USING POMDPS FOR DIALOG MANAGEMENT

Steve Young

Engineering Department, Cambridge University, CB2 1PZ, UK

ABSTRACT

This paper explains how partially observable Markov decision processes (POMDPs) can provide a principled mathematical framework for modelling the inherent uncertainty in spoken dialog systems. It briefly summarises the basic mathematics and explains why exact optimisation is intractable. It then describes a form of approximation called the *Hidden Information State model* which does scale and which can be used to build practical systems.

Index Terms— statistical dialog modelling; partially observable Markov decision processes (POMDPs); hidden information state model

1. INTRODUCTION

The structure of a conventional dialog system is outlined in Fig 1(a). It contains three major components: speech understanding, speech generation and dialogue management. The speech understanding component maps user utterances into some abstract representation of the user's intended speech act a_u and the speech generation component performs the inverse operation mapping the machine's response a_m into speech. The dialog manager is fundamentally different to the other components in that rather than performing a mapping function, it is concerned with decision-making with delayed rewards. Thus, unlike the other components, conventional supervised learning is not directly applicable to dialog management.

The core of the dialog manager is a data structure which represents the system's view of the world in the form of a machine state s_m. This machine state typically encodes an estimate of three distinct sources of information: the user's input act $\tilde{a}_u$, an estimate of the intended user goal $\tilde{s}_u$[1]. and some record of the dialog history $\tilde{s}_d$[2] Most conventional dialog managers rely on hand-crafted deterministic rules for interpreting each (noisy) user dialog act $\tilde{a}_u$ and updating the state (see for example the *information state update* approach [1]). Based on each new state estimate, a dialog policy is used to select an appropriate response in the form of a system speech act a_m. This dialog cycle continues until either the user's goal is satisfied or the dialog fails.

The designers of such systems have to deal with a number of problems. Since the user's state s_u is unknown and the decoded inputs $\tilde{a}_u$ are prone to errors, there is a significant chance that $\tilde{s}_m$ will be incorrect. Hence, the dialog manager must include quite complex error recovery procedures. Recognition confidence scores can reduce the incidence of misunderstandings but these require thresholds to be set which are themselves notoriously difficult to optimise. Modern recognisers can produce alternative recognition hypotheses but it is not clear in practice how these can be used effectively. Finally, the impact of decisons taken by the dialog manager do not necessarily have an immediate effect, hence dialog optimisation requires forward planning and this is extremely difficult in a deterministic framework.

As has been argued previously, taking a statistical approach to spoken dialog system design provides the opportunity for solving many of the above problems in a flexible and principled way[2]. Early attempts at using a statistical approach modelled the dialog system as a Markov decision process (MDP)[3, 4, 5]. MDPs provide a good statistical framework since they allow forward planning and hence dialog policy optimisation through reinforcement learning [6]. However, MDPs assume that the entire state is observable. Hence, they cannot account for either the uncertainty in the user state $\tilde{s}_u$ and dialog history $\tilde{s}_d$, or the uncertainty in the decoded user's dialog act $\tilde{a}_u$.

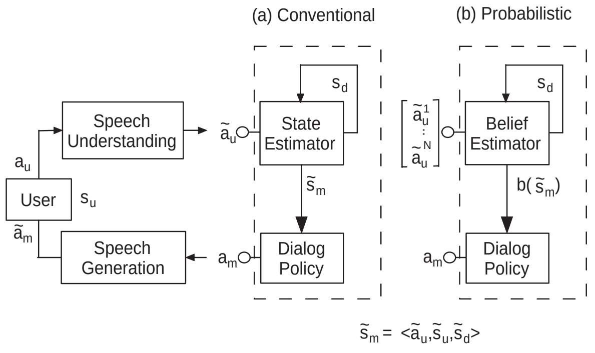

Fig. 1. Structure of a spoken dialog system. Part (a) shows a conventional dialog manager which maintains a single state estimate; (b) shows a dialog manager which maintains a distribution over all states and accepts an N-best list of alternative user inputs.

Fig 1(b) shows an alternative model for the dialog management component in which the uncertainty in the user's dialog act and the uncertainty in the machine state are shown explicitly. In this new model, the state estimator maintains a distribution across all states rather than a point-estimate of the most likely state. The dialog manager therefore tracks all possible dialog paths rather than just the most likely path. The ensuing dialog decision is then based on the distribution over all dialog states rather than just a specific state. This allows competing hypotheses to be considered in determining the machine's next move and simplifies error recovery since the dialog manager can simply shift its attention to an alternative hypothesis rather than trying to repair the existing one.

