Automated Explanations for MDP Policies

Omar Zia Khan, Pascal Poupart and James P. Black
David R. Cheriton School of Computer Science
University of Waterloo
200 University Avenue West, Waterloo, ON, N2L 3G1, Canada
{ozkhan, ppoupart, jpblack}@cs.uwaterloo.ca

Abstract

Explaining policies of Markov Decision Processes (MDPs) is complicated due to their probabilistic and sequential nature. We present a technique to explain policies for factored MDP by populating a set of domain-independent templates. We also present a mechanism to determine a minimal set of templates that, viewed together, completely justify the policy. We demonstrate our technique using the problems of advising undergraduate students in their course selection and evaluate it through a user study.

1 Introduction

Sequential decision making is a notoriously difficult problem especially when there is uncertainty in the effects of the actions and the objectives are complex. MDPs [10] provide a principled approach for automated planning under uncertainty. State-of-the-art techniques provide scalable algorithms for MDPs [9], but the bottleneck is gaining user acceptance as it is harder to understand why certain actions are recommended. Explanations can enhance the user's understanding of these plans (when the policy is to be used by humans like in recommender systems) and help MDP designers to debug them (even when the policy is to be used by machines, like in robotics). Our explanations highlight key factors through a set of explanation templates. The set of templates are sufficient, such that they justify the recommended action, yet also minimal, such that the size of the set cannot be smaller. We demonstrate our technique through a course-advising MDP and evaluate our explanations through a user study. A more detailed description of our work can be found in [6].

2 Background

A Markov decision process (MDP) is defined by a set S of states s, a set A of actions a, a transition model (the probability $Pr\left(s'|s,a\right)$ of an action a in state s leading to state s'), a reward model (the utility/reward $R\left(s,a\right)$ for executing action a in state s), and a discount factor $\gamma \in [0,1)$. Factored MDPs [1] are typically used for MDPs with large state space where states are determined by values of some variables. A scenario sc is defined as the set of states obtained by assigning values to a subset of state variables. A policy $\pi : S \rightarrow A$ is a mapping from states to actions. The value $V^{\pi}\left(s\right)$ of a policy π when starting in state s is the sum of the expected discounted rewards earned by executing policy π. A policy can be evaluated by using Bellman's equation $V^{\pi}\left(s\right) = R\left(s,\pi\left(s\right)\right) + \gamma \sum_{s' \in S} Pr\left(s'|s,\pi\left(s\right)\right) \cdot V^{\pi}\left(s'\right)$. We shall use an alternative method to evaluate a policy based on occupancy frequencies. The discounted occupancy frequency (hereafter referred as occupancy frequency) $\lambda^{\pi}_{s_0}\left(s'\right)$ is the expected (discounted) number of times we reach state s' from starting state s_0 by executing policy π. Occupancy frequencies can be computed by solving Eq. 1.

$$\lambda^{\pi}_{s_0}\left(s'\right) = \delta\left(s', s_0\right) + \gamma \sum_{s \in S} Pr\left(s'|s,\pi\left(s\right)\right) \cdot \lambda^{\pi}_{s_0}\left(s\right) \quad \forall s' \tag{1}$$

where $\delta(s', s_0)$ is a Kroenecker delta which assigns 1 when $s' = s_0$ and 0 otherwise. The occupancy frequencies for a scenario (or a set of scenarios), $\lambda^{\pi}_{s_0}(sc)$, is the expected number of times we reach a scenario sc, from starting state s_0, by executing policy π *i.e.*, $\lambda^{\pi}_{s_0}(sc) = \sum_{s \in sc} \lambda^{\pi}_{s_0}(s)$. Let sc_r be a set of scenarios with reward value r. The dot product of occupancy frequencies and rewards gives the value of a policy, as shown in Eq. 2.

$$V^{\pi}(s_0) = \sum_{r} \lambda^{\pi}_{s_0}(sc_r) \cdot r \tag{2}$$

An optimal policy π^* earns the highest value for all states (i.e., $V^{\pi *}(s) \geq V^{\pi}(s)\ \forall \pi, s$).

