Abstraction and Refinement for Solving Continuous Markov Decision Processes

Alberto Reyes[1] and Pablo Ibargüengoytia
Inst. de Inv. Eléctricas
Av. Reforma 113, Palmira, Cuernavaca, Mor., México
{areyes,pibar}@iie.org,mx

L. Enrique Sucar and Eduardo Morales
INAOE
Luis Enrique Erro 1, Sta. Ma. Tonantzintla, Pue., México
{esucar,emorales}@inaoep.mx

Abstract

We propose a novel approach for solving continuous and hybrid Markov Decision Processes (MDPs) based on two phases. In the first phase, an initial approximate solution is obtained by partitioning the state space based on the reward function, and solving the resulting discrete MDP. In the second phase, the initial abstraction is refined and improved. States with high variance in their value with respect to neighboring states are partitioned, and the MDP is solved locally to improve the policy. In our approach, the reward function and transition model are learned from a random exploration of the environment, and can work with both, pure continuous spaces; or hybrid, with continuous and discrete variables. We demonstrate empirically the method in several simulated robot navigation problems, with different sizes and complexities. Our results show an approximate optimal solution with an important reduction in state size and solution time compared to a fine discretization of the space.

1 Introduction

Markov Decision Processes (MDPs) have developed as a standard method for decision-theoretic planning. Traditional MDP solution techniques have the drawback that they require an explicit state representation, limiting their applicability to real-world problems. Factored representations (Boutilier et al., 1999) address this drawback via compactly specifying the state-space in factored form by using dynamic Bayesian networks or decision diagrams. Such Factored MDPs can be used to represent in a more compact way exponentially large state spaces. The algorithms for planning using MDPs, however, still run in time polynomial in the size of the state space or exponential in the number of state-variables; and do not apply to continuous domains.

Many stochastic processes are more naturally defined using continuous state variables which are solved as Continuous Markov Decision Processes (CMDPs) or general state-space MDPs by analogy with general state-space Markov chains. In this approach the optimal value function satisfies the Bellman fixed point equation:

$$V(s) = max_a[R(s,a) + \gamma \int_{s'} p(s'|s,a)V(s')ds'].$$

Where s represents the state, a a finite set of actions, R the immediate reward, V the expected value, P the transition function, and γ a discount factor.

The problem with CMDPs is that if the continuous space is discretized to find a solution, the discretization causes yet another level of ex-

[1] Ph.D. Student at ITESM Campus Cuernavaca, Av. Reforma 182-A, Lomas, Cuernavaca, Mor., México

ponential blow up. This "curse of dimensionality" has limited the use of the MDP framework, and overcoming it has become a relevant topic of research. Existing solutions attempt to replace the value function or the optimal policy with a finite approximation. Two recent methods to solve a CMDPs are known as grid-based MDP discretizations and parametric approximations. The idea behind the grid-based MDPs is to discretize the state-space in a set of grid points and approximate value functions over such points. Unfortunately, classic grid algorithms scale up exponentially with the number of state variables (Bonet and Pearl, 2002). An alternative way is to approximate the optimal value function $V(x)$ with an appropriate parametric function model (Bertsekas and Tsitsiklis, 1996). The parameters of the model are fitted iteratively by applying one step Bellman backups to a finite set of state points arranged on a fixed grid or obtained through Monte Carlo sampling. Least squares criterion is used to fit the parameters of the model. In addition to parallel updates and optimizations, on-line update schemes based on gradient descent, e.g., (Bertsekas and Tsitsiklis, 1996) can be used to optimize the parameters. The disadvantages of these methods are their instability and possible divergence (Bertsekas, 1995).

Several authors, e.g., (Dean and Givan, 1997), use the notions of abstraction and aggregation to group states that are similar with respect to certain problem characteristics to further reduce the complexity of the representation or the solution. (Feng et al., 2004) proposes a state aggregation approach for exploiting structure in MDPs with continuous variables where the state space is dynamically partitioned into regions where the value function is the same throughout each region. The technique comes from POMDPs to represent and reason about linear surfaces effectively. (Hauskrecht and Kveton, 2003) show that approximate linear programming is able to solve not only MDPs with finite states, but also be successfully applied to factored continuous MDPs. Similarly, (Guestrin et al., 2004) presents a framework that also exploits structure to model and solve factored MDPs although he extends the technique to be applied to both discrete and continuous problems in a collaborative setting.

