Minimal Sufficient Explanations for Factored Markov Decision Processes

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. Our explanations can be generated automatically at run-time with no additional effort required from the MDP designer. We demonstrate our technique using the problems of advising undergraduate students in their course selection and assisting people with dementia in completing the task of handwashing. We also evaluate our explanations for course-advising through a user study involving students.

Introduction

Often, a sequence of decisions must be taken by an individual or system. However, deciding on a course of action is notoriously difficult when there is uncertainty in the effects of the actions and the objectives are complex. Markov decision processes (MDPs) (Puterman 1994) provide a principled approach for automated planning under uncertainty. While such an automated approach harnesses the computational power of machines to optimize difficult sequential decision making tasks, the users no longer understand why certain actions are recommended. This lack of understanding is a serious bottleneck that is holding back the widespread use of MDPs as planning tools. Hence, there is a need for explanations that enhance the user's understanding and trust of these plans and help MDP designers to debug and validate their systems.

In MDPs, actions are selected according to the principle of maximum expected utility. Hence, explaining a decision amounts to explaining why the chosen action has highest expected utility. The expected utility depends on the probability of an event occurring and the utility of that event. Since MDPs deal with sequential planning, thus it is not intuitive to estimate the probability of an event occurring, possibly a few steps later. Further, in most cases the utility reflects the preference amongst difference states, rather than being a tangible entity such as money or time. To address these issues, we generate simple and easy-to-understand explanations that provide insight into the expected utility computation by exposing some key pieces of information. More specifically, we highlight the frequency of certain events that are more critical to the computation of the maximum expected utility through explanation templates. We do not concern ourselves with natural language generation and use templates to present explanations. We show how to populate these templates at run-time, using only the information in the MDP specification, through two different domains.

For course-selection advising, an MDP recommends courses to students, based on their previous performance. An explanation indicates why the chosen set of courses are the best option. We conduct a user study involving students to determine the effectiveness of our explanations. We also generate explanations for an MDP that assists people with dementia in handwashing. These explanations are geared towards the caregivers of these people who can evaluate whether the system is functioning properly.

Background and Related Work

Background

A Markov decision Process (MDP) is defined by a set S of states s, a set A of actions a, a transition model, a reward model, a discount factor γ, and a horizon h. We assume A and S are finite and variable names are meaningful and related to the concepts they represent. We focus on factored MDPs (Boutilier, Dearden, and Goldszmidt 2000) in which a set of state variables define the state space, with the state of the MDP determined by the values of all state variables. The set of states obtained by not assigning values to a subset of state variables is known as a scenario. A transition model $Pr : S' \times S \times A \rightarrow [0, 1]$ specifies the probability $Pr\left(s'|s, a\right)$ of an action a in state s leading to a state s'. A reward model $\rho : S \times A \rightarrow \mathbb{R}$ specifies the utility or reward $\rho\left(s, a\right)$ associated with executing action a in state s. The discount factor, $\gamma \in [0, 1)$, indicates if rewards accumulated earlier are preferred. The horizon h represents the number of future steps to consider while planning.

A policy $\pi : S \rightarrow A$ consists of a mapping from states to actions. The value $V^{\pi}\left(s\right)$ of a policy π when starting in state s can be measured as the sum of the expected discounted rewards earned while executing the policy as shown in Eq. 1.

$$V^{\pi}(s_0) = E\left[\sum_{t=0}^{\infty} \gamma^t \rho(s_t, \pi(s_t)) \middle| \pi, s_0\right] \tag{1}$$

A policy can be evaluated by using Bellman's equation in which the value of a state is measured as in Eq. 2.

$$V^{\pi}(s) = \rho(s, \pi(s)) + \gamma \sum_{s' \in S} Pr(s'|s, \pi(s)) V^{\pi}(s') \tag{2}$$

