Prediction-directed Compression of POMDPs

Abdeslam Boularias
Department of Computer Science
Laval University, Canada, G1K 7P4
boularias@damas.ift.ulaval.ca

Masoumeh Izadi
School of Computer Science
McGill University, Canada, H3A 2A7
mtabae@cs.mcgill.ca

Brahim Chaib-draa
Department of Computer Science
Laval University, Canada, G1K 7P4
chaib@damas.ift.ulaval.ca

Abstract

High dimensionality of belief space in Partially Observable Markov Decision Processes (POMDPs) is one of the major causes that severely restricts the applicability of this model. Previous studies have demonstrated that the dimensionality of a POMDP can eventually be reduced by transforming it into an equivalent Predictive State Representation (PSR). In this paper, we address the problem of finding an approximate and compact PSR model corresponding to a given POMDP model. We formulate this problem in an optimization framework. Our algorithm tries to minimize the potential error that missing some core tests may cause. We also present an empirical evaluation on benchmark problems, illustrating the performance of this approach.

1 Introduction

A number of representations for dynamical systems have been proposed during the past two decades beside Partially Observable Markov Decision Processes (POMDPs). However, they either impose restrictions to the underlying environment, or they do not seem to provide any advantages over POMDPs. Among these representations, PSRs (Predictive State Representations) [8] seem appealing for several main reasons. First, PSRs are grounded in the sequence of actions and observations of the agent, and hence relate the state representation directly to the agent's experience. Another reason why the predictive approach is believed to be a good choice for state representation is related to the generalization problem. Indeed, the predictive representation does not rely on a specific physical layout of an unobservable environment, so it has the potential of being useful for fast adaptation to a new similar environment. Finally, PSRs offer a representation for dynamical systems which is as general as POMDPs, and can be potentially more compact than POMDPs [7]. This is particularly useful for planning and predicting observations in large domains, where the dimensionality of the state space is a major drawback. However, the equivalent PSR of a POMDP model has generally almost the same dimensionality as the original model [3]. Consequently, it is natural to look for approximation algorithms which generate a subset of core tests more appropriate for planning purposes. We are also interested in reducing the uncertainty by having a problem formulation as detailed as possible. These two goals involve a trade-off between the dimensionality of the model and the accuracy of the solution that can be obtained. The reduced PSR model, which contains only a subset of core tests, conveys this trade-off. In this paper, we address the issue of dynamically generating a subset of core tests for predictive representations in partially observable domains. We are interested in a lossy compressed PSR with a number of core tests fewer than the number of states in the corresponding POMDP. We formulate this problem as an optimization problem that minimizes the loss function related to prediction of observations and rewards.

2 Background

2.1 POMDPs

Formally, a POMDP is defined by the following components: a finite set of hidden states S; a finite set of actions A; a finite set of observations Z; a set of transition functions $\{T^a\}$, such that $T^a(s,s')$ is the probability that the agent will end up in state s' after taking action a in state s; a set of observation functions $\{O^{a,o}\}$, such that $O^{a,o}(s)$ gives the probability that the agent receives observation o after taking

action a and getting to state s'; and a reward function R, such that $R(s,a)$ is the immediate reward received when the agent executes action a in state s. Additionally, there can be a discount factor $\gamma \in [0,1]$, which is used to weigh less rewards received farther into the future. To simplify notations, we use the $|S| \times |S|$ transition and observation matrices $\{T^{a,o}\}$, where $T^{a,o}(s,s')$ is the probability that the agent will end up in state s' and observes o after taking action a in state s. We also consider a reward vector R depending only on states, $R(s)$ is the reward received when the agent is in state s.

Instead of memorizing a complete history h_t of interaction with the system (a sequence of actions and observations), a probability distribution over hidden states, called the belief state, is sufficient for making predictions about the future observations and rewards. The belief state is an $|S|$-dimensional vector b_t, where $b_t(s) = Pr(s_t = s|h_t), s \in S$.

