POMDP Homomorphisms

Alicia Peregrin Wolfe
Department of Computer Science
University of Massachusetts, Amherst
Amherst, MA 01003
pippin@cs.umass.edu

Abstract

The problem of finding hidden state in a POMDP and the problem of finding abstractions for MDPs are closely related. In this paper, we analyze the connection between existing Predictive State Representation methods [3] and homomorphic reductions of Markov Processes [5, 6]. We formally define a POMDP homomorphism, then extend PSR reduction methods to find POMDP homomorphisms when the original POMDP is known. The resulting methods find more compact abstract models in tasks for which different observations have the same meaning.

1 Introduction

The task of representing hidden state in a POMDP is closely related to the task of finding an abstract model for an MDP: in both cases, we search for a model which does the best job possible of predicting some output, while remaining as small as possible.

The two problems have different approaches, however. In a POMDP we expand the state space to include extra distinctions between hidden states which are originally indistinguishable by the agent, while keeping the number of hidden states as small as possible. In model minimization, we eliminate distinctions between states while maintaining our ability to predict reward, or some other function of the state.

In some cases, however, a POMDP may also contain redundant observation information. Even in a POMDP it may not matter, for example, whether the mug the agent is holding is solid black or has pictures of pandas on it, as long as it is heat resistant. There is no reason for the POMDP to model separately the extra detail it would take to predict the decorations on a specific mug if either will do for the task of preparing coffee. We can, therefore, map the observation of holding a solid black mug and the observation of holding a panda mug to the same abstract observation — that we are holding a mug.

To find these observation mappings, the observations of the POMDP are divided into two parts: the part to predict, represented by an output function $y : O \rightarrow Y$, and the remainder, which is retained only when it helps to predict y. If this remaining part of the observation space is X, the complete observation space O is $Y \times X$.

In this paper we examine algorithms for finding the reduction from a known POMDP M to an abstract POMDP $\tilde{M}$ in which X is reduced to the most compact representation which yields a good prediction of Y. The intention in doing so is not to come up with a practical algorithm for finding $\tilde{M}$ — in practice inferring the abstract model directly from data without knowing M is more desirable. However, the precise definition of a valid reduction and algorithms for finding it from a known POMDP will hopefully enable better analysis of methods for finding such models from data, and ideally point the direction to good practical algorithms.

2 Background: CMP Homomorphisms

A Controlled Markov Process (CMP) is an MDP without the latter's reward function. A CMP with output is a CMP together with an output function that maps its state set to a set of outputs, or observations, Y.

We think of the output function as singling out some aspect of the CMP as being of interest for some reason. This function might be as simple as the location or color of an object in the state. Thus, a CMP with output is a tuple (S, A, T, y), where S, A, and T are as in an MDP, and y is the output function $y : S \times A \to Y$. Given any function $r : Y \to \mathbb{R}$, $(S, A, T, r \circ y)$ is an MDP whose reward function is the composition of r and y. We say that this MDP is *supported by* y.

A CMP homomorphism [6] is a mapping h from a CMP with output (S, A, T, y) to an abstract CMP with output (S', A', T', y'). h must preserve both the output function and some properties of the transition probabilities of M. Specifically, h consists of a set of mappings: $f : S \to S'$, and for each $s \in S$ a mapping $g_s : A \to A'$ that recodes actions in a possibly state-dependent way. The following properties must hold for all state and action pairs:

$$y'(f(s), g_s(a)) = y(s, a). \quad (1)$$

$$T'(f(s_i), g_{s_i}(a), f(s_j)) = \sum_{s_k \in f^{-1}(s_j)} T(s_i, a, s_k). \quad (2)$$

where $f^{-1}(s) = \{s_k | f(s) = f(s_k)\}$.

CMP Homomorphisms are a natural extension of Model Minimization [1] and MDP Homomorphisms [5].

The model formed by a CMP homomorphism can be used to learn the value function for any supported reward ($r \circ y$) and termination ($\beta \circ y$) functions. The optimal policy can then be *lifted* from M' to M.

3 Background: Linear PSRs

A POMDP is a tuple (S, A, Π, O), where S and A are defined as for MDPs, and Π is the transition function. However, S is not observable by the agent. Instead, each state generates a distribution over observations $P(o|s)$. The agent usually maintains a *belief state* or vector of probabilities $b_i = P(s_i)$ representing its belief about its current state. In a Predictive State Representation, the agent instead maintains a list of probabilities for a set of predictive tests.

A test t is a sequence of future action/observation predictions: $\{a_1, o_1, ...a_n, o_n\}$. A history h is some past sequence of actions and observations. The following notational shorthand is used: $P(t|h) = P(o_1...o_n|h, a_1, ...a_n)$.