An alternate method to evaluate a policy involves occupancy frequencies (Poupart 2005). We use this formulation while generating our explanations. The discounted occupancy frequency (hereafter referred as simply occupancy frequency) $\lambda_{s_0}^{\pi}(s')$ is the expected number of times we reach state s' from starting state s_0 by executing policy π. Occupancy frequencies can be computed by solving Eq. 3.

$$\lambda_{s_0}^{\pi}(s') = \delta(s', s_0) + \gamma \sum_{s \in S} Pr(s'|s, \pi(s)) \lambda_{s_0}^{\pi}(s) \quad \forall s' \tag{3}$$

where $\delta(s', s_0)$ is a Kroenecker delta which assigns 1 when $s' = s_0$ and 0 otherwise. We can define occupancy frequencies for scenarios, $\lambda_{s_0}^{\pi}(sc)$, as the expected number of times we reach a scenario sc, from starting state s_0, by executing policy π *i.e.*, $\lambda_{s_0}^{\pi}(sc) = \sum_{s \in sc} \lambda_{s_0}^{\pi}(s)$. The dot product of occupancy frequencies and rewards gives the value of a policy, as shown in Eq. 4.

$$V^{\pi}(s_0) = \sum_{s \in S} \lambda_{s_0}^{\pi}(s) \rho(s, \pi(s)) \tag{4}$$

In certain domains, it will be impossible to revisit a state so the occupancy frequency will lie in $[0, 1]$. But the frequency is not a probability so it can lie in $[0, h]$. For instance, the frequency of a scenario can be greater than 1, even in the course advising domain, but not exceed the horizon.

An optimal policy π^* earns the highest value for all states (i.e., $V^{\pi *}(s) \geq V^{\pi}(s)\ \forall \pi, s$). Optimal policies for MDPs can be computed using techniques such as value iteration in which Bellman's optimality equation (Eq. 5) is treated as an update rule that is applied iteratively.

$$V^{\pi^*}(s) = \max_a \left[\rho(s, a) + \gamma \sum_{s'} Pr(s'|s, a) V^*(s')\right] \tag{5}$$

Related Work

Explanations for intelligent systems such as expert and recommender systems have been studied widely (Tintarev and Masthoff 2007). Xplain (Swartout 1983) was an early example of an intelligent tutoring system that provided justifications of its decisions. In addition to the rules used by the expert system, it also needed additional domain knowledge to generate explanations. Another example is MYCIN (Clancey 1983) which provided execution traces as explanations. This approach is infeasible for MDPs as the computation is too complex to be explained directly. Herlocker et al. (1999) 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 in contrast to recommender systems in which an isolated decision is explained. McGuinness et al. (2007) 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.

Chajewska and Helpburn (1997) presented an approach for explanations in probabilistic systems by representing causality using Bayesian Networks and exploiting different links. Lacave and Díez (2002) surveyed the existing techniques and their limitation for explanation in Bayesian Networks. Lacave et al. (2007) 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. Also it is often infeasible to employ graphical approaches with MDPs due to large state spaces and probabilistic effects in sequential plans. Elizalde et al. (2007) 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; however, they also propose to generate explanations automatically. They present an algorithm that determines a relevant variable. The relevant variable is the variable most affected by the action, selected from those that define the value function, and is reported in a predefined template. Our approach is similar as we also use templates to generate explanations and analyze the effects of the optimal action. However, we do not restrict ourselves to a single relevant variable and consider the long-term effects of the optimal action (beyond one time step). We use generic, domain-independent templates and provide a technique to determine a minimum set of templates that can completely justify an action.

Explanations for MDPs

We want to present an explanation that answers the question, "*Why has this recommendation been made?*" We populate generic templates with domain-specific information from the MDP. The templates are populated at run-time, with a subset of these included in the explanation.