After taking action a and receiving observation o, the agent updates its belief state using Bayes' Rule:

$$b_{t+1} = \frac{b_t^T T^{a,o}}{b_t^T T^{a,o} e_{|S|}} \tag{1}$$

where $e_{|S|} = (1,1,\ldots,1)^T$ is an $|S|$-dimensional vector.
The expected reward for a belief state b_t is given by:

$$R(b_t) = b_t^T R \tag{2}$$

2.2 PSRs

PSRs [8, 7] are an alternative model for representing partially observable environments without using hidden variables. The fundamental idea of PSRs is to replace the probabilities on states by probabilities on future trajectories, called *tests*. A test q is an ordered sequence of actions and observations $a_1 o_1 \ldots a_k o_k$. The probability of q starting after a history h_t is defined by:

$$Pr(q|h_t) = Pr(o_1,\ldots,o_k|h_t,a_1,\ldots,a_k)$$

The probability of any test q depends linearly on the probabilities of a few tests, called *core tests*. We use Q to indicate the set of core tests. The reduced belief state is a $|Q|$-dimensional vector $\tilde{b}_t$, where $\tilde{b}_t(q) = Pr(q|h_t), q \in Q$. The probability of any test q is given by:

$$Pr(q|h_t) = \tilde{b}_t^T m_q$$

where m_q is a Q-dimensional weigh vector which is specific to the test q and independent on the history h_t.
After executing action a and receiving observation o, the reduced belief state is updated by using Bayes' Rule:

$$\forall q \in Q : Pr(q|h_t ao) = \frac{Pr(aoq|h_t)}{Pr(ao|h_t)}$$

So:

$$\tilde{b}_{t+1} = \frac{\tilde{b}_t^T M_{ao}}{\tilde{b}_t^T m_{ao}} \tag{3}$$

M_{ao} is a $|Q| \times |Q|$ matrix where each column corresponds to a weigh vector m_{aoq} used to calculate the probability of the core test extension aoq. The weigh vector m_{ao} is used to calculate the probability of ao, which is also an extension of a special core test $\emptyset$ corresponding to the empty sequence.

The matrix containing the predictions of all core tests given the underlying states is called the transformation matrix U ($|S| \times |Q|$). We have $U(s,q) = Pr(q|s), q \in Q, s \in S$. U can be found by a simple search for the maximum number of linearly independent vectors $U(.,q)$. Littman et al. [8] presented a polynomial time algorithm that finds incrementally all core tests in an iterative fashion, given a POMDP model. Since the columns of U are linearly independent, then $|Q| \leqslant |S|$. The matrix U can be used to map any belief state b_t in the POMDP model to a reduced belief state $\tilde{b}_t$ in the corresponding PSR model:

$$\tilde{b}_t = b_t^T U \tag{4}$$

2.3 Predicting rewards with PSRs

In order to predict observations, we need only to know the core tests set Q, the initial belief $\tilde{b}_0$, and the vectors m_{aoq} and m_{ao}, but what about rewards? In previous work on planning with PSRs [2, 3], the immediate reward was treated as a part of the observation. Consequently, the dimensionality of the observation space is multiplied by the number of reward values, which can be very high since the rewards are real valued. To deal with this problem, we propose a new method for predicting rewards in PSRs.

We first define the expected final reward of a test q by:

$$V(q|h) = Pr(q|h) \sum_{s \in S} Pr(s|hq) R(s) \tag{5}$$

$V(q|h)$ represents the expected reward that we will get at the end of the test q. The expected final reward of any test q depends linearly on the expected final rewards of a few tests, that we call *reward core tests*. We use Q_r to indicate the set of reward core tests. Notice that Q_r is defined independently from Q, and can be different from Q. The reduced reward belief state is a $|Q_r|$-dimensional vector $\tilde{b}_{r_t}$, where $\tilde{b}_{r_t}(q) = V(q|h_t), q \in Q_r$. The expected final value of any test q is given by:

$$V(q|h_t) = \tilde{b}_{r_t}^T m_q$$

where m_q is a $|Q_r|$-dimensional weigh vector. We also use Bayes' Rule to update $V(q|h)$ when we execute an action a and receive an observation o:

$$\forall q \in Q_r : V(q|h_t ao) = \frac{V(aoq|h_t)}{Pr(ao|h_t)}$$

then:

$$\tilde{b}_{r_{t+1}} = \frac{\tilde{b}_{r_t}^T M_{ao}^r}{\tilde{b}_t^T m_{ao}} \quad (6)$$

M_{ao}^r is a $|Q_r| \times |Q_r|$ matrix where each column corresponds to a weigh vector m_{aoq_r} used to calculate the expected final value of the reward core test extension aoq_r. The PSR observation model is used to calculate $Pr(ao|h_t)$.