Our approach is related to these works, however it differs on several aspects. First, it is based on *qualitative models* (Kuipers, 1986), which are particularly useful for domains with continuous state variables. It also differs in the way the abstraction is built. We use training data to learn a decision tree for the reward function, from which we deduce an abstraction called *qualitative states*. This initial abstraction is refined and improved via a local iterative process. States with high variance in their value with respect to neighboring states are partitioned, and the MDP is solved locally to improve the policy. At each stage in the refinement process, only one state is partitioned, and the process finishes when any potential partition does not change the policy. In our approach, the reward function and transition model are learned from a random exploration of the environment, and can work with both, pure continuous spaces; or hybrid, with continuous and discrete variables.

We have tested our method in simulated robot navigation problems, in which the state space is continuous, with several scenarios with different sizes and complexities. We compare the solution of the models obtained with the initial abstraction and the final refinement, with the solution of a discrete MDP obtained with a fine discretization of the state space, in terms of differences in policy, value and complexity. Our results show an approximate optimal solution with an important reduction in state size and solution time compared to a fine discretization of the space.

The rest of the paper is organized as follows. The next section gives a brief introduction to MDPs. Section 3 develops the abstraction process and Section 4 the refinement stage. In Section 5 the empirical evaluation is described. We conclude with a summary and directions for future work.

2 Markov Decision Processes

A Markov Decision Process (MDP) (Puterman, 1994) models a sequential decision problem, in which a system evolves in time and is controlled by an agent. The system dynamics is governed by a probabilistic transition function that maps states and actions to states. At each time, an agent receives a reward that depends on the current state and the applied action. Thus, the main problem is to find a control strategy or *policy* that maximizes the expected reward over time.

Formally, an MDP is a tuple $M =< S, A, \Phi, R >$, where S is a finite set of states $\{s_1, \ldots, s_n\}$. A is a finite set of actions for all states. $\Phi : A \times S \times S$ is the state transition function specified as a probability distribution. The probability of reaching state s' by performing action a in state s is written as $\Phi(a, s, s')$. $R : S \times A \rightarrow \Re$ is the reward function. $R(s, a)$ is the reward that the agent receives if it takes action a in state s. A policy for an MDP is a mapping $\pi : S \rightarrow A$ that selects an action for each state. A solution to an MDP is a policy that maximizes its expected value. For the discounted infinite–horizon case with any given discount factor $\gamma \in [0, 1)$, there is a policy V^* that is optimal regardless of the starting state that satisfies the *Bellman* equation (Bellman, 1957):

$$V^*(s) = max_a\{R(s, a)+\gamma\Sigma_{s' \in S}\Phi(a, s, s')V^*(s')\}$$

Two popular methods for solving this equation and finding an optimal policy for an MDP are: (a) value iteration and (b) policy iteration (Puterman, 1994).

2.1 Factored MDPs

A common problem with the MDP formalism is that the state space grows exponentially with the number of domain variables, and its inference methods grow in the number of actions. Thus, in large problems, MDPs becomes impractical and inefficient. Factored representations avoid enumerating the problem state space, producing a more concise representation that makes solving more complex problems tractable.

In a factored MDP, the set of states is described via a set of random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, where each X_i takes on values in some finite domain $Dom(X_i)$. A state $\mathbf{x}$ defines a value $x_i \in Dom(X_i)$ for each variable X_i. Thus, the set of states $S = Dom(X_i)$ is exponentially large, making it impractical to represent the transition model explicitly as matrices. Fortunately, the framework of dynamic Bayesian networks (DBN) (Dean and Kanazawa, 1989) gives us the tools to describe the transition model function concisely. In these representations, the post-action nodes (at the time $t + 1$) contain matrices with the probabilities of their values given their parents' values under the effects of an action.