Templates for Explanations

The reward function reflects the preference amongst different states or scenarios. Rewards are generally assigned to states or scenarios that have certain semantic value associated with them. The policy for an MDP is computed by maximizing the sum of expected discounted rewards (Eq. 5). Our explanations indicate how this expectation is being maximized by executing the optimal action. Our approach anticipates the effects of an action and shows the contributions of those effects to the sum of expected rewards. The expected reward is the sum of products of the occupancy frequency of each state/scenario with its reward. An explanation for choosing an action could be that 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 probabilities) are populated at run-time.

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

The frequency λ can be higher than 1 if a state can be revisited. While the frequencies are discounted they still represent an expectation of the number of times a state will be visited. To understand this, consider an alternate yet equivalent representation of the discount factor in which $1-\gamma$ is the termination probability, *i.e.*, the probability of the MDP terminating at each step.

We proposed slightly different versions of the above templates in our earlier work (Khan, Poupart, and Black 2008). While these templates provide a method to present explanations, multiple templates can be populated even for non-optimal actions; a non-optimal action may have the highest frequency of reaching a scenario with a high reward, but it still may not have the maximum expected utility. Thus, we need to identify a set of templates to include in the explanation to justify the optimal action. This is our main contribution in this work.

Minimal Sufficient Explanations (MSE)

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 (*i.e.*, 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 sufficiency constraint is useful for designers of MDPs trying to debug their model. The minimality constraint is useful for users trying to understand the policy with as little information as is necessary. If needed, we can relax the minimality or sufficiency constraints and provide more or less templates depending upon the audience.

Let s_0 be the state where we need to explain why $\pi^*(s_0)$ is an optimal action. Let $\pi^{s_0,a}$ be defined as $\pi^{s_0,a}(s) = \begin{cases} a & \text{if } s = s_0 \\ \pi^*(s) & \text{otherwise} \end{cases}$ Note that $\pi^{s_0,a}$ and π^* are equivalent if $a = \pi^*(s_0)$. We can express the expected utility of executing this policy, as $V^{\pi^{s_0,a}}$. This is equivalent to the action-value function (Sutton and Barto 1998), also known as the Q-function, $Q^{\pi^*}(s_0, a)$. We can compute the value of $V^{\pi^{s_0,a}}$ or $Q^{\pi^*}(s_0, a)$ using Eq. 4. Since a template is populated by a frequency and a state/scenario, let us define a term t which encapsulates this information as $t(s, \pi^*, s_0) = \lambda_{s_0}^{\pi^*}(s)\,\rho(s, \pi^*(s))$. Now V^{π^*} can be computed using Eq. 6.

$$V^{\pi^*} = \sum_{s \in S} t(s, \pi^*, s_0) \tag{6}$$

The MSE comprises a subset of the terms in Eq. 6. For this reason, the expected utility of the terms included in the MSE, V_{MSE}, cannot exceed that of the optimal action, V^{π^*}, but must be higher than that of any other action, *i.e.*, $V^{\pi^*} \geq V_{MSE} > Q^{\pi^*}(s_0, a) \quad \forall a \neq \pi^*(s_0)$. In the worst case, all terms will have to be included in the explanation. To compute the MSE, we arrange all terms in Eq. 6 in descending order, and then select the first k terms of this sequence, necessary to ensure that $V_{MSE} \geq Q^{\pi^*}(s_0, a)$. We can compute V_{MSE} using Eq. 7

$$V_{MSE} = \sum_{i \leq k} t_i + \sum_{i > k} \lambda_{s_0}^{\pi^*}(s_i)\,\overline{r} \tag{7}$$

V_{MSE} comprises two components. First, we include the expected utility from all the terms in the MSE, *i.e.*, $\sum_{i \leq k} t_i$. Second, for every term not included in the MSE, we need to consider its worst case by adding utility computed by using the minimum possible reward, $\overline{r}$, to the MSE. The second component is needed to ensure sufficiency if rewards are negative, and minimality otherwise.