The matrix containing the expected final values of all reward core tests given the underlying states is indicated by U_r (an $|S| \times |Q_r|$ matrix). As for U, U_r can be found by a simple search for the maximum number of linearly independent columns $U_r(.,q_r)$, and we have also $|Q_r| \leqslant |S|$. The immediate reward is given by $V(\emptyset|h)$, the expected reward of the empty test $\emptyset$, so $U_r(.,\emptyset) = R$. We use m_r, the empty test weight vector, to find the expected immediate reward $V(\emptyset|h_t) = b_{r_t}^T m_r$ given a reduced reward belief state b_{r_t}.

In order to predict rewards, we need the PSR reward model $\langle Q_r, \tilde{b}_{r_0}, m_r, \{m_{aoq_r} : \forall a \in A, \forall o \in Z, \forall q_r \in Q_r\}\rangle$, and the PSR observation model $\langle Q, \tilde{b}_0, \{m_{aoq}, m_{ao} : \forall a \in A, \forall o \in Z, \forall q \in Q\}\rangle$. The reduced belief of the observation model is used to predict observations and to update the reduced belief of the reward model.

$$\begin{aligned} &\textit{minimize:} && c_1 \sum_{ao} \varepsilon_{ao} + c_2 \sum_{aoq} \varepsilon_{aoq} \\ &\textit{subject to:} \\ &&& \forall a \in A, \forall o \in Z : \\ &&& \|T^{ao} e_{|S|} - U m_{ao}\|_\infty \leqslant \varepsilon_{ao} \\ &&& \forall a \in A, \forall o \in Z, \forall q \in Q : \\ &&& \|T^{ao} U(.,q) - U m_{aoq}\|_\infty \leqslant \varepsilon_{aoq} \end{aligned} \quad (7)$$

$$\begin{aligned} &\textit{minimize:} && c_3 \varepsilon_r + c_4 \sum_{aoq_r} \varepsilon_{aoq_r} \\ &\textit{subject to:} \\ &&& \|R - U_r m_r\|_\infty \leqslant \varepsilon_r \\ &&& \forall a \in A, \forall o \in Z, \forall q_r \in Q_r : \\ &&& \|T^{ao} U_r(.,q_r) - U_r m_{aoq_r}\|_\infty \leqslant \varepsilon_{aoq_r} \end{aligned} \quad (8)$$

Table 1. The two linear programs used to find the approximate PSR parameters.

3 PSRs and Krylov subspaces

PSRs are related to another compression method known as Value-directed Compression (VDC) [6]. To show the link between PSRs and VDC, we will first review a linear algebraic concept called *Krylov subspace*.

Let $\mathcal{V}$ be a vector space containing the vector v, a Krylov subspace $Kr(M,v)$ is the subspace of $\mathcal{V}$ that contains the vector v and that is invariant with respect to the matrix M, i.e. $\forall u \in Kr(M,v) : Mu \in Kr(M,v)$. So $Kr(M,v) = \{v, Mv, MMv, MMMv, \dots\}$. This definition can be generalized to subspaces with many matrices $M_1, M_2, \dots, M_k$, a *Krylov* subspace $Kr(\{M_i\},v)$ is the subspace of $\mathcal{V}$ that contains the vector v and that is invariant with respect to the matrices $\{M_i\}$, i.e. $\forall M \in \{M_i\}, \forall u \in Kr(\{M_i\},v) : Mu \in Kr(\{M_i\},v)$. A basis of $Kr(M,v)$ is constructed by keeping the first linearly independent vectors of the list $\{v, Mv, MMv, \dots\}$. In fact, one can prove that if $Kr(M,v)$ is n-dimensional then $\forall m \geqslant n$: the vector $M^m v$ can be written as a linear combination of the vectors $M^k v, k < n$.