If the decoded user input act is regarded as an observation, then the dialog model shown in Fig 1(b) is a Partially Observable MDP (POMDP)[7]. The distribution over dialog states is called the *belief state* $\boldsymbol{b}$ and dialog policies are based on $\boldsymbol{b}$ rather than the true underlying state. The key advantage of the POMDP formalism is that

[1]Examples of user goals are "the need for flight information between London and New York", "the need to find a Chinese restaurant near the centre of town", "the desire to order three Pepperoni pizza's", etc.

[2]Since both a_u and s_u are noisy, the record of dialog history is also noisy, hence the tilde on s_d.

it provides a complete and principled framework for modelling the inherent uncertainty in a spoken dialog system. Thus, it naturally accommodates the implicit uncertainty in the estimate of the user's goal and the explicit uncertainty in the N-best list of decoded user acts. Associated with each dialog state is a reward. The choice of reward function is a dialog design issue, but it will typically provide positive rewards for satisfying the user's goal, and negative rewards for failure and wasting time. As with regular MDPs, dialogue optimisation is equivalent to finding a decision policy which maximises the total reward.

The use of POMDPs for any practical system is, however, far from straightforward. Firstly, in common with MDPs, the state space of a practical SDS is very large and if represented directly, it would be intractable. Secondly, since a belief distribution $\boldsymbol{b}$ over a discrete state s of cardinality $n+1$ lies in a real-valued n-dimensional simplex, a POMDP is equivalent to an MDP with a continuous state space $\boldsymbol{b} \in \Re^n$. Thus, a POMDP policy is a mapping from partitions in n-dimensional belief space to actions. Not surprisingly these are extremely difficult to construct and whilst exact solution algorithms do exist, they do not scale to problems with more than a few states/actions.

This paper gives an overview of POMDPs for use in dialog management and then describes a form of POMDP which can support very large hierarchical state spaces. This new model is inspired by the information state approach to dialog system implementation, and hence it is called the Hidden Information State (HIS) framework for statistical dialog systems. The HIS model provides a compact method of state representation based on partitioning, and it allows efficient policy optimisation by mapping between the full state space and a much reduced summary space.

2. POMDPS FOR DIALOGUE MANAGEMENT

2.1. POMDP Basics

Formally, a Partially Observable MDP is defined as a tuple $\{S_m, A_m, T, R, O, Z, \lambda, \boldsymbol{b}_0\}$ where S_m is a set of machine states; A_m is a set of actions that the machine may take; T defines a transition probability $P(s'_m|s_m, a_m)$; R defines the expected (immediate, real-valued) reward $r(s_m, a_m)$; O is a set of observations; Z defines an observation probability $P(o'|s'_m, a_m)$; λ is a geometric discount factor $0 \leq \lambda \leq 1$; and $\boldsymbol{b}_0$ is an initial belief state.

A POMDP operates as follows. At each time-step, the machine is in some unobserved state $s_m \in S_m$. Since s_m is not known exactly, a distribution over states is maintained called a belief state such that the probability of being in state s_m given belief state $\boldsymbol{b}$ is $b(s_m)$ [3]. Based on the current belief state $\boldsymbol{b}$, the machine selects an action $a_m \in A_m$, receives a reward $r(s_m, a_m)$, and transitions to a new (unobserved) state s'_m, where s'_m depends only on s_m and a_m. The machine then receives an observation $o' \in O$ which is dependent on s'_m and a_m. Finally, the belief distribution $\boldsymbol{b}$ is updated based on o' and a_m as follows:

$$\begin{aligned} b'(s'_m) &= P(s'_m|o', a_m, \boldsymbol{b}) \\ &= \frac{P(o'|s'_m, a_m, \boldsymbol{b})P(s'_m|a_m, \boldsymbol{b})}{P(o'|a_m, \boldsymbol{b})} \\ &= \frac{P(o'|s'_m, a_m)\sum_{s_m \in S_m} P(s'_m|a_m, \boldsymbol{b}, s_m)P(s_m|a_m, \boldsymbol{b})}{P(o'|a_m, \boldsymbol{b})} \\ &= k \cdot P(o'|s'_m, a_m) \sum_{s_m \in S_m} P(s'_m|a_m, s_m) b(s_m) \end{aligned} \quad (1)$$

[3]In other words, a belief state is a vector whose component values give the probabilities of being in each machine state.

where $k = P(o'|a_m, \boldsymbol{b})$ is a normalisation constant[7]. Maintaining this belief state as the dialog evolves is called *belief monitoring*.