The PSR representation is based on maintaining the probabilities of a small set of tests $Q = \{q_1, ...q_n\}$ which forms a basis for all other tests. In the case of a linear PSR this means that for all tests t, $P(t|h) = P^T(Q|h)m_t$ for some weight vector m_t.

The vector of Q predictions must also be a sufficient statistic for itself at the next time step. In order to update Q from one time step to another, the update rule is:

$$P(q_i|hao) = \frac{P(aoq_i|h)}{P(ao|h)} = \frac{P^T(Q|h)m_{aoq_i}}{P^T(Q|h)m_{ao}}.$$

The smallest complete representation of the PSR therefore includes a set of core tests Q and a set of m_t vectors for each ao pair, as well as each aoq_i sequence.

4 POMDP Homomorphisms

In a POMDP homomorphism the output function $y : O \to Y$ maps the complete observation space onto a simpler function, such as the location of a specific object or the agent's energy level. Rather

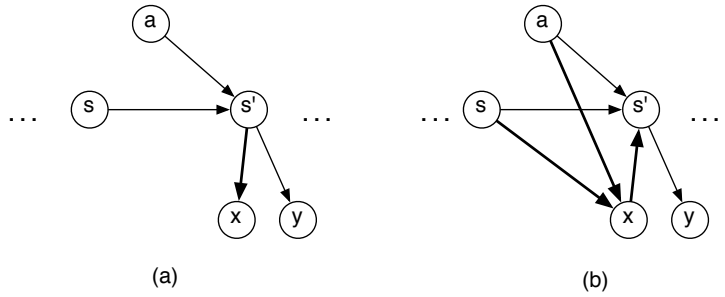

Figure 1: Two alternate graphical models for a single transition from time t to $t+1$ in a POMDP. The first model (a) is the canonical form, while in (b) the link between the next state s' and the x portion of the observation has been reversed.

than define a homomorphic mapping over states and actions, POMDP homomorphisms map the actions and observations of the original POMDP to abstract action A' and observation O' sets. The goal is to find a mapping which enables the agent to predict y as well as it could using the original POMDP.

MDP/CMP Homomorphisms are defined by two sets of constraints, over the output function and transition function. POMDP Homomorphisms are defined by three sets of constraints, since in addition to the output function and transition function, the abstract observation function must obey certain constraints in order to be consistent with the original POMDP model.

Define a state mapping $f : S \rightarrow \tilde{S}$, action mapping $g : A \rightarrow \tilde{A}$, and action specific observation mapping $k_a : O \rightarrow \tilde{O}$. Each mapping may be many to one. Together, these mappings define a mapping from the original POMDP (S, A, Π, O) to an abstract POMDP $(\tilde{S}, \tilde{A}, \tilde{\Pi}, \tilde{O})$.

In order to be a valid homomorphism, the mappings must obey several constraints. The first constraint over these mappings is identical to the first CMP Homomorphism constraint: all states in the same block of the partition must have the same distribution for the output variable:

$$P(y|s) = P(y|f(s)) \tag{3}$$

The final two constraints encode consistency in the transition and observation functions. Predictions of abstract next states and abstract observations must be of the same quality in the abstract model as they would be given the true state and action:

$$\forall \tilde{s} \in \tilde{S}, P(\tilde{s}|f(s), g(a), k_a(o)) = \sum_{s_i \in f^{-1}(s_{t+1})} P(s_i|s, a, o) \tag{4}$$

$$\forall \tilde{o} \in \tilde{O}, P(\tilde{o}|s, a) = \sum_{o_i \in k_a^{-1}(\tilde{o})} P(o_i|s, a). \tag{5}$$

It is important to note that the final constraint indicates that all actions which map to the same abstract action must have the same set of abstract observations with non-zero probability.

The constraints involve a slightly different view of POMDP state prediction than is typically used. The more typical POMDP model is represented in Figure 1 (a). The "current" state in this model is used to predict the output. The POMDP "transition" function T^{ao} is defined as $P(s', o|s, a) = P(o|s') \cdot P(s'|s, a) = P(o|s') \cdot P(s'|s, a)$.

In Figure 1 (b), the link from s' to o has been reversed to reflect the fact that o's main purpose is to provide information about s'. This reversal necessitates the addition of links from s and a to o in order to represent the original distribution. The result is a network in which all the directed paths in the model for each time slice end at y. It is then possible to reduce the model by following the reverse paths back from y, and "reducing" each link. This results in the factorization $P(s', o|s, a) = P(o|s, a) \cdot P(s'|s, a, o)$ which can easily be seen to correspond to the constraints 4 and 5.

