, K., Huber, R., Spilker, J., Nöth, E.: How to find trouble in communication. Speech Communication 40(1-2), 117–143 (2003)
6. Bhatt, K., Argamon, S., Evens, M.: Hedged responses and expressions of affect in human/human and human/computer tutorial interactions. In: Forbus, K., Gentner, D., Regier, T. (eds.) Proceedings of the 26th Annual Conference of the Cognitive Science Society (CogSci 2004), Chicago, Illinois, USA, pp. 114–119 (2004)
7. Boutilier, C., Poole, D.: Computing optimal policies for partially observable decision processes using compact representations. In: Proceedings of the 13th National Conference on Artificial Intelligence (AAAI 1996), Portland, Oregon, USA, vol. 2, pp. 1168–1175 (1996)
8. Brooks, A., Makarenkoa, A., Williamsa, S., Durrant-Whytea, H.: Parametric POMDPs for planning in continuous state spaces. Robotics and Autonomous Systems 54(11), 887–897 (2006)
9. Bui, T.H.: Multimodal dialogue management - State of the art. Tech. rep., University of Twente (2006)
10. Bui, T.H.: Toward affective dialogue management using partially observable markov decision processes. Ph.D. thesis, University of Twente (2008)
11. Bui, T.H., Poel, M., Nijholt, A., Zwiers, J.: A tractable hybrid DDN-POMDP approach to affective dialogue modeling for probabilistic frame-based dialogue systems. Natural Language Engineering 15(2), 273–307 (2009)
12. Bui, T.H., Rajman, M., Melichar, M.: Rapid dialogue prototyping methodology. In: Sojka, P., Kopeček, I., Pala, K. (eds.) TSD 2004. LNCS (LNAI), vol. 3206, pp. 579–586. Springer, Heidelberg (2004)
13. Bui, T.H., van Schooten, B., Hofs, D.: Practical dialogue manager development using POMDPs. In: Keizer, S., Bunt, H., Paek, T. (eds.) Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue (SIGdial 2007), Antwerp, Belgium, pp. 215–218 (2007)
14. Bui, T.H., Zwiers, J., Nijholt, A., Poel, M.: Generic dialogue modeling for multi-application dialogue systems. In: Renals, S., Bengio, S. (eds.) MLMI 2005. LNCS, vol. 3869, pp. 174–186. Springer, Heidelberg (2006)
15. Eckert, W., Levin, E., Pieraccini, R.: User modelling for spoken dialogue system evaluation. In: Prococeedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU 1997), pp. 80–87. IEEE, Santa Barbara (1997)
16. Hauskrecht, M.: Value-function approximations for partially observable Markov decision processes. Journal of Artificial Intelligence Research (JAIR) 13, 33–94 (2000)
17. Heylen, D., Nijholt, A., op den Akker, R.: Affect in tutoring dialogues. Applied Artificial Intelligence 19, 287–311 (2005)
18. Hoey, J., von Bertoldi, A., Poupart, P., Mihailidis, A.: Assisting persons with dementia during handwashing using a partially observable markov decision process. In: Proceedings of the 5th International Conference on Vision Systems (ICVS 2007), Bielefeld, Germany (2007)
19. Howard, R.A.: Dynamic Programming and Markov Process. The MIT Press, Cambridge (1960)
20. Jurafsky, D., Martin, J.: Speech and Language Processing: An Introduction to Natural Language Processing. Prentice Hall, Englewood Cliffs (2000)
21. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence 101(1-2), 99–134 (1998)
22. Levelt, W.J.: Speaking: From Intention to Articulation. The MIT Press, Cambridge (1989)

23. Levin, E., Pieraccini, R., Eckert, W.: A stochastic model of human-machine interaction for learning dialogue strategies. IEEE Transactions on Speech and Audio Processing 8(1), 11–23 (2000)
24. Littman, M.L., Cassandra, A.R., Kaelbling, L.P.: Learning policies for partially observable environments: Scaling up. In: Prieditis, A., Russell, S.J. (eds.) Proceedings of the 12th International Conference on Machine Learning (ICML 1995), pp. 362–370. Morgan Kaufmann, Tahoe City (1995)
25. Martinovsky, B., Traum, D.R.: The error is the clue: Breakdown in human-machine interaction. In: Proceedings of the ISCA Tutorial and Research Workshop on Error handling in Spoken Dialogue Systems (EHSD 2003), Château d'Oex, Vaud, Switzerland, pp. 11–16 (2003)
26. McTear, M.: Spoken dialogue technology: Enabling the conversational user interface. ACM Computing Survey 34(1) (2002)
27. Monahan, G.E.: A survey of partially observable markov decision processes: Theory, models, and algorithms. Management Science 28-1, 1–16 (1982)
28. Ortony, A., Clore, G.L., Collins, A.: The Cognitive Structure of Emotions. Cambridge University Press, Cambridge (1988)
29. Paek, T., Horvitz, E.: Conversation as action under uncertainty. In: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI 2000), pp. 455–464. Morgan Kaufmann, San Francisco (2000)
30. Picard, R.W.: Affective Computing. The MIT Press, Cambridge (1997)