At each time step t, the machine receives a reward $r(\boldsymbol{b}_t, a_{m,t})$ based on the current belief state $\boldsymbol{b}_t$ and the selected action $a_{m,t}$. The cumulative, infinite horizon, discounted reward is called the *return* and it is given by:

$$R = \sum_{t=0}^{\infty} \lambda^t r(\boldsymbol{b}_t, a_{m,t}) = \sum_{t=0}^{\infty} \lambda^t \sum_{s_m \in S_m} b_t(s_m) r(s_m, a_{m,t}). \quad (2)$$

Each action $a_{m,t}$ is determined by a policy $\pi(\boldsymbol{b}_t)$ and building a POMDP system involves finding the policy π^* which maximises the return. Unlike the case of MDPs, the policy is a function of a continuous multi-dimensional variable and hence its representation is not straightforward. However, it can be shown that the optimal policy is always piecewise linear and convex in belief space[8]. Hence, it can be represented by a set of *policy vectors* where each vector $\boldsymbol{v}_i$ is associated with an action $a(i) \in A_m$ and $v_i(s)$ equals the expected value of taking action $a(i)$ in state s. Given a complete set of policy vectors, the optimal value function and corresponding policy is

$$V^{\pi^*}(\boldsymbol{b}) = \max_i \{\boldsymbol{v}_i . \boldsymbol{b}\} \quad (3)$$

and

$$\pi^*(\boldsymbol{b}) = a(\operatorname*{argmax}_i \{\boldsymbol{v}_i . \boldsymbol{b}\}). \quad (4)$$

This representation is illustrated in Fig 2(a) for the case of $|S_m| = 2$ and a value function requiring just 3 distinct linear segments. The value function itself is the upper heavy line. The linear segments divide belief space into 3 regions and the optimal action to take in each region is the action associated with the uppermost vector in that region. So for example, if $b < x$ in Fig. 2, then action $a(1)$ would be chosen, if $x < b < y$ then action $a(2)$ would be chosen, and so on.

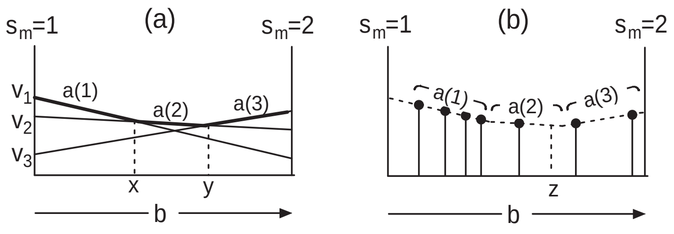

Fig. 2. POMDP Value function representation: (a) shows exact representation; (b) shows grid-based representation.

The optimal exact value function can be found by working backwards from the terminal state in a process called *value iteration*. At each iteration t, policy vectors are generated for all possible action/observation pairs and their corresponding values are computed in terms of the policy vectors at step $t-1$. As t increases, the estimated value function converges to the optimal value function from which the optimal policy can be derived. Many spurious policy vectors are generated during this process, and these can be pruned to limit the combinatorial explosion in the total number of vectors[7, 9]. Unfortunately, this pruning is itself computationally expensive and in practice, exact optimisation is not tractable. However, approximate solutions can still provide useful policies. The simplest approach is to discretise belief space and then use standard MDP optimisation methods [6]. Since belief space is potentially very large, grid points are concentrated on those regions which are likely to be visited[10, 11]. This is illustrated in Fig 2(b). Each belief point represents the value function at that point and it will have associated with it the corresponding optimal action to take. When an action is required for an arbitrary belief point $\boldsymbol{b}$, the nearest belief point is found and its action is used. This can lead to errors and hence the distribution of grid points in belief space is very important. For example, in Fig. 2(b), if $b = z$ then $a(3)$ would be selected although from Fig. 2(a) it can be seen that the optimal action was actually $a(2)$.

Grid-based methods are often criticised because they do not scale well to large state spaces[12]. The HIS model described below avoids the scaling problem by mapping the full belief space into a much reduced summary space where grid-based approximations appear to work reasonably well. A popular approach which is intermediate between exact and grid-based methods is to use *Point-based Value Iteration (PBVI)*[12, 13]. This is a hybrid method in which only a small set of belief points are computed, but a full policy vector is computed at each point. This allows value functions to be interpolated between grid points allowing it to deal with larger state spaces.

2.2. The SDS POMDP

As discussed in the introduction, when using a POMDP to model a spoken dialog system, it is natural to factor the machine state into three components $s_m = (s_u, a_u, s_d)$ [14][4] The belief state b is then a distribution over these three components. The observation o is the estimate of the user dialog act $\tilde{a}_u$. In the general case this will be an N-best list of hypothesised user acts, each with an associated probability, i.e.

$$o \quad = \quad [(\tilde{a}_u^1, p_1), (\tilde{a}_u^2, p_2), \ldots, (\tilde{a}_u^N, p_N)] \tag{5}$$

such that $p_n = P(\tilde{a}_u^n|o)$ for $n = 1 \ldots N$.