3 Qualitative MDPs

3.1 Qualitative states

We define a qualitative state space Q as a set of states $q_1, q_2, ..q_n$ that have different utility properties. These properties map the state space into a set of *partitions*, such that each partition corresponds to a group of continuous states with a similar reward value. In a qualitative MDP, a state partition q_i is a region bounded by a set of constraints over the continuous dimensions in the state space. The relational operators used in this approach are $<$ and $\geq$. For example, assuming that the immediate reward is a function of the linear position in a robot navigation domain, a qualitative state could be a region in an $x_0 - x_1$ coordinates system bounded by the constraints: $x_0 \geq val(x_0)$ and $x_1 \geq val(x_1)$, expressing that the current x_0 coordinate is limited by the interval $[val(x_0), \infty]$, and the x_1 coordinate by the interval $[val(x_1), \infty]$. It is evident that a qualitative state can cover a large number of states (if we consider a fine discretization) with similar properties.

Similarly to the reward function in a factored MDP, the state space Q is represented by a decision tree (Q–tree). In our approach, the decision tree is automatically induced from data. Each leaf in the induced decision tree is labeled with a new qualitative state. Even for leaves with the

same reward value, we assign a different qualitative state value. This generates more states but at the same time creates more guidance that helps produce more adequate policies. States with similar reward are partitioned so each q–state is a continuous region. Figure 1 shows this tree transformation in a two dimensional domain.

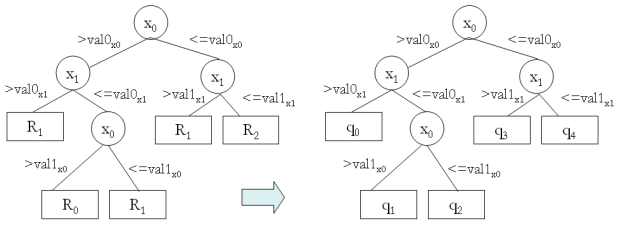

Figure 1: Transformation of the reward decision tree into a Q-tree. Nodes in the tree represent continuous variables and edges evaluate whether this variable is less or greater than a particular bound.

Each branch in the Q–tree denotes a set of constraints for each partition q_i. Figure 2 illustrates the constraints associated to the example presented above, and its representation in a 2-dimensional space.

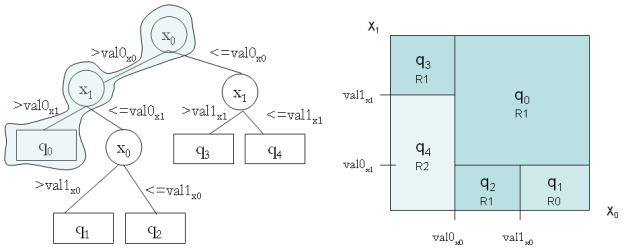

Figure 2: In a Q-tree, branches are constraints and leaves are qualitative states. A graphical representation of the tree is also shown. Note that when an upper or lower variable bound is infinite, it must be understood as the upper or lower variable bound in the domain.

3.2 Qualitative MDP Model Specification

We can define a qualitative MDP as a factored MDP with a set of hybrid qualitative–discrete factors. The qualitative state space Q, is an additional factor that concentrates all the continuous variables. The idea is to substitute all these variables by this abstraction to reduce the dimensionality of the state space. Initially, only the continuous variables involved in the reward function are considered, but, as described in Section 4, other continuous variables can be incorporated in the refinement stage. Thus, a Qualitative MDP state is described in a factored form as $\mathbf{X} = \{X_1, \ldots, X_n, Q\}$, where $X_1, \ldots, X_n$ are the discrete factors, and Q is a factor that represents the relevant continuous dimensions.

3.3 Learning Qualitative MDPs

The Qualitative MDP model is learned from data based on a random exploration of the environment with a 10% Gaussian noise introduced on the actions outcomes. We assume that the agent can explore the state space, and for each state–action can receive some immediate reward. Based on this random exploration, an initial partition, Q_0, of the continuous dimensions is obtained, and the reward function and transition functions are induced.

Given a set of state transition represented as a set of random variables, $O^j = \{\mathbf{X_t}, \mathbf{A}, \mathbf{X_{t+1}}\}$, for $j = 1, 2, ..., M$, for each state and action A executed by an agent, and a reward (or cost) R^j associated to each transition, we learn a qualitative factored MDP model:

1. From a set of examples $\{O, R\}$ obtain a decision tree, RDT, that predicts the reward function R in terms of continuous and discrete state variables, $X_1, \ldots, X_k, Q$. We used J48, a Java re- implementation of C4.5 (Quinlan, 1993) in Weka, to induce a pruned decision tree.