31. Pietquin, O.: A framework for unsupervised learning of dialogue strategies. Ph.D. thesis, Universitaires de Louvain (2004)
32. Puterman, M.L.: Markov decision processes. In: Heyman, D., Sobel, M. (eds.) Handbook in Operations Research and Management Science, vol. 2, pp. 331–434. Elsevier, Amsterdam (1990)
33. de Rosis, F., Novielli, N., Carofiglio, V., Cavalluzzi, A., Carolis, B.D.: User modeling and adaptation in health promotion dialogs with an animated character. Journal of Biomedical Informatics 39(5), 514–531 (2006)
34. Roy, N., Pineau, J., Thrun, S.: Spoken dialogue management using probabilistic reasoning. In: Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL 2000), pp. 93–100. ACL, Hong Kong (2000)
35. Roy, N., Thrun, S.: Coastal navigation with mobile robots. In: Solla, S.A., Leen, T.K., Müller, K.R. (eds.) Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999], pp. 1043–1049. The MIT Press, Denver (2000)
36. Schatzmann, J., Thomson, B., Weilhammer, K., Ye, H., Young, S.: Agenda-based user simulation for bootstrapping a POMDP dialogue system. In: Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2007), pp. 149–152. ACL, Rochester (2007)
37. Schatzmann, J., Weilhammer, K., Stuttle, M., Young, S.: A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. Knowledge Engineering Review 21(2), 97–126 (2006)
38. Scheffler, K., Young, S.J.: Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In: Marcus, M. (ed.) Proceedings of the 2nd International Conference on Human Language Technology Research (HLT 2002), pp. 12–18. Morgan Kaufmann, San Francisco (2002)
39. Smallwood, R.D., Sondik, E.J.: The optimal control of partially observable Markov processes over a finite horizon. Operations Research 21-5, 1071–1088 (1973)

40. Sondik, E.J.: The optimal control of partially observable markov decision processes. Ph.D. thesis, Stanford University (1971)
41. Spaan, M.T.J., Vlassis, N.: Perseus: Randomized point-based value iteration for POMDPs. Journal of Artificial Intelligence Research (JAIR) 24, 195–220 (2005)
42. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, Cambridge (1998)
43. Traum, D., Larsson, S.: The information state approach to dialogue management. In: van Kuppevelt, J., Smith, R.W. (eds.) Current and New Directions in Discourse and Dialogue, ch. 15, pp. 325–353. Kluwer Academic Publishers, Dordrecht (2003)
44. Williams, J., Poupart, P., Young, S.: Partially Observable Markov Decision Processes with Continuous Observations for Dialog Management. In: Williams, J., Poupart, P., Young, S. (eds.) Recent Trends in Discourse and Dialogue, chap, pp. 191–217. Springer, Heidelberg (2008)
45. Williams, J., Young, S.: Scaling up POMDPs for dialogue management: the summary POMDP method. In: Proceedings of the IEEE workshop on Automatic Speech Recognition and Understanding (ASRU 2005), Cancún, Mexico, pp. 250–255 (2005)
46. Williams, J.D.: Partially observable Markov decision processes for dialog management. Ph.D. thesis, Cambridge University (2006)
47. Williams, J.D., Poupart, P., Young, S.: Factored partially observable Markov decision processes for dialogue management. In: Zukerman, I., Alexandersson, J., Jönsson, A. (eds.) Proceedings of the 4th Workshop on Knowledge and Reasoning in Practical Dialog Systems (KRPD 2005), Edinburgh, Scotland, pp. 76–82 (2005)
48. Williams, J.D., Poupart, P., Young, S.: Partially observable Markov decision processes with continuous observations for dialogue management. In: Proceedings of the 6th SigDial Workshop on Discourse and Dialogue, SIGdial 2005 (2005)
49. Williams, J.D., Young, S.: Partially observable markov decision processes for spoken dialog systems. Computer Speech and Language 21(2), 393–422 (2007)
50. Young, S.: Talking to machines (statistically speaking). In: Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP 2002), Denver, Colorado, USA, pp. 9–16 (2002)
51. Young, S., Gasić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., Yu, K.: The hidden information state model: A practical framework for pomdp-based spoken dialogue management. Computer Speech and Language (2009)
52. Zhang, B., Cai, Q., Mao, J., Guo, B.: Spoken dialog management as planning and acting under uncertainty. In: Proceedings of the 7th European Conference on Speech Communication and Technology (EUROSPEECH 2001), Aalborg, Denmark, pp. 2169–2172 (2001)
53. Zhang, N.L.: Efficient planning in stochastic domains through exploiting problem characteristics. Tech. Rep. HKUST-CS95-40, Hong Kong University of Science and Technology (1995)
54. Zhang, N.L., Zhang, W.: Speeding up the convergence of value iteration in partially observable markov decision processes. Journal of Artificial Intelligence Research (JAIR) 14, 29–51 (2001)