The transition function for an SDS-POMDP follows directly by substituting the factored state into the regular POMDP transition function and making some reasonable independence assumptions, i.e.

$$\begin{aligned} P(s'_m|s_m, a_m) &= P(s'_u, a'_u, s'_d|s_u, a_u, s_d, a_m) \\ &= P(s'_u|s_u, a_m)P(a'_u|s'_u, a_m)P(s'_d|s'_u, a'_u, s_d, a_m) \end{aligned} \tag{6}$$

This is the *transition model*. Making similar reasonable independence assumptions regarding the observation function gives,

$$P(o'|s'_m, a_m) = P(o'|s'_u, a'_u, s'_d, a_m) = P(o'|a'_u) \tag{7}$$

This is the *observation model*.

The above factoring simplifies the belief update equation since substituting (6) and (7) into (1) gives

$$\begin{aligned} & b'(s'_u, a'_u, s'_d) = \\ & k \cdot \underbrace{P(o'|a'_u)}_{\substack{\text{observation}\\ \text{model}}} \underbrace{P(a'_u|s'_u, a_m)}_{\substack{\text{user action}\\ \text{model}}} \sum_{s_u} \underbrace{P(s'_u|s_u, a_m)}_{\substack{\text{user goal}\\ \text{model}}} \\ & \cdot \sum_{s_d} \underbrace{P(s'_d|s'_u, a'_u, s_d, a_m)}_{\text{dialog model}} \quad b(s_u, s_d) \end{aligned} \tag{8}$$

As shown by the labelling in (8), the probability distribution for a'_u is called the *user action model*. It allows the observation probability that is conditioned on a'_u to be scaled by the probability that the user would speak a'_u given the goal s'_u and the last system prompt a_m. This enables a posterior ranking of N-best recognition output to be made based on the dialog system's current beliefs. The *user goal model* determines the probability of the user goal switching from s_u to s'_u following the system prompt a_m. Finally, the *dialog model* enables information relating to the dialog history to be maintained such as grounding and focus. Fig 3 shows the SDS-POMDP in the form of a dynamic Bayesian network influence diagram.

The detailed form of the statistical models needed in (8) will depend on the detailed structure of the dialogue manager. The observation model will typically be derived directly from the N-best list output by the recogniser and the associated posterior probability of each hypothesis in the list. The user action model depends on the representation of dialog acts and the user goal. However, it will usually be some form of discrete distribution with parameters estimated from dialog corpora. The user goal model is often of limited concern since for many dialog applications, it can be assumed that the user will change his/her goal relatively infrequently. Finally, the dialog model is usually a deterministic encoding based on some form of grounding model (eg. [17]). It yields probability one when the updated dialog hypothesis (ie a specific combination of s_u, a_u and s_d) is consistent with the history and zero otherwise.[5] Actual examples of these probabilistic models will be provided by the HIS system described below.

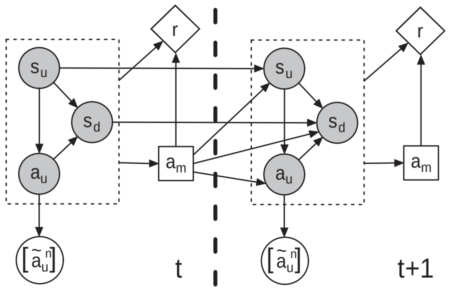

Fig. 3. SDS-POMDP dialog framework as a Bayesian Network

2.3. Utilising a Summary Space

In most practical spoken dialog systems, the state space and the set of possible user dialog acts are very large. As a result, belief monitoring as in (8) is computationally very expensive and direct policy optimisation is intractable. In the description of the HIS model below, a method of state partitioning will be described which allows efficient belief monitoring for practical dialog systems. However, something rather more aggressive is needed for policy optimisation.

As explained previously, modelling a dialog system as a POMDP involves maintaining a probability distribution over all possible machine states. Although very unlikely states can be pruned away, when performing belief monitoring it is important to maintain a large spread of states so that the system can recover from misunderstandings caused by recognition errors. When performing policy optimisation, however, the subsequent machine action is likely to focus on just the most likely states. This suggests maintaining two coupled state spaces: the full space called the *master state space* and a much simpler space called the *summary state space*[18, 19]. The summary state space consists of the top N user goal states (s_u) from master space (where N is typically 1 or 2) and a simplified encoding of the user action a_u and dialog history s_d. The summary

[4]Note that alternative POMDP formulations can also be used for SDS eg [15, 16].