2. Obtain from the decision tree, RDT, the set of constraints for the continuous variables relevant to determine the qualitative states (q–states) in the form of a Q-tree. In terms of the domain variables, we obtain a new variable Q representing the reward-based qualitative state space whose values are the q–states.

3. Qualify data from the original sample in such a way that the new set of attributes are the Q variables, the remaining discrete state variables not included in the decision tree, and the action A. This transformed data set can be called the qualified data set.

4. Format the qualified data set in such a way that the attributes follow a temporal causal ordering. For example variable Q_t must be set before Q_{t+1}, $X1_t$ before $X1_{t+1}$, and so on. The whole set of attributes should be the variable Q in time t, the remaining system variables in time t, the variable Q in time t+1, the remaining system variables in time t+1, and the action A.

5. Prepare data for the induction of a 2-stage dynamic Bayesian net. According to the action space dimension, split the qualified data set into $|A|$ sets of samples for each action.

6. Induce the transition model for each action, A_j, using the K2 algorithm (Cooper and Herskovits, 1992).

This initial model represents a high-level abstraction of the continuous state space and can be solved using a standard solution technique, such as value iteration, to obtain the optimal policy. This approach has been successfully applied in some domains. However, in some cases, our abstraction can miss some relevant details of the domain and consequently produce suboptimal policies. We improve this initial partition through a refinement stage described in the next section.

4 Qualitative State Refinement

We have designed a value-based algorithm that recursively selects and partitions abtract states with high utility variance. Given an initial partition and a solution for the qualitative MDP, the algorithm proceeds as follows:

While there is an unmarked partition greater than the minimum size:

1. Select an unmarked partition (state) with the highest variance in its utility value with respect to its neighbours

2. Select the dimension (state variable) with the highest difference in utility value with its contiguous states

3. Bisect the dimension (divide the state in two equal-size parts)

4. Solve the new MDP

5. If the new MDP has the same policy as before, mark the original state before the partition and return to the previous MDP, otherwise, accept the refinement and continue.

The minimum size of a state is defined by the user and is domain dependent. For example, in the robot navigation domain, the minimum size depends on the smallest goal (area with certain reward) or on the robot step size. A graphical representation of this process is shown in figure 3. Figure 3 (a) shows an initial partition for two qualitative variables where each qualitative state is a set of ground states with similar reward. Figure 3 (b) shows the refined two-dimension state space after applying the splitting state algorithm.

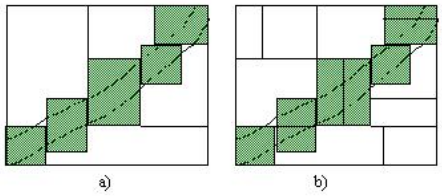

Figure 3: Qualitative refinement for a two-dimension state space. a) initial partition by reward. b) refined state space

5 Experimental Results

We tested our approach in a robot navigation domain using a simulated environment. In this setting goals are represented as light-colored square regions with positive immediate reward, while non-desirable regions are represented by

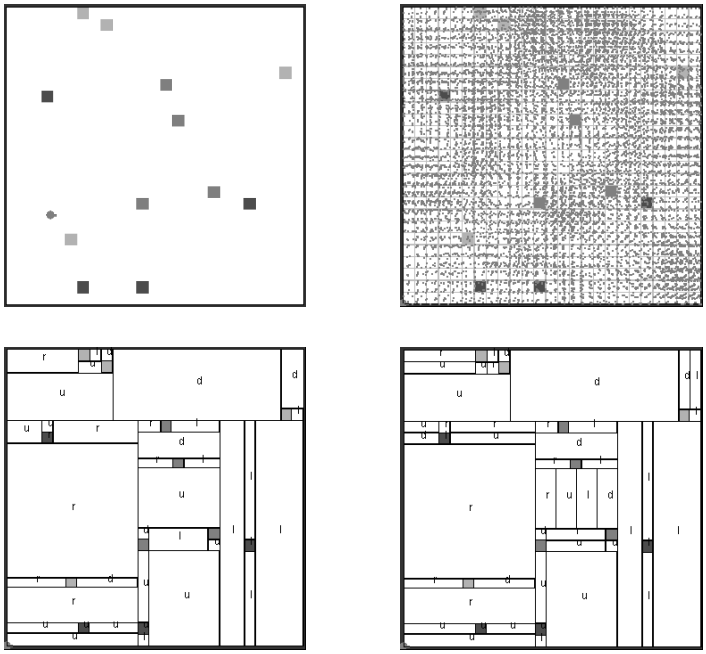

Figure 4: Abstraction and refinement process. Upper left: reward regions. Upper right: exploration process. Lower left: initial qualitative states and their corresponding policies, where u=up, d=down, r=right, and l=left. Lower right: refined partition.