3 Explanations for MDPs

3.1 Templates for Explanations

Our explanation answers the question, "*Why has this action been recommended?*" by populating generic templates, at run-time, with domain-specific information from the MDP *i.e.*, occupancy frequency of a scenario. The reward function implicitly partitions the state space in regions with equal reward value. These regions can be defined as partial variable assignments corresponding to scenarios or sets of scenarios. An explanation then could be the frequency of reaching a scenario is highest (or lowest). This is especially useful when this scenario also has a relatively high (or low) reward. Below we describe templates in which the underlined phrases (scenarios and their frequencies) are populated at run-time.

- **Template 1:** "*ActionName* is the only action that is likely to take you to $Var_1 = Val_1, Var_2 = Val_2, ...$ about λ times, which is higher (or lower) than any other action"
- **Template 2:** "*ActionName* is likely to take you to $Var_1 = Val_1, Var_2 = Val_2, ...$ about λ times, which is as high (or low) as any other action"
- **Template 3:** "*ActionName* is likely to take you to $Var_1 = Val_1, Var_2 = Val_2, ...$ about λ times"

While these templates provide a method to present explanations, multiple templates can be populated even for non-optimal actions; a non-optimal action can have the highest frequency of reaching a scenario without having the maximum expected utility. Thus, we need to identify a set of templates that justify the optimal action.

3.2 Minimal Sufficient Explanations

We define an explanation as sufficient if it can prove that the recommendation is optimal, *i.e.*, the selected templates show the action is optimal without needing additional templates. A sufficient explanation cannot be generated for a non-optimal action since an explanation for another action (*e.g.*, the optimal action) will have a higher utility. A sufficient explanation is also minimal if it includes the minimum number of templates needed to ensure it is sufficient. The minimality constraint is useful for users and sufficiency constraint is useful for designers.

Let s_0 be the state where we need to explain why $\pi^*(s_0)$ is an optimal action. We can compute the value of the optimal policy $V^{\pi^*}(s_0)$ or the Q-function[1] $Q^{\pi^*}(s_0, a)$ using Eq. 2. Since a template is populated by a frequency and a scenario, the utility of this pair in a template is $\lambda^{\pi^*}_{s_0}(sc_r) \cdot r$. Let E be the set of frequency-scenario pairs that appear in an explanation. If we exclude a pair from the explanation, the utility is $\lambda^{\pi^*}_{s_0}(sc_i) \cdot \bar{r}$, where r_{min} is the minimum value for the reward variable. This definition indicates that the worst is assumed for the scenario in this pair. The utility of an explanation V_E is

[1] In reinforcement learning, the Q-function $Q^{\pi}(s, a)$ denotes the value of executing action a in state s followed by policy π.

$$V_E = \sum_{i \in E} \lambda_{s_0}^{\pi^*}(sc_i) \cdot r_i + \sum_{j \notin E} \lambda_{s_0}^{\pi^*}(sc_j) \cdot r_{min} \tag{3}$$

where the first part includes the utility from all the pairs in the explanation and the second part considers the worst case for all other pairs. For an explanation to be sufficient, its utility has to be higher than the next best action, *i.e.*, $V^{\pi^*} \geq V_E > Q^{\pi^*}(s_0, a) \quad \forall a \neq \pi^*(s_0)$. For it to be minimal, it should use the fewest possible pairs. Let us define the gain of including a pair in an explanation as the difference between the utility of including versus excluding that pair ($\lambda_{s_0}^{\pi^*}(sc_i) \cdot r_i - \lambda_{s_0}^{\pi^*}(sc_i) \cdot r_{min}$). To find a minimal sufficient explanation, we can sort the gains of all pairs in descending order and select the first k pairs that ensure $V_E \geq Q^{\pi^*}(s_0, a)$. This provides our minimal sufficient explanation.

3.3 Workflow and Algorithm

The designer identifies the states and actions, and specifies the transition and reward functions. The optimal policy is computed, using a technique such as value iteration, and is consulted to determine the optimal action. Now an explanation can be requested. The pseudo code for the algorithm to compute a minimal sufficient explanation is shown in Algorithm 1.