[5]The dialog model provides a convenient place to incorporate heuristics.

action space consists of a list of high level abstractions of possible machine responses. A dialog turn then consists of first updating the belief state by evaluating (1) in master space. The updated belief state $\boldsymbol{b}$ is then mapped into a summary state $\hat{\boldsymbol{b}}$ where an optimised dialog policy is applied to compute a new summary machine action $\hat{a}_m$. The summary machine action is then mapped back into master space where it is converted to a specific machine dialog act a_m and a response is output to the user.

Policy optimisation is also carried out using the coupled spaces. As with conventional MDPs, there are two main approaches: model-based and on-line. Model-based optimisation relies entirely on the system dynamics defined by the models in (7) and (8). Hence, the system dynamics of the summary POMDP must be first estimated from the master POMDP using a random policy and sampling. Once the summary space distributions are known, any grid-based optimisation scheme can be used. In PBVI, for example, a representative set of belief points is generated by simulating dialogs. Starting from some initial belief $\boldsymbol{b}_0$, PBVI stochastically simulates a single turn by generating each possible machine action and then sampling a corresponding state and observation from the probability models. Each newly reached belief point is compared to the existing belief points and if it is sufficiently different it is retained. This is repeated until the desired set of belief points is obtained. Policy optimisation then utilises a standard dp recursion to iteratively refine the single policy vector associated with each belief point (see [12] for details). The use of PBVI and other grid-based approaches to summary space optimisation is described in more detail in [18, 19].

On-line approaches use *Q-learning* which is conceptually much simpler. Starting from some initial policy, the system engages with a user as in normal operation. As with PBVI described above, each new belief point encountered in summary space is compared with all existing belief points and if it is sufficiently different it is retained. Associated with each retained summary space belief point $\hat{\boldsymbol{b}}_i$ is a function $Q(\hat{\boldsymbol{b}}_i, \hat{a}_m)$ whose value is the expected total reward obtained by choosing summary action $\hat{a}_m$ from state $\hat{\boldsymbol{b}}_i$. This function is estimated either by batch learning or sequentially using exactly the same methods as for MDPs (see [6, Ch6/7]). Once the Q values have been estimated, the policy is found trivially by choosing the action which maximises each Q value. Since even the summary state space is very large, several thousand dialogs are required for on-line learning and this is rarely practicable with real users. Hence, user simulation techniques are typically used instead [20, 21].

3. THE HIDDEN INFORMATION STATE MODEL

The remainder of this paper briefly describes the Hidden Information State (HIS) model as a specific example of a POMDP-based dialog framework which does scale to handle real-world applications.

3.1. State Partitioning

The key idea underlying the HIS model is that at any point in a dialog, most of the possible user goal states have identical beliefs simply because there has been no evidence offered by the user to distinguish them. Significant computation can therefore be saved by grouping these states into equivalence classes. The HIS model therefore assumes that at any time t, the space of all user goals S_u can be divided into a number of equivalence classes where the members of each class are tied together and are indistinguishable. These equivalence classes are called *partitions*. Initially, all states $s_u \in S_u$ are in a single partition p_0. As the dialog progresses, this root partition is repeatedly split into smaller partitions. This splitting is binary i.e. $p \rightarrow \{p', p - p'\}$ with probability $P(p'|p)$. Since multiple splits can occur at each time step, this binary split assumption places no restriction on the possible refinement of partitions from one turn to the next. Given that user goal space is partitioned in this way, beliefs can be computed based on partitions of S_u rather than on the individual states of S_u. Initially the belief state is just $b_0(p_0) = 1$. Whenever a partition p is split, its belief mass is reallocated as,

$$b(p') = P(p'|p)b(p) \text{ and } b(p - p') = (1 - P(p'|p))b(p) \quad (9)$$

Note that this splitting of belief mass is simply a reallocation of existing mass, it is not a belief update, rather it is *belief refinement*.

entity	$\rightarrow$	venue(name,type,area)	1.0
type	$\rightarrow$	bar(drinks,music)	0.4
type	$\rightarrow$	restaurant(food,pricerange)	0.3
area	=	(central|east|west| . . .)	
food	=	(Italian|Chinese| . . .)	

Table 1. Example Ontology Rules for fictitious Tourist Information domain.

The space of all user goals is described by a set of simple ontological rules of the form illustrated in Table 1 taken from a fictitious Tourist Information domain. These rules describe the hierarchical structure of the data and the specific values which can be assigned to terminal nodes.[6] Since non-terminal nodes can be expanded in different ways, node expansion rules (indicated by a $\rightarrow$) have an associated prior probability corresponding to the partition split probability $p(p'|p)$ described above.