dark-colored squares with negative reward. The remaining regions in the navigation area receive 0 reward (white). Experimentally, we express the size of a rewarded (non zero reward) as a function of the navigation area. Rewarded regions are multivalued and can be distributed randomly over the navigation area. The number of rewarded squares is also variable. Since obstacles are not considered robot states, they are not included.

The robot sensor system included the x-y position, angular orientation, and navigation bounds detection. In a set of experiments the possible actions are discrete orthogonal movements to the right, left, up, and down. Figure 4 upper left shows an example of a navigation problem with 26 rewarded regions. The reward function can have six possible values. In this example, goals are represented as different-scale light colors. Similarly, negative rewards are represented with different-scale dark colors. The planning problem is to automatically obtain an optimal policy for the robot to achieve its goals avoiding negative rewarded regions.

The qualitative representation and refinement were tested with several problems of different sizes and complexities, and compared to a fine discretization of the environment in terms of precision and complexity. The precision is evaluated by comparing the policies and values per state. The *policy precision* is obtained by comparing the policies generated with respect to the policy obtained from a fine discretization. That is, we count the number of *fine* cells in which the policies are the same:

$$PP = (NEC/NTC) \times 100,$$

where PP is the policy precision in percentage, NEC is the number of fine cells with the same policy, and NTC is the total number of fine cells. This measure is pessimistic because in some states it is possible for more than one action to have the same or similar value, and in this measure only one is considered correct.

The utility error is calculated as follows. The utility values of all the states in each representation is first normalized. The sum of the absolute differences of the utility values of the corre-

Table 1: Description of problems and comparison between a "normal" discretization and our qualitative discretization.

Problem					Discrete				Qualitative			
					Learning		Inference		Learning		Inference	
id	no. reward cells	reward size (% dim)	no. reward values	no. samples	no. states	time (ms)	no. iterations	time (ms)	no. states	time (ms)	no. iterations	time (ms)
1	2	20	3	40,000	25	7,671	120	20	8	2,634	120	20
2	4	20	5	40,000	25	1,763	123	20	13	2,423	122	20
3	10	10	3	40,000	100	4,026	120	80	26	2,503	120	20
4	6	5	3	40,000	400	5,418	120	1,602	24	4,527	120	40
5	10	5	5	28,868	400	3,595	128	2,774	29	2,203	127	60
6	12	5	4	29,250	400	7,351	124	7,921	46	2,163	124	30
7	14	3.3	9	50,000	900	9,223	117	16,784	60	4,296	117	241

sponding states is evaluated and averaged over all the differences.

Figure 4 shows the abstraction and refinement process for the motion planning problem presented above. A color inversion and gray scale format is used for clarification. The upper left figure shows the rewarded regions. The upper right figure illustrates the exploration process. The lower left figure shows the initial qualitative states and their corresponding policies. The lower right figure shows the refined partition.

Table 1 presents a comparison between the behavior of seven problems solved with a simple discretization approach and our qualitative approach. Problems are identified with a number as shown in the first column. The first five columns describe the characteristics of each problem. For example, problem 1 (first row) has 2 reward cells with values different from zero that occupy 20% of the number of cells, the different number of reward values is 3 (e.g., -10, 0 and 10) and we generated 40,000 samples to build the MDP model.

Table 2 presents a comparison between the qualitative and the refined representation. The first three columns describe the characteristics of the qualitative model and the following describe the characteristics with our refinement process. They are compared in terms of utility error in %, the policy precision also in %, and the time spent in the refinement in minutes.