VDC algorithm finds a basis matrix F for the *Krylov* subspace $Kr(\{T^{a,o}\},R)$ by iteratively enumerating linearly independent vectors of this space. The first column of F is R, then all the vectors $T^{a_1 o_1} R, \forall a_1 \in A, \forall o_1 \in Z$ are generated, and the linearly independent vectors are added to F. In step k, we generate only the vectors $T^{a_k o_k} T^{a_{k-1} o_{k-1}} \dots T^{a_1 o_1} R$ such that $T^{a_{k-1} o_{k-1}} \dots T^{a_1 o_1} R$ are already in F, because we know that the extensions of linearly dependent vectors in a Krylov subspace are also linearly dependent vectors. This process is repeated until no more independent vectors can be added to F. The basis matrix F is used as a transformation matrix for mapping belief states into reduced belief states and planning within the reduced space.

By noticing that for any test $q = a_1 o_1 \dots a_k o_k$ we have $V(q) = U_r(.,q) = T^{a_k o_k} \dots T^{a_1 o_1} R$, we can conclude that $F = U_r$. Consequently, VDC algorithm returns the reward model of a PSR. As for the observation model, we can show that the transformation matrix U returned by Littman et al. [8] algorithm (which proceeds as in VDC) is in fact a basis of the Krylov subspace $Kr(\{T^{a,o}\}, e_{|S|})$. The probability vector of the empty test is given by $Pr(\emptyset) = e_{|S|}$, and the probability of any test q is given by $Pr(q) = U(.,q) = T^{a_k o_k} \dots T^{a_1 o_1} e_{|S|}$.

VDC algorithm can make predictions about future rewards, but cannot update the reduced belief states after receiving an observation o, thus it cannot be used for online planning for example. Our approach, Prediction-directed Compression (PDC), can be seen as a generalization of VDC for making predictions about rewards (with the PSR reward model) as well as about observations (with the PSR observation model), thus it can be used online.

4 Lossy compression of POMDPs with PSRs

Linear exact transformation is considered insufficient in practice. This is a motivation to further investigate a lossy

transformation which scales better. Building on the lossy compression version of VDC, we develop an algorithm for finding compact PSRs. Given a POMDP model and the required reduced dimensions $|Q|$ and $|Q_r|$, our algorithm finds the best parameters of an approximate PSR model that minimize the loss function, which measures the difference between the predictions of the accurate POMDP model and the predictions of the approximate PSR model. We use here the optimization equations (7) and (8) to find the parameters of the approximate PSR. These equations are nonlinear, but can be solved with a linear program by using an iterative technique. To achieve that, we alternate between solving the first LP presented in table (1) to find the parameters m_{ao} and m_{aoq} while keeping U fixed, and solving the same LP to find U while keeping the parameters m_{ao} and m_{aoq} fixed. We do the same thing for the second LP, used to find the reward model. In our first algorithm, the matrices U and U_r are initialized with random values. The parameters c_1, c_2, c_3 and c_4 are the weight associated to the errors on m_{ao}, m_{aoq}, m_r and m_{aoq_r} respectively, they are fixed *a priori*.

This algorithm needs many iterations before converging to an optimum, and it is often a local optimum, so we should try several random initializations before getting the global optimum. To alleviate this problem, we initialize the matrix U with the first $|Q|$ linearly independent vectors of $Kr(\{T^{a,o}\}, e_{|S|})$. Similarly, the matrix U_r is initialized with the first $|Q_r|$ linearly independent vectors of $Kr(\{T^{a,o}\}, R)$. These vectors can be found in polynomial time by a depth-first search in the spaces $Kr(\{T^{a,o}\}, e_{|S|})$ and $Kr(\{T^{a,o}\}, R)$. The remaining parameters of the approximate PSR model are found in one iteration by using several small linear programs. This idea is illustrated in Algorithm 1. Notice that if we set $|Q| = |Q_r| = |S|$ then our algorithm will return an exact PSR model that is equivalent to the original POMDP.

Input: A POMDP: (S, A, Z, T, R). The number of core tests $|Q| \leqslant |S|$. The number of reward core tests $|Q_r| \leqslant |S|$;
Output: The parameters U, U_r, m_{ao}, m_{aoq}, m_r, m_{aoq_r} of the compressed model, $\forall a \in A, \forall o \in Z, \forall q \in Q, \forall q_r \in Q_r$;
$U \leftarrow |Q|$ first independent vectors of $Kr(\{T^{a,o}\}, e_{|S|})$;
$U_r \leftarrow |Q_r|$ first independent vectors of $Kr(\{T^{a,o}\}, R)$;
for $a \in A$ **do**
  for $o \in Z$ **do**
    Find m_{ao} by solving the following LP:

$$\begin{aligned} &\textit{minimize } \varepsilon_{ao} \\ &\textit{s.t. } \|T^{ao} e - U m_{ao}\|_\infty \leqslant \varepsilon_{ao} \end{aligned}$$

    for $q \in Q$ **do**
      Find m_{aoq} by solving the following LP:

$$\begin{aligned} &\textit{minimize } \varepsilon_{aoq} \\ &\textit{s.t. } \|T^{ao} U(.,q) - U m_{aoq}\|_\infty \leqslant \varepsilon_{aoq} \end{aligned}$$

    end
    for $q_r \in Q_r$ **do**
      Find m_{aoq_r} by solving the following LP:

$$\begin{aligned} &\textit{minimize } \varepsilon_{aoq_r} \\ &\textit{s.t. } \|T^{ao} U_r(.,q_r) - U_r m_{aoq_r}\|_\infty \leqslant \varepsilon_{aoq_r} \end{aligned}$$

    end
  end
end
Find the vector m_r by solving the following LP:

$$\begin{aligned} &\textit{minimize } \varepsilon_r \\ &\textit{s.t. } \|R - U_r m_r\|_\infty \leqslant \varepsilon_r \end{aligned}$$

Algorithm 1: Prediction-directed Compression.

5 Analysis of the value approximation error

In POMDPs, every policy π is characterized by an $|S|$-dimensional vector called *value function* V_π. $V_\pi(s)$ is the expected cumulated reward of executing π from state s. Similarly, a reduced value function $\tilde{V}_\pi$ is a $|Q_r|$-dimensional vector defining the value of π over the reduced belief space.

We define ε_{π_t}, the error of the value function for a given policy π_t with horizon t, as the difference between the value vector V_{π_t} of π_t, according to the accurate POMDP model, and the value vector $U_r \tilde{V}_{\pi_t}$ of the same policy, according to the approximate PSR reward model, returned by PDC. Also, let $\varepsilon_{\pi_t^*}$ be the worst such error over the policies space.

$$\varepsilon_{\pi_t^*} = \max_{\pi_t} \varepsilon_{\pi_t} = \max_{\pi_t} \|V_{\pi_t} - U_r \tilde{V}_{\pi_t}\|_\infty \qquad (9)$$

This error is bounded as follows:

$$\begin{aligned}
\varepsilon_{\pi_t^*} &= \max_a \|(R - U m_r) + \gamma \sum_{o \in Z} \max_{\pi_{t-1}} (T^{ao} V_{\pi_{t-1}} - U_r M^r_{ao} \tilde{V}_{\pi_{t-1}})\|_\infty \\
&\leqslant \max_a \|R - U m_r\|_\infty + \gamma \sum_{o \in Z} \max_{\pi_{t-1}} \|T^{ao} V_{\pi_{t-1}} - U_r M^r_{ao} \tilde{V}_{\pi_{t-1}}\|_\infty \\
&\leqslant \varepsilon_r + \gamma \| \sum_{o \in Z} \max_{\pi_{t-1}} (T^{ao} V_{\pi_{t-1}} - T^{ao} U_r \tilde{V}_{\pi_{t-1}} \\
&\qquad - U_r M^r_{ao} \tilde{V}_{\pi_{t-1}} + T^{ao} U_r \tilde{V}_{\pi_{t-1}})\|_\infty \\
&\leqslant \varepsilon_r + \gamma \| \sum_{o \in Z} \max_{\pi_{t-1}} (T^{ao} V_{\pi_{t-1}} - T^{ao} U_r \tilde{V}_{\pi_{t-1}})\|_\infty \\
&\qquad + \gamma \sum_{o \in Z} \max_{\pi_{t-1}} \|T^{ao} U_r \tilde{V}_{\pi_{t-1}} - U_r M^r_{ao} \tilde{V}_{\pi_{t-1}}\|_\infty \\
&\leqslant \varepsilon_r + \gamma \| \sum_{o \in Z} \max_{\pi_{t-1}} ((T^{ao})(V_{\pi_{t-1}} - U_r \tilde{V}_{\pi_{t-1}}))\|_\infty \\
&\qquad + \gamma \sum_{o \in Z} \max_{\pi_{t-1}} \|T^{ao} U_r - U_r M^r_{ao}\|_\infty \|\tilde{V}_{\pi_{t-1}}\|_\infty \\
&\leqslant \varepsilon_r + \gamma \varepsilon_{\pi^*_{t-1}} \| \sum_{o \in Z} T^{ao}\|_\infty + \gamma \sum_{o \in Z} \max_{\pi_{t-1}} \varepsilon_{aoq_r} \|\tilde{V}_{\pi_{t-1}}\|_\infty \\
&\leqslant \gamma \varepsilon_{\pi^*_{t-1}} + \gamma |Z| \varepsilon_{aoq_r} \|\tilde{V}^*\|_\infty + \varepsilon_r \\
&\leqslant \frac{\varepsilon_r + \gamma |Z| \varepsilon_{aoq_r} \|\tilde{V}^*\|_\infty}{1 - \gamma}
\end{aligned}$$