Partitions of user goal space are represented by a forest of trees where each tree represents a single partition. This forest of trees is stored in such a way that no partition is duplicated and the sum of the probability of all partitions is always unity. At the start of a dialog, there is just one partition represented by a single root node with belief mass unity. Each incoming user act is matched against each partition in turn. If there is no match, the ontology rules are consulted and the system attempts to create a match by expanding the tree. This expansion will result in partitions being split and their belief mass redistributed. This is illustrated in Fig. 4 in which a partition representing a generic "venue" is split as the result of the user requesting a "bar". The original "type" node had a probability mass of 1.0 and this is redistributed according to the prior in the corresponding ontology rule, 0.4 to the new partition and 0.6 remains with the original. If the user subsequently mentioned a "restaurant", this remaining mass of 0.6 would be split again.

3.2. Belief Monitoring

Belief monitoring in the HIS model is based on the standard SDS-POMDP update formula given in (8). However, a further simplification can be obtained by defining the duration of a dialog as spanning the interaction needed to satisfy a single goal. This results in the user goal model simplifying trivially to a delta function

$$P(s'_u|s_u) = \delta(s'_u, s_u). \quad (10)$$

[6] It should be noted that apart from the database itself, there is no other application dependent data or code in a HIS dialog manager.

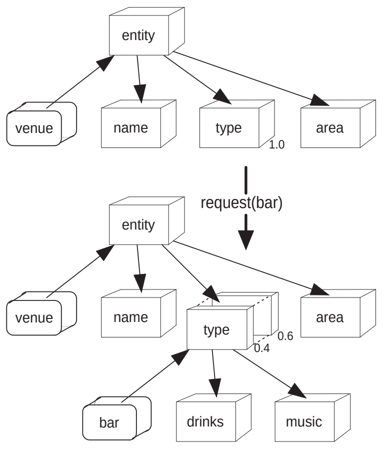

Fig. 4. Illustration of Partition Splitting

Substituting (9) and (10) into (8) gives the belief update equation for the HIS model

$$b'(p', a'_u, s'_d) = \\ k \cdot \underbrace{P(o'|a'_u)}_{\text{observation model}} \underbrace{P(a'_u|p', a_m)}_{\text{user action model}} \\ \cdot \sum_{s_d} \underbrace{P(s'_d|p', a'_u, s_d, a_m)}_{\text{dialog model}} \underbrace{P(p'|p) b(p, s_d)}_{\text{belief refinement}} \quad (11)$$

where p is the parent of p'. Thus, there is now just one summation required for each distinct belief point. Since many dialog hypotheses have very low probability and can therefore be pruned, it is simplest to ignore this summation and just compute all possible dialog hypotheses associated with each p and then merge those with a common s_d.

3.3. The HIS Model Dialogue Cycle

The overall operation of the prototype HIS system is summarised in Fig. 5. Each user utterance is decoded into an N-best list of dialog acts. Each incoming act plus the previous system act are matched against the forest of user goals and partitions are split as needed. Each user act a_u is then duplicated and bound to each partition p. Each partition will also have a set of dialog histories s_d associated with it. The combination of each p, a_u and updated s_d forms a new dialog hypothesis h_k whose beliefs are evaluated using (11).

Once all dialog hypotheses have been evaluated and any duplicates merged, the master belief state $\boldsymbol{b}$ is mapped into summary space $\hat{\boldsymbol{b}}$ and the nearest policy belief point is found. The associated summary space machine action $\hat{a}_m$ is then mapped back to master space and the machine's actual response a_m is output. The cycle then repeats until the user's goal is satisfied.

3.4. Component Models

The current HIS system computes the required probabilities as follows. Firstly, since the observation is an N-best list of user acts, the observation probability $P(o|a_u)$ can be approximated by p_i where $a_u = \tilde{a}_u^i$ (see (5)) i.e. the posterior probability of the N-best list element corresponding to a_u.