Note that these constraints do not allow for state dependent action mappings. While state specific action and observation recodings are possible in POMDPs, their construction is complex: if the agent can, at the end of any given history h, believe that it could be in either s_i or s_j, s_i and s_j must have the same action and observation mapping. It is more appropriate, therefore, to consider creating history specific action and observation recodings. This is, for the most part, beyond the scope of this paper — this work assumes that the action and observation recodings are the same in all states. Observation mappings, however, may depend on the last action taken, which is a type of short one step history dependence.

In order to find a homomorphic reduction, the state, action and observation spaces can be repeatedly partitioned to find a mapping that satisfies these constraints, in a similar but slightly more complex fashion to the CMP Homomorphism finding algorithms. The algorithm has the following steps, if f_{old} is the state mapping from the previous iteration:

1. Partition S such that constraint 3 is satisfied
2. Partition S such that equivalent states have the same action set available: in other words, such that constraints 4 and 5 are satisfied ($P(f_{old}(s')|s,a,x) = P(f_{old}(s')|f(s),a,x)$ and $P(k_a(x)|s,a) = P(k_a(x)|f(s),a)$).
3. Partition the (a,x) elements of $A \times X$ such that $P(f_{old}(s')|f(s),a,x) = P(f_{old}(s')|f(s),g(a),k_a(x))$. This creates a mapping consistent with constraint 4 in all states.
4. Partition A such that constraint 5 is satisfied for all states, given the a,o partition from the last step (i.e., $P(k_a(x)|f(s),a) = P(k_a(x)|f(s),g(a))$).
5. loop back to (2) until the partition is stable (at worst, the partition will separate every individual state, action, and observation).

This algorithm finds a mapping from one POMDP to a potentially smaller abstract POMDP, but retains the state-based structure of the POMDP. Alternatively, we can combine the linear PSR algorithms and this reduction to form an algorithm which takes advantage of both types of compression simultaneously.

5 Linear PSR Homomorphisms

The algorithm in [3] provides a good starting point for a linear PSR style POMDP homomorphism finding algorithm. This original algorithm proceeds in stages: at iteration j, it finds a set of predictions Q_j which is a basis set for all j-step predictions in the POMDP.

Rather than predicting all j-step observation sequences, however, a homomorphism finding algorithm should predict *abstract* j-step observations. Each Q_j defines a state mapping, where any two states which have the same predictions for all basis vectors are mapped to the same state. For example, in the following table of prediction probabilities, if q_1 and q_2 are the tests in Q (their predictions are the basis vectors for all other predictions), any states with the same predictions for these two tests map to the same state for the current set of tests. In the example, s_1 and s_3 map to the same abstract state, but s_2 does not.

Table 1: default

	q_1	...	q_2	...
s_1	0.3		0.2	
s_2	0.4		0.5	
s_3	0.3		0.2	

Q_j is not built based on the true POMDP, however. Instead, we start with a POMDP which does not use all predictions, and over multiple iterations refine the observation space of the POMDP to improve predictions of y.

We define $\hat{M}_j$ to be an abstract POMDP which ignores some information about x. More specifically, this POMDP ignores the dependence of x on the previous state, and only models its ability to predict

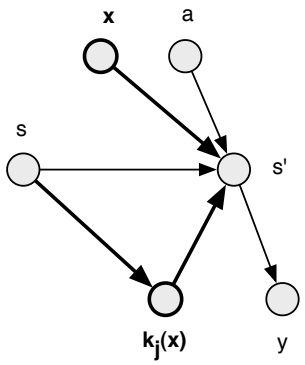

Figure 2: The graphical model for the approximate POMDP $\hat{M}_j$

the next state (removing the link from s to x in Figure **??**). The effect is to treat x as an action, rather than as an observation: $\hat{M}_j$ is the POMDP you get by ignoring $P(x|s)$ and assuming the agent can set x. It is only the current abstract observation, $k_a(x)$, which is modeled as a true observation which depends on the previous state. The resulting model is shown in Figure 2, with changes from the original model highlighted in bold. Initially, $k_1(x)$ maps all observations to a single abstract observation: each iteration of the algorithm refines this abstraction until further refinements do not improve the prediction of y.

In order to partition the a's and x's, the algorithm examines m_{aoq}/m_{ao} values in for the true action/observations, and the current abstract basis set Q_j. Those which have different predictions are split. At the end of the algorithm, any pair of a, o values for which m_{aoq}/m_{ao} is equivalent for all $q \in Q$ is mapped to the same abstract $\tilde{a}, \tilde{o}$.

The proof that this algorithm finds a representation which satisfies the constraints for a POMDP homomorphism is beyond the scope of this paper.