In Eq. 6, the total number of terms will equal the size of the state space. This can be computationally prohibitive for large state spaces. Typically, factored MDPs are used in such cases. Also, the reward function can be defined by a set $\mathcal{R}$ of reward variables R such that the sum of their values r is the reward at any given state (e.g. $\rho(s,a) = \sum_{R \in \mathcal{R}} r_{R,s,a}$). Let $sc_{R=r}$ define the scenario for which reward variable R has value r[1], and $V_f^{\pi^*}$ represent the utility of executing π^* for a factored MDP. In Eq. 7, $\overline{r}$ represents the minimum value for the reward function. With multiple reward variables, every variable may have its own minimum value which can be used instead. Let $\overline{r}_i$ define the minimum value for the reward variable used in term i in the sorted sequence. Now, we can rewrite Eq. 6 and 7 as Eq. 8 and 9 respectively.

$$V_f^{\pi^*} = \sum_{R \in \mathcal{R}} \sum_{r \in dom(R)} \lambda_{s_0}^{\pi^*}(sc_{R=r})\, r \tag{8}$$

$$V_{fMSE} = \sum_{i \leq k} t_i + \sum_{i > k} \lambda_{s_0}^{\pi^*}(sc_i)\,\overline{r}_i \tag{9}$$

[1]For the sake of completeness, we would like to point out that we can also handle the special case where a set of scenarios for reward variable R have value r by computing the occupancy frequency for each scenario independently and then adding them to create a single term for use in Eq. 8.

The number of terms is now significantly lower since we only need a single term per value of each reward variable. This allows us to compute an MSE even for domains with large state spaces.

We know that the optimal policy is invariant to positive linear transformations on the reward function. We also want this property in the MSE to ensure that the MSE only changes if the model has changed. This will assist designers in debugging the model efficiently.

Proposition 1: MSE remains invariant under affine transformations of the reward function.

Proof: Let $\hat{V_f}^{\pi}$ denote the expected utility for any policy π when rewards have been scaled by adding any constant c. If we substitute r by $r + c$ in Eq. 8 we get $\hat{V_f}^{\pi} = V_f^{\pi} + c \sum_{R \in \mathcal{R}} \sum_{r \in dom(R)} \lambda_{s_0}^{\pi}(sc_{R=r})\, r$. Since occupancy frequencies computed for an MDP must add up to the horizon ($\sum_{r \in dom(R)} \lambda^{\pi}(sc_{R=r})\, r = h$), so $\hat{V_f}^{\pi} = V_f^{\pi} + c|\mathcal{R}|h$, where $|\mathcal{R}|$ is the total number of reward variables. Similarly $\hat{V}_{fMSE} = V_{fMSE} + c|\mathcal{R}|h$. Since $V_{fMSE} > V_f^{\pi}$, thus $\hat{V}_{fMSE} > \hat{V_f}^{\pi}$ for any constant c. Also $\hat{V}_{fMSE}$ will comprise the same scaled reward values and frequencies as those in V_{fMSE} otherwise the explanation would not remain either sufficient or minimal. For discounted domains, the frequencies will add up to the expected discounted horizon instead of h. A similar proof can also be presented for the case where rewards are multiplied by a positive constant. ■

Workflow and Algorithm

The basic workflow for our explanation process is as follows. The designer identifies the states and actions, and specifies the transition and reward functions of an MDP. The optimal policy is computed by using a technique such as value iteration. Now the designer/user can consult an optimal policy to determine an optimal action and request an explanation. Our system will compute an MSE using the algorithm given below.