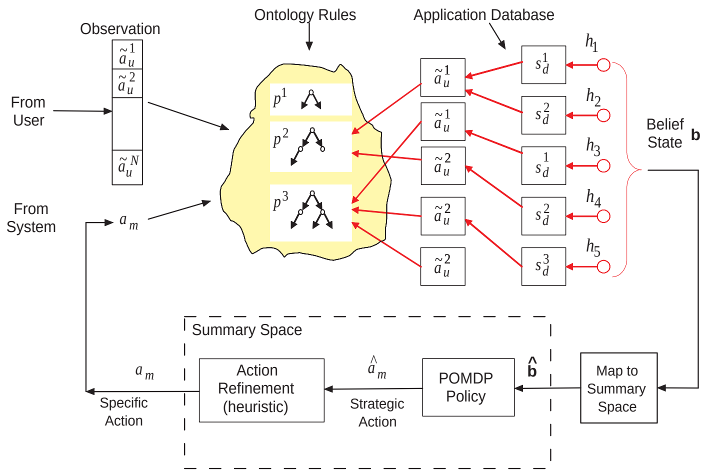

Fig. 5. Overview of Prototype HIS Dialogue Manager

User dialog acts take the form $act(a = v)$ where act is the dialog type, a is an attribute and v is its value [for example, *request(food=chinese)*], the user action model is then approximated by

$$P(a'_u|p', a_m) \approx P(\mathcal{T}(a'_u)|\mathcal{T}(a_m))P(\mathcal{M}(a'_u)|p') \qquad (12)$$

where $\mathcal{T}(\cdot)$ denotes the *type* of the dialog act and $\mathcal{M}(\cdot)$ denotes whether or not the dialog act *matches* the current partition p'. The first term on the RHS of (12) is estimated from a dialog corpus, the second term is set to 1 if the act matches and zero otherwise. As noted earlier, the dialog model $P(s'_d|p', a'_u, s_d, a_m)$ is set to 1 if s'_d is consistent with p', a'_u, s_d, a_m and 0 otherwise.

3.5. Summary Space Mapping and Policy Optimisation

In the current HIS system, a belief point in summary state space is a hybrid of continuous probability values and discrete variables. It consists of the probability of the 1st and 2nd best dialog hypotheses; the type of the user act a_u associated with the most likely state; a summary status of the top partition (initial, partly expanded, unique state, etc) and a summary of the dialog history of the top hypothesis (initial, partly grounded, fully grounded, etc). The distance metric to compare two dialog points strongly encourages discrete components to match exactly by returning a large distance if any discrete component differs, and a smaller Euclidean distance between the continuous components. The summary action set consists of strategic actions such as *greet*, *request*, *implicit_confirm*, *explicit_confirm*, *submit*, *inform*, *hangup*, etc. These are mapped back to master space by assuming they operate on the most likely hypothesis and the detailed dialog history maintained in master space enables the selection of appropriate attributes and values. For example, if the summary action is *implicit_confirm*, the system uses the dialog history to identify an unconfirmed node and a required node from the top hypothesis and then combines these to generate a response such as *impconfirm(venue=restaurant,food=?)* (i.e. "You want a restaurant. What kind of food would you like?").

Policy optimisation currently utilises on-line batch Q-learning using an external user simulator and an ϵ-greedy policy (i.e. the dialog manager follows its current best policy except that with probability ϵ it substitutes a random action). Learning follows classical Monte Carlo policy iteration. The current policy is held constant for 5000 dialogues whilst the Q-values associated with each summary space belief point are updated, the accumulated Q values are then used to generate a new policy, and the cycle repeats until the average reward stabilises.

4. CONCLUSIONS

This paper has argued that partially observable Markov decision processes (POMDPs) provide a principled mathematical framework for modelling the inherent uncertainty in spoken dialog systems. The three key ideas underlying this use of POMDPs are firstly that by maintaining a distribution over all possible machine states, it is possible to accumulate multiple tracks of possibly conflicting evidence. Secondly, by treating the recognition output as an observation from which the dialog state can be inferred, it is possible to weight the understanding component's confidence in each recognised user act with the probability of that act given the current dialog state. Thirdly, by casting the system as a Markov process it is possible to define and optimise a control policy to automatically maximise a desired reward function. Thus, the use of POMDPs for dialog offers, in principle at least, more robust dialogs, reduced need for manual tuning, application independence and the ability to adapt on-line.

The direct implementation of a POMDP leads to a system which is computationally intractable. However, this can be overcome by applying two simple refinements. Firstly, states can be grouped into partitions and beliefs can be based on these partitions. As long as partitions can be split on demand during the dialog, this causes no loss of accuracy but greatly reduces the computational requirements for belief monitoring. Secondly, the complex state/action space of a practical real world dialog system can be mapped into a much simpler summary space which focuses on just the top 1 or 2 machine state hypotheses. Since the majority of machine responses would refer to these hypotheses anyway, policy optimisation in this reduced summary space can be just as effective as optimisation in the master space whilst avoiding the complexity problems inherent in the latter.