Algorithm 1 Computing Minimal Sufficient Explanations

```
1  //Inputs:  Starting State: s0, Optimal Policy: π*
2  //Outputs: Minimal Sufficient Explanation, MSE
3  ComputeMSE(s0,π*)
4   for r in R
5   |   sc[r] = ComputeScenarios(r)
6   |   λ[r] = ComputeOccupancyFrequency(sc[r])
7   |   utilTemplate [sc, π*,s0]= λ[r]*r
8   |   utilNoTemplate[sc, π^a,s0]= λ[r]*r_min
9   |   netUtil[sc,r] = utilTemplate[sc, π*,s0] - utilNoTemplate[sc, π^a,s0]
10  end
11  sortedNet = sortDescending(netUtil)
12  V_E = 0; V_templates = 0; V_noTemplates = sum(utilNoTemplate);
13  k=1; MSE, Pairs={}
14  do
15  |   add sc[r], λ[r] for sortedNet[k] in Pair
16  |   V_templates  +=  utilTemplate[k]
17  |   V_noTemplates  -=  sortedNet[k]
18  |   V_E = V_templates + V_noTemplates
19  |   k++
20  while (V_E < V_next)
21  MSE = generateTemplates(Terms)
22 return MSE
23 //end computeMSE
```

The function `ComputeScenarios` returns the set of scenarios with reward value r which is available in the encoding of the reward function. The function `ComputeOccupancyFrequency` is the most expensive step which corresponds to solging the system of linear system defined in Eq. 1, which has a worst case complexity that is cubic in the size of the state space. However, in practice, the running time can often be sublinear by using variable elimination to exploit conditional independence and algebraic decision diagrams [5] to automatically aggregate states with identical values/frequencies. The function `GenerateTemplates` chooses an applicable template, from the list of templates, in the order of the list, with the last always applicable.

4 Experiments and Evaluation

4.1 Sample Explanations

We ran experiments on course-advising and hand-washing MDPs [6]. We only discuss the course-advising domain here due to space considerations. The transition model was obtained by using historical data collected over several years at the University of Waterloo. The reward function provides rewards for completing different degree requirements. The horizon of this problem is 3 steps,

each step representing one term and the policy emits a pair of courses to take in that term. The problem has 117.4 million states. We precomputed the optimal policy since it does not need to be recomputed for every explanation. We were able to compute explanations in approximately 1 second on a Pentium IV 1.66 GHz laptop with 1GB RAM using Java on Windows XP with the optimal policy and second best action precomputed. A sample explanation is shown below.

- Action *TakeCS343&CS448* is the best action because:-
  - It's likely to take you to $CoursesCompleted = 6$, $TermNumber = Final$ about 0.86 times, which is as high as any other action

4.2 User Study with Students

We conducted a user study to evaluate explanations for course advising. We recruited 37 students and showed 3 different recommendations with explanations for different states. For each explanation, they were asked to rate it on various factors such as comprehension, trust-worthiness and usefulness with partial results shown in Figure 1. 59% (65/111) of the respondents indicated that they were able to understand our explanation without any other information; the rest also wanted to know the occupancy frequencies for some other actions. We can provide this information as it is already computed. 76% (84/111) believed that the explanation provided by our system was accurate, with a few wanting to know our sample size to judge the accuracy. 69% (77/111) indicated that they would require extra information beyond that presented in the explanation. When asked what other type of information is needed, we discovered that they wanted the model to cater to preferences such as student's interest, future career plans, and level of difficulty rather than the explanation being inadequate for our existing model. An important indicator of the usefulness of these explanations is that 71% (79/111) of the students mentioned that the explanation provided them with extra information that helped them in making a decision. Also while some students, 23% (26/111), initially disagreed with the recommendation, in 35% (9/26) of these cases our explanation convinced them to change their mind and agree with the original recommendation. The rest disagreed primarily because they wanted a more elaborate model, so no explanation could have convinced them.

We also asked students if they were provided with our system, in addition to the option of discussing their choices with an undergraduate advisor, would they use it. 86% of them mentioned they would use it from home and 89% mentioned they would use it before meeting with an advisor to examine different options for themselves. These numbers indicate substantial interest in our explanations. The explanations generated by our system are generic, while those provided by the advisors are domain-specific. The user study indicates that these two types of explanations are complementary and students would like to access our explanations in addition to consulting advisors.