Domain	Network	Shuttle	Grid	Coffee Reward	Hallway	Hallway2	Dialogue
Original dimension	7	8	16	32	60	92	433
Reduced dimension (RMSD)	1(0.1575)	1(0.3867)	1(0.4595)	1(1.3307)	5(0.1234)	5(0.1272)	2(0.0279)
	2(0.1224)	2(0.3966)	2(0.3052)	2(1.3316)	10(0.0572)	10(0.1361)	3(0.0255)
	3(0.0159)	3(0.3920)	3(0.1155)	3(0.3749)	15(0.0585)	15(0.0947)	5(0.0169)
	4(0.0228)	4(0.3877)	4(0.1445)	4(0.1065)	20(0.0590)	20(0.0392)	8(0.0000)
	5(0.0029)	5(0.2851)	5(0.1349)	5(0.1065)	25(0.0590)	25(0.0369)	10(0.0000)
	6(0.0067)	6(0.1479)	6(0.1009)	6(0.0027)			

Table 2. Average error of PDC on predicting observations, as function of the reduced dimension.

Domain	Hallway	Hallway2	Spoken Dialogue	Coffee Observations	Coffee Reward	Coffee Total
Reduced dimension (time)	5 (20.59)	5 (20.46)	2 (350.93)	1(0.14)	1(0.22)	1(0.36)
	10(35.42)	10(41.92)	3(557.09)	2(0.20)	2(0.36)	2(0.56)
	15(60.87)	15(69.46)	5(759.2)		3(0.48)	3(0.68)
	20(74.47)	20(107.28)	8(1166.04)		4(0.78)	4(0.98
	25(94.93)	25(157.73)	10(1459.56)		5(0.95)	5(1.15)
	30(114.9)	30(184.23)			6(0.98)	6(1.18)

Table 3. Runtime of PDC in seconds, as function of the reduced dimension.

where $\|\tilde{V}^*\|_\infty = \max_\pi \|\tilde{V}_\pi\|_\infty$. In these substitutions, inspired by the proof of Poupart [5], we used the properties: $\|A+B\|_\infty \leqslant \|A\|_\infty + \|B\|_\infty$ and $\|AB\|_\infty \leqslant \|A\|_\infty \|B\|_\infty$.

6 Empirical evaluation

6.1 Predicting observations and rewards

A complete model of a system should be able to predict the probability of an arbitrary observation o and the expected immediate reward, after a history h_i. After finding the approximate PSR model of a given problem by the means of Algorithm 1, we test the accuracy of this model by simulating a train of t steps and compare the predictions of the reduced model to the predictions of the original POMDP. The initial belief state b_0 of the POMDP is generated randomly, and the initial reduced belief states are given by $\tilde{b}_0 = b_0^T U$, $\tilde{b}_{r_0} = b_0^T U_r$. At each step i, we sample an action a uniformly, and calculate the prediction $Pr(o|a,h_i)$ of the exact model and the prediction $\hat{Pr}(o|a,h_i)$ of the approximate model, for each observation $o \in Z$. We also calculate the expected rewards $R(h_i)$ and $V(\emptyset|h_i)$. Then, we sample the next underlaying state and generate an observation o according to T^{ao}. The observation o and the action a are used by both models to update their internal beliefs, each one using its own parameters and update function.