1) Compute $sc_{R=r}$, the scenario which comprises the set of all states that lead to each value r of each reward variable R. This information is directly available from the dependencies encoded in the reward function. For each partial assignment of value r to variable R, note the set of states that receive reward r to compute the scenario.
2) For every scenario $sc_{R=r}$, compute the occupancy frequency $\lambda_{s_0}^{\pi^*}(sc_{R=r})$ for every action using Eq. 3. The occupancy frequency for a scenario is computed efficiently by summing the occupancy frequencies of each state in it using variable elimination. The recurrence is terminated after a number of steps equal to the horizon of the MDP or when convergence is achieved (due to the discount factor) for infinite horizon problems.
3) Compute the term $t(s, \pi^*, s_0)$ and $\lambda_{s_0}^{\pi^*}(sc_i)\,\overline{r}_i$ for every scenario $sc_{R=r}$. They respectively represent the advantage and disadvantage of including and excluding a term from the MSE.
4) Sort $t_i - \lambda_{s_0}^{\pi^*}(sc_i)\,\overline{r}_i$ in descending order and select the first k terms from this sequence to include in the MSE for which $V_{MSE} > Q^{\pi^*}(s_0, a) \quad \forall a \neq \pi^*(s_0)$. Note that t_i and $\lambda_{s_0}^{\pi^*}(sc_i)\,\overline{r}_i$ respectively represent the advantage and disadvantage associated with including and excluding a term from the MSE, so their difference indicates the benefit of this term in the explanation versus excluding it.
5) Present each term in the explanation to the user in one of the defined templates. Choose templates using the following criteria.
    a) Use template 1 if the optimal action has the highest (or lowest) expected frequency to reach that scenario by a significant margin[2].
    b) Use template 2 if the optimal action has the highest (or lowest) expected frequency, but not by a significant margin.
    c) Use template 3 if neither of the previous templates can be used.

The expensive step in the algorithm is Step 2 that involves computing the occupancy frequency using Eq. 3. The rest are basic arithmetic operations and logical comparisons except sorting the sequence of terms in Step 4. Computing occupancy frequencies is similar to policy evaluation whose complexity is that of solving a linear system of equations, which is cubic in the state space. To speed up computation, we compute occupancy frequencies by performing variable elimination using algebraic decision diagrams (ADD). ADDs provide a compact representation that automatically aggregates states with identical values/frequencies, needed for scenarios, thereby significantly reducing the running time. More details on the use of ADDs in MDPs can be found in (Hoey et al. 1999).

Discussion

For any given state, an MSE is guaranteed to exist; in the worst case it will need to include all the terms in the MSE from Eq. 6 or Eq. 8. Thus, the upper bound on the number of templates displayed to the user is also given by this number of terms, which will depend on the structure of the MDP being explained. A relatively large number of terms in the MSE will indicate that the effect of the optimal action is not substantially different from that of at least one other action. We can also argue that for every term there is at least one template that can be used to present the information to the user since Template 3 can always be used. While template 3 may not seem to provide much information in itself, it does indicate that there are better or worse actions available if the scenario being depicted is of particular interest to the user or designer.

Our technique can be used in finite and discounted infinite horizon problems. For discounted MDPs, the frequencies are discounted so if their value appears non-intuitive, the user may have to be explained that a discount factor is being used. If an infinite horizon MDP is only

[2]In our implementation, a significant margin means twice as high as the next highest. It can be adjusted depending on the domain.

Table I
EXPLANATIONS FOR COURSE ADVISING DOMAIN (REWARD VARIABLES=2, VALUES PER VARIABLE=2+2, MAX. TERMS=4)

Terms in MSE	1	2	3–4
Frequency	134	48	0
Mean ±STD of $Q^{\pi^*}(s_0, a')/V^{\pi^*}$	0.46±0.41	0.81±0.24	-

Table II
EXPLANATIONS FOR HANDWASHING DOMAIN (REWARD VARIABLES=3, VALUES PER VARIABLE=2+2+15, MAX. TERMS=19)

Terms in MSE	1	2	3	4	5	6	7–19
Frequency	0	142	94	119	2	25	0
Mean ±STD of $Q^{\pi^*}(s_0, a')/V^{\pi^*}$	-	0.51±0.22	0.62±0.10	0.68±0.04	0.61±0.15	0.69±0.05	-

discounted for computational convenience, the explanation may again appear non-intuitive. As explained, we also cater to the case where multiple goals are expressed through different reward variables.