5 Relationship to Other Explanations Strategies

Explanations have been considered an essential component of intelligent reasoning systems and various strategies have been devised to generate them. Explanations for expert systems are generally in the form of execution traces, such as in MYCIN [2]. Execution traces indicate the rules used in arriving at a conclusion. There are no specific rules in an MDP and the optimal decision is made by maximizing the expected utility which involves considering all of the transition and reward function. Thus, in our explanation we highlight the more important parts of the transition and reward function. Xplain [12] is an example of an intelligent tutoring system that also provided justifications of its decisions. In addition to the rules used by the expert system, it also needed additional domain knowledge to generate these explanations. Our current approach does not use any additional domain knowledge, however this also means we cannot justify the correctness of the transition or reward function. We can only argue about the optimal action using the specified transition and reward functions. Explanations in single-shot recommender systems [13] and case-based reasoning systems [11] are typically based on identifying similar clusters of users or cases and then demonstrating the similarity of the current choice to a cluster or case. Since MDPs are not based on the principle of recommending actions based on similarity, such an approach to generate explanations would be infeasible. Herlocker et al. [4] presented the idea of highlighting key data leading to a recommendation for explanations in recommender systems. Our approach is also motivated by this idea with the key difference that choices in MDPs also impact future states and actions rather than

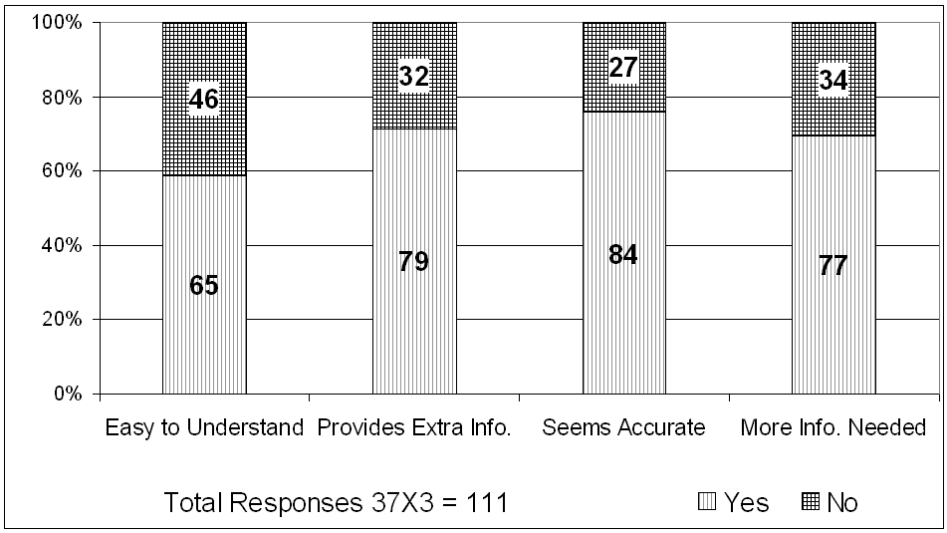

Figure 1: User Perception of MDP-Based Explanations

explaining an isolated decision. McGuinness et al. [8] identify several templates to present explanations in task processing systems based on predefined workflows. Our approach also uses templates, but we cannot use predefined workflows due to the probabilistic nature of MDPs.

Lacave et al. [7] presented several approaches to explain graphical models, including Bayesian networks and influence diagrams. Their explanations require a background in decision analysis and they present utilities of different actions graphically and numerically. We focus on users without any knowledge of utility theory. Elizalde et al. [3] present an approach to generate explanations for an MDP policy that recommends actions for an operator in training. A set of explanations is defined manually by an expert and their algorithm determines a relevant variable to be presented as explanation. Our approach does not restrict to a single relevant variable and considers the long-term effects of the optimal action (beyond one time step). We also use generic, domain-independent templates and provide a technique to determine a minimum set of templates that can completely justify an action.

6 Significance and Implications

While there has been a lot of work on explanations for intelligent systems, such as expert, rule-based, and case-based reasoning systems, there has not been much work for probabilistic and decision-theoretic systems. The main reason behind this discrepancy is the difference in processes through which they arrive at their conclusions. For probabilistic and decision-theoretic systems, there are well-known axioms of probability and theorems from utility theory that are applied to perform inference or compute a policy. Therefore, experts do not need to examine the reasoning trace to determine if the inference or policy computation process is correct. The trace would essentially refer to concepts such as Bayes' theorem, or the principle of maximum expected utility etc, which do not need to be verified. Instead, the input, *i.e.*, transition and reward function, need to be verified. With recent advances in scalability and the subsequent application of MDPs to real-world problems, now explanation capabilities are needed. The explanation should highlight portions of the input that lead to a particular result.