The root mean squired deviation (RMSD) computes the deviation between the predictions of the reduced model and the predictions of the accurate model. It is given by:

$$RMSD_o = \sqrt{\frac{1}{t \times |Z|} \sum_{i=1}^{t} \sum_{o=1}^{|Z|} (\hat{Pr}(o|h_i,a_i) - Pr(o|h_i,a_i))^2}$$

$$RMSD_r = \sqrt{\frac{1}{t} \sum_{i=1}^{t} (V(\emptyset|h_i) - R(h_i))^2}$$

In our experiments, we have considered $t = 100$, this is sufficient because most of the planning algorithms generally do not exceed this horizon. However, the error on the predictions become more important as the horizon grows, this is due to the fact that the update function of reduced belief states uses approximate values, so, after a long sequence, the cumulated errors make the PSR belief state relatively distant from the POMDP belief state.

We tested our compression algorithm on 7 different standard benchmarks taken from the literature [1, 6]: Coffee domain (32 states, 2 actions, 3 observations), 4×4 grid (16 states, 4 actions, 2 observations), Shuttle (8 states, 3 actions, 5 observations), Network (7 states, 4 actions, 2 observations), Hallway problem (60 states, 5 actions, 21 observations), Hallway2 problem (92 states, 5 actions, 17 observations), and Spoken Dialogue problem (433 states, 37 actions, 16 observations). Table 2 presents the average error on predicting observations as function of the number of core tests used for every problem. This error is contained in the interval $[0,1]$ because it measures a difference between two probabilities. For Coffee problem, we reported only the error on predicting rewards, the error on predicting observations becomes null by using only 2 core tests. For the other problems, we reported the error on predicting observations, because the error on rewards was almost null even when we use only one core test. In these specific domains, all the states have a null reward, except for the final state. With a random policy, we could rarely get to these final states, so the cumulated reward was null for both PSR and POMDP.

For small problems (Shuttle, Network and 4×4 grid), we used the first version of PDC, where all the parameters of the approximate PSR are found by solving several LPs. With larger domains (Coffee, Hallway, Hallway2 and Spo-

	RTBSS with Approximate PSR				RTBSS with POMDP	
Domain	$\|Q\|$	$\|Q_r\|$	Runtime per step	Average reward	Runtime per step	Average reward
Coffee	1	1	0.002	-1.99	0.35	-1.49
	2	2	0.007	-1.49		
Hallway	20	20	20.70	0.03	27.58	0.06
	40	40	15.89	0.06		
Hallway2	20	20	1.56	0.00	80.20	0.02
	40	40	5.70	0.01		

Table 4. Average reward and runtime in milliseconds of RTBSS using POMDP and Approximate PSR models, as function of the reduced dimension.

ken Dialogue), this approach becomes untractable, so we set the columns of U and U_r to the linearly independent vectors of $Kr(\{T^{a,o}\}, e_{|S|})$ and $Kr(\{T^{a,o}\}, R)$ (see Algorithm 1).

The effective compression time for each large problem is reported in Table 2, the experiments were performed using ILOG Cplex 10 solver on an AMD Athlon machine with a 1.80 GH processor and 1.5 GB ram.

In all the experiments, the compression of the POMDP was successful, because we needed a few core tests to construct an approximate PSR model with low prediction error. For the 4×4 grid problem for example, we used only 6 core tests instead of 16 to make predictions on future observations with an average error of 0.10 over the 100 steps. A random prediction makes an average error of 0.55 in most of the problems. For Network problem, we can predict the future observations of the system by using only 3 tests, while the prediction error is nearly null (0.02). For the Coffee domain, PSR model is able to predict the expected reward with an almost null error, by using 6 core tests. This gain is even more interesting in larger domains, such as Hallways and Spoken Dialogue. For this latter domain, the prediction error was almost null with a few core tests, because the observations depend only on the initial belief state, which is a uniform distribution over states (In this problem, the final state is reached after only 2 or 3 actions in general).

We notice also that in general, the average error decreases when the number of core tests grows, because with more variables, the LP has higher degree of freedom, and can adjust the parameters in order to minimize more the loss function. The error becomes null when $|Q| = |Q_r| = |S|$.