In model checking, safety and liveness properties are used to debug or validate models. While our approach does not explicitly indicate whether an action is chosen because it leads to a dead-end or is optimal because it is the only action that avoids a dead-end, this information is implicit in the MSE since a dead-end would be a scenario with low reward, and the optimal action would have a low frequency of reaching it.

Experiments and Evaluation

Experimental Results

We ran experiments on two different domains. Our course-advising MDP has 4 core courses and 7 elective courses (from which the student has to choose), with each course having 4 possible letter grades and belonging to a certain area. It has 21 possible actions, with each action representing a pair of elective courses. The objective is to pass 6 elective courses in 3 terms by taking at least one course in 3 different areas. The transition model was obtained by using historical data collected over several years (for 14,000 undergraduate students) at the University of Waterloo. The reward function provides rewards for completing different degree requirements with the reward function decomposed in two different variables, one for each degree requirement, with 2 values per variable. The horizon of this problem is 3 steps, each step representing one term and it is undiscounted. The other domain was the handwashing POMDP developed by Hoey et al. (2007), available online, to assist people with dementia in handwashing. We converted their POMDP into an MDP, assuming all states are observable and changing variable names and values to make them more user-friendly. The horizon for this domain is 100 steps and discount factor is 0.95. There are three different reward variables in the reward function with 19 distinct values for rewards. Explanations for this domain are intended for a caregiver/nurse to evaluate the validity of the prompt, and not for the person washing hands.

We computed MSEs for different starting states in the course advising and handwashing MDPs. The results are shown in Table I and Table II. We can see that the MSE generally contains very few terms for both domains, more evident in the handwashing domain in which there was a total of 19 terms, and only 6 were needed for any given optimal action in 382 different starting states generated randomly. It is natural to expect an explanation to be more complicated if two policies have similar effects. We estimate the complexity of an explanation by the number of terms included in it. Also we define two policies to have similar effects if the ratio of their normalized expected utilities is close to 1. We can see from both tables that if the ratio between the expected utility of the second best policy, $Q^{\pi^*}(s_0, a')$, and optimal policies, V^{π^*}, is high then the explanation includes more terms. On the other hand, if the ratio is low it means that the optimal action is much superior and we can see that fewer terms are needed in the MSE. This result is in line with our intuition.

Two sample explanations, one from each domain, are shown below.

- Action *TakeCS343&CS448* is the best action because:-
  - It is likely to take you to $\underline{CoursesCompleted = 6}$, $\underline{TermNumber = Final}$ about $\underline{0.86}$ times, which is as high as any other action
- Action *DoNothingNow* is the best because:-
  - It is likely to take you to $\underline{handswashed = YES}$, $\underline{planstep = Clean\&Dry}$ about $\underline{0.71}$ times, which is the higher than any other action
  - It is likely to take you to $\underline{prompt = NoPrompt}$ about $\underline{12.71}$ times, which is as high as any other action[3]

The occupancy frequency in the last template is higher than 1 because it is possible visit a state multiple times with a reward for not issuing a prompt every time.

We precomputed the optimal policy for both domains since they do not need to be recomputed for every explanation. We were able to compute explanations for the course advising and handwashing problems in approximately 1 and 4 seconds respectively 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. Note that the course advising problem has 117.4 million

[3]The variable names have been changed from those in (Hoey et al. 2007)