Real-world MDPs are difficult to design because they can involve millions of states. There are no existing tools for experts to examine and/or debug their models. The current design process involves successive iterations of tweaking various parameters to achieve a desirable output. At the end, the experts still cannot verify if the policy indeed accurately reflects their requirements. Our explanations provide hints to experts in debugging by indicating the components of the model that are being utilized in the decision-making process at the current step. This allows experts to verify whether the correct components are being used and focus the tweaking of the model.

Current users have to trust an MDP policy blindly, with no explanations whatsoever regarding the process of computing the recommendation or the confidence of the system in this recommendation. They cannot observe which factors have been considered by the system while making the recommendation. Our explanations can provide users with the information that the MDP is using to base its recommendation. This is especially important if user preferences are not accurately encoded.

If experts or users disagree with the optimal policy, the next step would be to automatically update the model based on interaction, *i.e.*, update the transition and reward functions if the user/expert disagree with the optimal policy despite the explanation. Any such automatic update of the model needs to be preceded by a proper understanding of the existing model, which can only be achieved through explanations, such as those provided by our system.

Just like the optimal policies for MDPs from different domains can be computed using the same underlying techniques, our technique to generate explanations is also generic and can be employed for an MDP from any domain. We have used the same approach described here to generate minimal sufficient explanations for the handwashing MDP [6]. The mechanism to present the explanation to users can then be tailored for various domains. Often a fancier graphical presentation may be more useful than a text-based template. Our focus is to produce generic explanations that can then be transformed for presentation in a user-friendly format.

7 Conclusion

We presented a mechanism to generate explanations for factored MDP in any domain without requiring any additional effort from the MDP designer. We introduced the concept of a minimal sufficient explanation through which an action can be explained using the fewest possible templates. We showed that our explanations can be generated in near-real time and conducted a user study to evaluate their effectiveness. The students appreciated the extra information provided by our generic explanations. Most of the students considered the combination of our explanation with the advisor explanation more effective than either one alone.

In the future, it would be interesting to extend this work to partially observable MDPs. Since the states are not directly observable, it is not obvious how one could generate an explanation that refers to the frequency with which some states are visited. It would also be interesting to extend this work to reinforcement learning problems where the parameters of the model (i.e., transition probabilities and reward function) are unknown or at best partially known. Finally, when an explanation is provided and the user insists that the recommended action is suboptimal, then it would be interesting to close the loop by updating the model to take into account the feedback provided by the user.

References

[1] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1-2):49–107, 2000.

[2] W. J. Clancey. The epistemology of a rule-based expert system – a framework for explanation. *Artificial Intelligence*, 20:215–251, 1983.

[3] F. Elizalde, E. Sucar, A. Reyes, and P. deBuen. An MDP approach for explanation generation. In *AAAI Workshop on Explanation-Aware Computing*, 2007.

[4] J. Herlocker. Explanations in recommender systems. In *CHI' 99 Workshop on Interacting with Recommender Systems*, 1999.

[5] Jesse Hoey, Robert St-aubin, Alan Hu, and Craig Boutilier. SPUDD: Stochastic planning using decision diagrams. In *UAI*, pages 279–288, Stockholm, Sweden, 1999.

[6] Omar Zia Khan, Pascal Poupart, and James P. Black. Minimal sufficient explanations for factored markov decision processes. In *ICAPS*, Thessaloniki, Greece, 2009.

[7] C. Lacave, M. Luque, and F.J. Dez. Explanation of Bayesian networks and influence diagrams in Elvira. *IEEE Transactions on Systems, Man, and Cybernetics*, 37(4):952–965, 2007.

[8] D. McGuinness, A. Glass, M. Wolverton, and P. da Silva. Explaining task processing in cognitive assistants that learn. In *Proceedings of AAAI Spring Symposium on Interaction Challenges for Intelligent Assistants*, 2007.

[9] Warren B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley, 2nd edition, 2011.

[10] Martin Puterman. *Markov Decision Processes*. Wiley, 1994.

[11] Frode Sørmo, Jörg Cassens, and Agnar Aamodt. Explanation in case-based reasoning—perspectives and goals. *Artificial Intelligence Review*, 24(2):109–143, 2005.

[12] W. R. Swartout. Xplain: A system for creating and explaining expert consulting programs. *Artificial Intelligence*, 21:285–325, 1983.

[13] N. Tintarev and J. Masthoff. A survey of explanations in recommender systems. In *ICDE Workshop on Recommender Systems & Intelligent User Interfaces*, 2007.