Finally, a practical system based on these principles has been briefly described called the Hidden Information State system. The HIS system has been implemented for a tourist information domain and apart from a set of ontology rules it has no application specific code, and it was trained entirely from data. Having established a viable framework, the challenge now is to demonstrate that the potential of POMDPs can be translated into competitive spoken dialog systems.

5. ACKNOWLEDGEMENTS

Many of the ideas presented in this paper arose from discussions with colleagues and students in the Machine Intelligence Laboratory at the Cambridge University Engineering Department. In particular, Jason Williams contributed significantly to the development of POMDPs and Jost Schatzmann to the development of User Models and User Simulation. Some of the work described in this paper was supported by the EU Framework 6 TALK Project.

6. REFERENCES

[1] S Larsson and D Traum, "Information State and Dialogue Management in the TRINDI Dialogue Move Engine Toolkit," *Natural Language Engineering*, pp. 323–340, 2000.

[2] SJ Young, "Talking to Machines (Statistically Speaking)," in *Int Conf Spoken Language Processing*, Denver, Colorado, 2002.

[3] E Levin, R Pieraccini, and W Eckert, "Using Markov Decision Processes For Learning Dialogue Strategies," in *Proc Int Conf Acoustics, Speech and Signal Processing*, Seattle,USA, 1998.

[4] E Levin, R Pieraccini, and W Eckert, "A Stochastic Model of Human-Machine Interaction for Learning Dialog Strategies," *IEEE Trans Speech and Audio Processing*, vol. 8, no. 1, pp. 11–23, 2000.

[5] SJ Young, "Probabilistic Methods in Spoken Dialogue Systems," *Philosophical Trans Royal Society (Series A)*, vol. 358, no. 1769, pp. 1389–1402, 2000.

[6] RS Sutton and AG Barto, *Reinforcement Learning: An Introduction*, Adaptive Computation and Machine Learning. MIT Press, Cambridge, Mass, 1998.

[7] LP Kaelbling, ML Littman, and AR Cassandra, "Planning and Acting in Partially Observable Stochastic Domains," *Artificial Intelligence*, vol. 101, pp. 99–134, 1998.

[8] EJ Sondik, *The Optimal Control of Partially Observable Markov Decision Processes*, Phd, Stanford University, 1971.

[9] ML Littman, "The Witness Algorithm: solving partially observable Markov decision processes," Tech. Rep., Brown University, December 1994 1994.

[10] RI Brafman, "A Heuristic Variable Grid Solution Method for POMDPs," in *AAAI*, Cambridge, MA, 1997.

[11] B Bonet, "An e-Optimal Grid-based Algorithm for Partially Observable Markov Decision Processes," in *Proceedings of the Nineteenth International Conference on Machine Learning(ICML 2002),*, Sydney, Australia, 2002.

[12] J Pineau, G Gordon, and S Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," in *Proc Int Joint Conference on AI (IJCAI)*, Acapulco, Mexico, 2003, pp. pp1025–1032.

[13] MTJ Spaan and N Vlassis, "Perseus: randomized point-based value iteration for POMDPs," Tech. Rep., Universiteit van Amsterdam, 2004.

[14] JD Williams, P Poupart, and SJ Young, "Factored Partially Observable Markov Decision Processes for Dialogue Management," in *4th Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Edinburgh, 2005.

[15] N Roy, J Pineau, and S Thrun, "Spoken Dialogue Management Using Probabilistic Reasoning," in *Proceedings of the ACL 2000*, 2000.

[16] B Zhang, Q Cai, J Mao, and B Guo, "Planning and Acting under Uncertainty: A New Model for Spoken Dialogue System," in *Proc 17th Conf on Uncertainty in AI*, Seattle, 2001.

[17] D Traum, "Computational Models of Grounding in Collaborative Systems," in *Working Papers of the AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems*, 1999, pp. 124–131.

[18] JD Williams and SJ Young, "Scaling up POMDPs for Dialogue Management: the Summary POMDP Method," in *IEEE workshop on Automatic Speech Recognition and Understanding (ASRU2005)*, Puerto Rico, 2005.

[19] JD Williams and SJ Young, "Scaling POMDPs for dialog management with composite summary point-based value iteration (CSPBVI)," in *AAAI Workshop on Statistical and Empirical Approaches for Spoken Dialogue Systems*, Boston, 2006.

[20] K Scheffler and SJ Young, "Automatic Learning of Dialogue Strategy using Dialogue Simulation and Reinforcement Learning," in *HLT 2002*, San Diego, USA, 2002.

[21] J Schatzmann, K Georgila, and SJ Young, "Quantitative Evaluation of User Simulation Techniques for Spoken Dialogue Systems," in *6th SIGdial Workshop on DISCOURSE and DIALOGUE*, Lisbon, 2005.