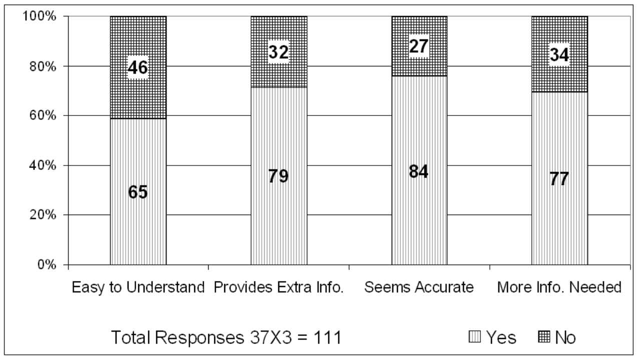

Figure 1. User Perception of MDP-Based Explanations

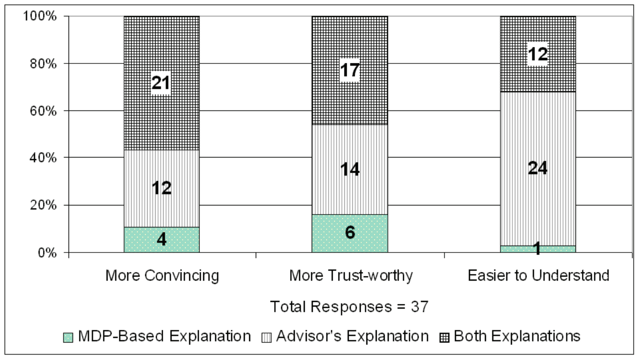

Figure 2. Comparison of MDP-Based and Advisor Explanations

states and the handwashing problem has 207,360 states. The simpler reward function (two reward variables with two values each) in the course advising domain resulted in faster execution despite a larger number of states. We conducted a user study to evaluate explanations for course advising with advisors and students.

Feedback by Advisors

We presented our explanations for several states to undergraduate advisors and sought their feedback. The advisors noted that students not performing well would benefit from our explanations as they would help them focus on requirements they are likely to not fulfill. They also mentioned that grade-conscious students would also appreciate our explanations. The advisors considered the model used by the MDP, *i.e.*, the transition probabilities, as a useful tool to validate or correct their perception about various courses being easy or hard. They were apprehensive that it would be difficult for an online system to replace them as they consider more factors than completing degree requirements, which include preferences such as student's interests, career path, difficulty level of the course, stereotypes about areas, courses or professors. There has been some research on preference elicitation to model student preferences for course advising in MDPs (Royalty et al. 2002), so we explained that it is possible to extend our model to include such factors, but it is outside the scope of our work on explaining MDPs.

User Study with Students

We recruited 37 students from the University of Waterloo's CS department and showed 5 recommendations with explanations for different states. For each explanation, they were asked to rate the explanations on various factors such as comprehension, trust-worthiness and usefulness. For 3 states, the explanation was computed by our technique and for the other 2, the explanations were provided by advisors.

The results regarding the user's perceptions of our explanations are shown in Figure 1. 59% (65/111) of the respondents indicated that they were able to understand our explanation without any other information. Most of the students who wanted more information either wanted to know the occupancy frequencies for some other actions to get a comparison, or knowledge about the technique used to compute the transition probabilities. For the first concern, we can provide this information as it is already computed in Step 2 of algorithm for the MSE. For the second concern, we attribute this curiosity to our audience mostly being students in CS interested in understanding the system.

In 76% (84/111) of the cases, it was believed that the explanation provided by our system was accurate. A few students wanted to know our sample size to judge the accuracy. Quite a few respondents, 69% (77/111), indicated that they would require extra information beyond that presented in the explanation. When asked what other type of information they may be interested in, we discovered that they considered the model inadequate and wanted a richer model that was able to cater to preferences such as student's interest, future career plans, and level of difficulty rather than the explanation being inadequate for the 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. Another indicator of their influence is that while students initially disagreed 23% (26/111) of the times with the recommendation, in 35% (9/26) of these cases our explanation convinced them to change their mind and agree with the original recommendation. In most of the other cases, again the students disagreed because their final decision depended upon some factor, not modeled by our system, so their opinion could not have been changed by any explanation.