As can be seen from Table 1, there is a significant reduction in the complexity of the problems using our abstraction approach. This can be clearly appreciated from the number of states and processing time required to solve the problems. This is important since in complex domains where it can be difficult to define an adequate abstraction or solve the resulting MDP problem, one option is to create abstractions and hope for suboptimal policies. To evaluate the quality of the results Table 2 shows that the proposed abstraction produces on average only 9.17% error in the utility value when compared against the values obtained from the dicretized problem as can be seen in Table 2. Finally, since an initial refinement can miss some relevant aspects of the domain, a refinement process may be necessary to obtain more adequate policies. Our refinement process is able to maintain or improve the utility values in all cases. The percentage of policy precision with respect to the initial qualitative model can sometimes decrease with the refinement process. This is due to our pessimistic measure.

Table 2: Comparative results between the initial abstraction and the proposed refinement process.

	Qualitative		Refinement		
id	util. error (%)	policy precis. (%)	util. error (%)	policy precis. (%)	refin. time (min)
1	7.38	80	7.38	80	0.33
2	9.03	64	9.03	64	0.247
3	10.68	64	9.45	74	6.3
4	12.65	52	8.82	54.5	6.13
5	7.13	35	5.79	36	1.23
6	11.56	47.2	11.32	46.72	10.31
7	5.78	44.78	5.45	43.89	23.34

6 Conclusions and Future Work

In this paper, a new approach for solving continuous and hybrid infinite-horizon MDPs is described. In the first phase we use an exploration strategy of the environment and a machine learning approach to induce an initial state abstraction. We then follow a refinement process to improve on the utility value. Our approach creates significant reductions in space and time allowing to solve quickly relatively large problems. The utility values on our abstracted representation are reasonably close (less than 10%) to those obtained using a fine discretization of the domain. A new refinement process to improve the results of the initial proposed abstraction is also presented. It always improves or at least maintains the utility values obtained from the qualitative abstraction. Although tested on small solvable problems for comparison purposes, the approach can be applied to more complex domains where a simple dicretization approach is not feasible.

As future research work we will like to improve our refinement strategy to select a better segmentation of the abstract states and use an alternative search strategy. We also plan to test our approach in other domains.

Acknowledgments

This work was supported in part by the *Instituto de Investigaciones Eléctricas*, México and CONACYT Project No. 47968.

References

R.E. Bellman. 1957. *Dynamic Programming.* Princeton U. Press, Princeton, N.J.

D. P. Bertsekas and J.N. Tsitsiklis. 1996. *Neuro-dynamic programming.* Athena Sciences.

D. P. Bertsekas. 1995. A counter-example to temporal difference learning. *Neural Computation*, 7:270–279.

B. Bonet and J. Pearl. 2002. Qualitative mdps and pomdps: An order-of-magnitude approach. In *Proceedings of the 18th Conf. on Uncertainty in AI, UAI-02*, pages 61–68, Edmonton, Canada.

C. Boutilier, T. Dean, and S. Hanks. 1999. Decision-theoretic planning: structural assumptions and computational leverage. *Journal of AI Research*, 11:1–94.

G. F. Cooper and E. Herskovits. 1992. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning.*

T. Dean and R. Givan. 1997. Model minimization in Markov Decision Processes. In *Proc. of the 14th National Conf. on AI*, pages 106–111. AAAI.

T. Dean and K. Kanazawa. 1989. A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150.

Z. Feng, R. Dearden, N. Meuleau, and R. Washington. 2004. Dynamic programming for structured continuous Markov decision problems. In *Proc. of the 20th Conf. on Uncertainty in AI (UAI-2004). Banff, Canada.*

C. Guestrin, M. Hauskrecht, and B. Kveton. 2004. Solving factored mdps with continuous and discrete variables. In *Twentieth Conference on Uncertainty in Artificial Intelligence (UAI 2004)*, Banff, Canada.

M. Hauskrecht and B. Kveton. 2003. Linear program approximation for factored continuous-state Markov decision processes. In *Advances in Neural Information Processing Systems NIPS(03)*, pages 895– 902.

B. Kuipers. 1986. Qualitative simulation. *AI*, 29:289–338.

M.L. Puterman. 1994. *Markov Decision Processes.* Wiley, New York.

J.R. Quinlan. 1993. *C4.5: Programs for machine learning.* Morgan Kaufmann, San Francisco, Calif., USA.