6 Output Function vs. Value Function Homomorphisms

There is yet another way to use the algorithm in the proceeding section, if the most compact representation for only one specific task is desired. Initializing the basis vectors as in [4] results in a model for a specific reward function. The only initial basis vector in this case is a single vector comprised of the immediate reward for each state. The original algorithm in [4] predicts all outputs of the POMDP, as well as the value function. However, when combined with the modifications listed in the previous section, the algorithm predicts only the essential aspects of the observation function needed to predict the value function.

7 Results

The following results compare the following 4 algorithms:

1. The linear PSR reduction finding algorithm from [3].
2. Output homomorphism finding algorithm ((1) with observation mappings added).
3. The value function reduction algorithm from [4].
4. Value function homomorphism finding algorithm ((3) with observation mappings added).

There were two sets of experiments on the same POMDP. shown in figure 3. In the first experiment, shown in Table 3, $a = b = c = 1$, thus, both y and the value function for the reward $r = y$ are identical at states 6 and 7. In the second experiment, shown in Table 2, $a = 2, b = 1, c = 1.5$. Thus, if $r = y$ the one step value at states 6 and 7 is the same, though the prediction of y is not.

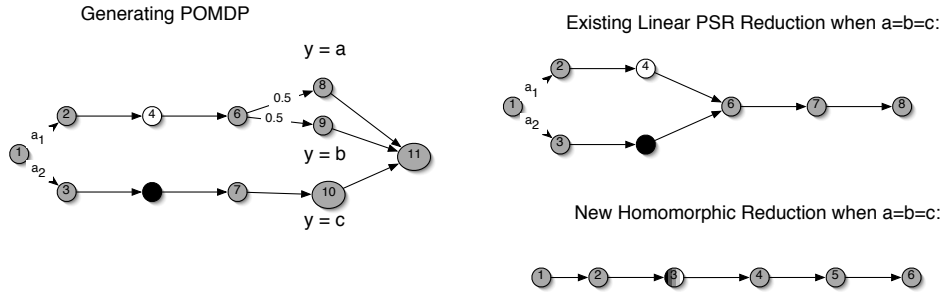

Figure 3: Simple POMDP. Unless otherwise noted, transitions are deterministic and identical for actions a_1 and a_2. The color of nodes is observed ($X = \{black, white, grey\}$). Note that the homomorphic reduction merges the observations of $white$ and $black$ into a single $white \vee black$ observation, while the Linear PSR reduction does not.

Table 2: $a = b = c = 1$

Algorithm	Basis Set Size	Abstract State Space Size	Percent of Original Size
PSR	7	8	0.73
PSRHM	5	6	0.55
VAL	7	8	0.73
VALHM	5	6	0.55

The difference between the algorithms consists essentially of whether they merge the observations $white$ and $black$ at states 4 and 5 into a single abstract observation to form one long chain of states. Note that in these experiments, the difference between the size of the Homomorphic reduction and the Linear PSR reduction can be made arbitrarily large by increasing the number of states in the chain between state 1 and states 4 and 5 (where the Linear PSR splits the state space unnecessarily).

8 Discussion and Future Work

Further work is necessary to determine which $\hat{M}$ definition is more useful. In the experiments thus far, their performance is identical, however, these were very simple experiments. Treating x as an action, rather than ignoring it, is more expensive, since there are more "actions" to prepend to the prediction basis vectors. However, it seems likely that this is the method which will extend more easily to the case where the true POMDP model is not known.

Tests on more complex domains, such as Blocks World and Towers of Hanoi are also essential to be sure the algorithm scales. The most interesting algorithmic extension will be finding history (rather than state) specific action and observation mappings, so that symmetries between regions of the state space where the action and observation mappings are not identical may be identified.

References

[1] Thomas Dean and Robert Givan. Model minimization in markov decision processes. In *Proceedings of AAAI*, 1997.

[2] Masoumeh T. Izadi and Doina Precup. Model minimization by linear psr. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 2005.

[3] Michael L. Littman, Richard S. Sutton, and Satinder P. Singh. Predictive representations of state. In *Advances In Neural Information Processing Systems*, volume 14, 2001.

[4] Pascal Poupart and Craig Boutilier. Value-directed compression of pomdps. In *Advances in Neural Information Processing Systems 15 (NIPS)*, pages 1547 –1554, Vancouver, BC, 2002.

[5] B Ravindran. *An Algebraic Approach to Abstraction in Reinforcement Learning*. PhD thesis, University of Massachusetts, 2004.

Table 3: $a = 2, b = 1, c = 1.5$

Algorithm	Basis Set Size	Abstract State Space Size	Percent of Original Size
PSR	10	11	1.0
PSRHM	10	11	1.0
VAL	7	10	0.91
VALHM	5	8	0.73

[6] Alicia Peregrin Wolfe and Andrew G. Barto. Decision tree methods for finding reuseable mdp homomorphisms. In *Proceedings of the 21st National Conference on Artificial Intelligence*, 2006.