To compare our explanations with advisor explanations, we asked students whether they preferred our explanation, the advisor explanation, or having access to both of them simultaneously. These results are shown in Figure 2. 57% (21/37) students found that the most convincing option was to access both explanations, as opposed to 32% in favor of only advisor explanations and 11% in favor of only our explanations. Similarly, 46% (17/37) students considered both explanations viewed together as more trust-worthy as opposed to having only advisor explanations (38%) or only our explanations (16%). As expected, most of the students (65% or 24/37) found it easier to understand advisor explanations as they were more personal and humanoriented. This is understandable since the advisors employ domain-specific constructs in their explanations while our explanations are totally generic. However, a few students (32%) also indicated that having a combination of the two would be easier to understand. Finally, we also asked students if they were provided access to our system over the Internet, in addition to the option of discussing their choices with an undergraduate advisor, would they use our system. 86% of them mentioned they would use it from home while trying to determine their choice of courses, 89% mentioned they would use it before meeting with an advisor to examine different options for themselves, and 70% mentioned they would use it after meeting with advisors to arrive at a final decision. Among the 30%, who indicated they would not use it after meeting advisors, many expected the advisors to incorporate the information from our explanations in their advice and thus considered it redundant to check it themselves. In any case, these numbers are very encouraging as they show substantial interest in our explanations.

The explanations generated by our system are generic, while those provided by the advisors are domain-specific. Our results show that these two types of explanations are complementary and students would like to access our explanations in addition to consulting advisors. A recurring theme during the user study was students inquiring about a facility to compare different choices, *i.e.*, asking the question "*Why is action a better than action b*?", especially if their preferred action was different from the recommended action. We have extended our system to answer such questions by presenting the MSE for the recommended action π^* and then populating the templates for the same terms for their preferred action *b*. The comparison demonstrates how *a* is better than *b* as $V_{MSE} \geq Q^{\pi^*}(s_0, b)$.

Conclusion and Future Work

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 terms. We showed that our explanations can be generated in near-real time for different domains. We also conducted a user study to evaluate the effectiveness of our explanations for course advising. The users appreciated the extra information provided by our generic explanation but also required some domain-specific information. Most of the students considered the combination of our explanation with the advisor explanation more effective than either one alone.

Injecting domain-specific information in explanations increases user-acceptance, though requires additional effort from the designers. As future work, we intend to develop a generic mechanism to represent domain-specific information so that it may be inserted in our explanations. We plan to extend our work to partially observable MDPs. The complication in such explanations is to cater for the observation function rather than having a single known current state for which the optimal policy has been computed and is being executed.

References

Boutilier, C.; Dearden, R.; and Goldszmidt, M. 2000. Stochastic dynamic programming with factored representations. *Artificial Intelligence* 121(1-2):49–107.

Chajewska, U., and Halpern, J. 1997. Defining explanation in probabilistic systems. In *UAI*.

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

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

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

Hoey, J.; St-aubin, R.; Hu, A.; and Boutilier, C. 1999. SPUDD: Stochastic planning using decision diagrams. In *UAI*, 279–288.

Hoey, J.; von Bertoldi, A.; Poupart, P.; and Mihailidis, A. 2007. Assisting persons with dementia during handwashing using a partially observable Markov decision process. In *ICVS*.

Khan, O. Z.; Poupart, P.; and Black, J. P. 2008. Explaining recommendations generated by MDPs. In *ECAI-2008 Workshop on Explanation Aware Computing (ExaCt)*.

Lacave, C., and Díez, F. J. 2002. A review of explanation methods for Bayesian networks. *The Knowledge Engineering Review* 17:107–127.

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

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

Poupart, P. 2005. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes*. Ph.D. Dissertation, University of Toronto.

Puterman, M. 1994. *Markov Decision Processes*. Wiley.

Royalty, J.; Holland, R.; Dekhtyar, A.; and Goldsmith, J. 2002. POET, the online preference elicitation tool. In *AAAI Workshop on Preferences in AI and CP: Symbolic Approaches*.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

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

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