6.2 Online planning

We tested our approach on the problem of online planning. To do so, we implemented RTBSS (Real-Time Belief Space Search) [4] algorithm and updated it for dealing with approximate PSRs. RTBSS is based on a look-ahead search that is applied online each time the agent has to make a decision, i.e. after each observation. RTBSS is particularly interesting for large real-time environments where offline solutions are not applicable because of their complexity. In our case, we set the search depth to 3, and used *Blind Policy* as lower bound heuristic and *Q-MDP* as upper bound. The adaptation of RTBSS for dealing with approximate PSRs is straightforward, since we already provided a mechanism for updating reduced belief states, and calculating the expected reward corresponding to such belief states. However, Blind Policy and Q-MDP heuristics are defined on states and it is not obvious how to use these heuristics in PSRs since the states are replaced by core tests. Nevertheless, these values can be treated in the same way we treated immediate rewards and preserved during the compression. Let U and L be two $|S|$-dimensional vectors such that $U(S)$ is an upper bound on the value of state s and $L(S)$ is a lower bound on the value of state s. L and U are calculated offline by using the provided POMDP model. Let u and l be two $|Q_r|$-dimensional vectors such that $\tilde{b}_{r_t}^T u$ and $\tilde{b}_{r_t}^T l$ are respectively an upper and a lower bounds on the value of the reduced belief state $\tilde{b}_{r_t}$. The vectors u and l are found by adding the following constraints to the linear program (8) of Table 1:

$$\begin{aligned} \|L - U_r l\|_\infty &\leqslant \varepsilon_l \\ \|U - U_r u\|_\infty &\leqslant \varepsilon_u \end{aligned}$$

Table 4 illustrates the reward and runtime per decision step of RTBSS using both POMDP and Approximate PSR models, as function of the reduced dimension. These values are averaged over 100 decision steps. For the Coffee domain, we can see that we need at most two observation core tests and two reward core tests to get the same average reward as the exact POMDP model, while the runtime per step is significantly reduced. The results for Hallways are also very promising. In fact the runtime is always reduced and the average reward is close to the reward of POMDP. Notice that for Hallway2, we apparently need higher dimensions in order to achieve the same results as with the exact POMDP.

7 Conclusions and future work

Current work on planning methods for POMDPs demonstrates that finding optimal policies for POMDPs is untractable in practice. In this paper, we investigated the possibility of finding an approximate reduced PSR model from

a given larger POMDP. We formulated this problem in linear programming framework. We illustrated a theoretical bound for value function error using the approximate PSR model. Using an approximate model seems to have a definite advantage in PSRs as confirmed by our experimental results. Our results show that the reduced model can predict the observations probabilities and rewards values with high precision compared to the exact model. The impact of this method is more pronounced for problems with special structure. However, there are more potential advantageous in applying reduced PSRs instead of POMDPs for planning. Preliminary results on using approximate PSR in online planning show that we can always find a reduced dimension where planning with approximate PSR takes less time than with the original POMDP, without important loss in reward. The immediate next step to this research is to perform more elaborated experiments on using the approximate PSR in planning, with larger and structured domains. The other extension is to study how to efficiently select the most promising independent vectors of Krylov subspaces that are used to initialize the matrices U and U_r.

References

[1] A. Cassandra. *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, Brown University, Providence, USA, 1998.

[2] M. Izadi. *On Knowledge Representation and Decision Making under Uncertainty.* PhD thesis, McGill University, Montreal, Canada, 2007.

[3] M. James, S. Singh, and M. Littman. Planning with Predictive State Representations. In *Proceedings of 2004 International Conference on Machine Learning and Applications (ICML'04)*, pages 304–311, 2004.

[4] S. Paquet, L. Tobin, and B. Chaib-draa. An online POMDP algorithm for complex multiagent environments. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS'05)*, pages 970–977, 2005.

[5] P. Poupart. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes.* PhD thesis, Toronto, Canada, 2005.

[6] P. Poupart and C. Boutilier. Value-Directed Compression of POMDPs. In *Advances in Neural Information Processing Systems (NIPS'02)*, pages 1547–1554, 2002.

[7] S. Singh, M. James, and M. Rudary. Predictive State Representations: A New Theory for Modeling Dynamical Systems. In *Uncertainty in Artificial Intelligence: Proceedings of the Twentieth Conference (UAI'04)*, 2004.

[8] S. Singh, M. Littman, N. Jong, D. Pardoe, and P. Stone. Learning Predictive State Representations. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML'03)*, 2003.