Explaining Recommendations Generated by MDPs

Omar Zia Khan, Pascal Poupart, and James P. Black

David R. Cheriton School of Computer Science, University of Waterloo,
200 University Ave. W, Waterloo, ON, Canada N2L 3G1
{ozkhan,ppoupart,jpblack}@uwaterloo.ca

Abstract. There has been little work in explaining recommendations generated by Markov Decision Processes (MDPs). We analyze the difficulty of explaining policies computed automatically and identify a set of templates that can be used to generate explanations automatically at run-time. These templates are domain-independent and can be used in any application of an MDP. We show that no additional effort is required from the MDP designer for producing such explanations. We use the problem of advising undergraduate students in their course selection to explain the recommendation for selecting specific courses to students. We also propose an extension to leverage domain-specific constructs using ontologies so that explanations can be made more user-friendly.

1 Introduction

In many situations, a sequence of decisions must be taken by an individual or system (e.g., course selection by students, inventory management for a factory, etc.). 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) [1] provide a principled approach for automated planning under uncertainty. While the beauty of an automated approach is that the computational power of machines can be harnessed to optimize difficult sequential decision making tasks, the drawback is that users no longer understand why certain actions are recommended. This lack of understanding is a serious bottleneck that is currently holding back the widespread use of MDPs in recommender systems. Hence, there is a need for explanations that enhance the user's understanding and trust of these recommendations.

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. However, this is complicated by the fact that the numerical value of utility is not meaningful in most cases, and the computation of the expected utility is usually too complex to be explained easily. To address this, we generate simple and easy-to-understand explanations that provide some insight into the expected utility computation by exposing some of the important factors. More specifically, we are interested in answering two types of questions: why was a particular action recommended, or why was it not

recommended? We do not concern ourselves with the natural language aspects of explanation generation. Instead, we devise explanation templates to answer these questions. We also demonstrate how to populate these templates at runtime using only the information in the MDP and without any extra effort on the part of the developer of the system. In certain cases, the explanations can be more meaningful if additional domain knowledge is available. We use an ontology to represent domain-specific facts and show how to use this information to enrich the explanations. Finally, we demonstrate our approach in the domain of course-selection advising, where an MDP recommends elective courses to upper-level undergraduates, based on their previous performance.

2 Background and Related Work

A Markov decision Process (MDP) is formally defined by a set of states S and a set of actions A, a transition model and a reward model. (A and S are assumed to be finite throughout the paper.) A set of state variables defines the state space, with the current state of the MDP determined by the current values of all state variables. We assume that the variable names are meaningful and related to the concepts being represented. A transition model $Pr : S' \times S \times A \rightarrow [0, 1]$ specifies the probability $Pr(s'|s, a)$ of an action a in state s leading to a state s'. A reward model $R : S \times A \rightarrow \mathbb{R}$ specifies the utility or reward $R(s, a)$ associated with executing action a in state s. A policy $\pi : S \rightarrow A$ consists of a mapping from states to actions. The value $V^{\pi}(s)$ of a policy π when starting in state s is measured by the sum of the expected discounted rewards earned while executing the policy, with γ representing the discount factor.

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

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 (Equation 2) is treated as an update rule that is applied iteratively. Essentially, the utility of a state is determined by adding the immediate reward and the expected discounted utility of the next state, determined by choosing the optimal action.

$$V^*(s) = \max_a \left[R(s, a) + \gamma \sum_{s'} Pr(s'|s, a)\, V^*(s')\right] \tag{2}$$

Thus, we can always explain that the recommended action maximizes expected utility, although this may not provide much information to the user.

2.1 Course Selection Advising

Throughout the paper, we use the domain of course advising to illustrate our approach for explanation generation. The model for course advising is a factored

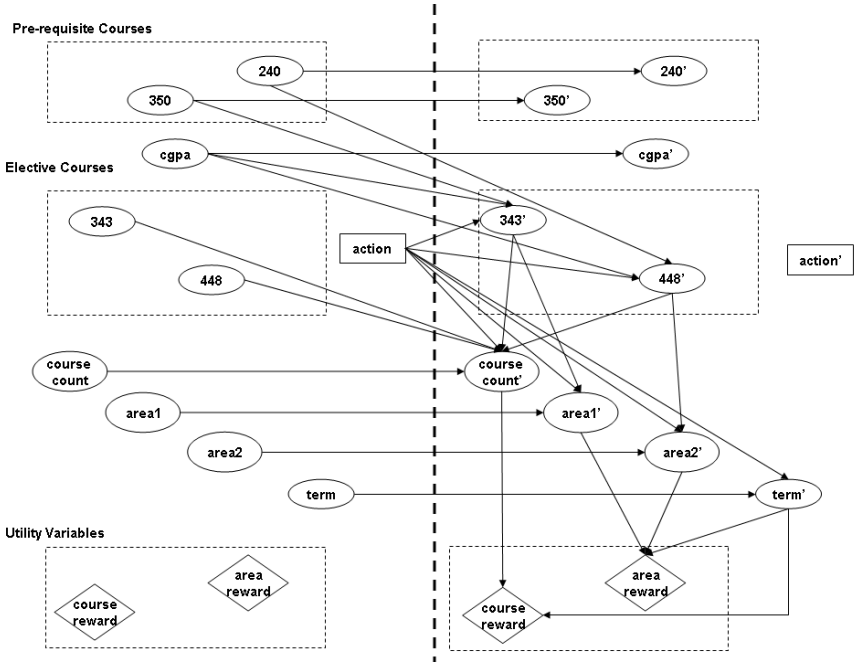

Fig. 1. Dynamic Decision Network encoding of the MDP

MDP that recommends elective courses to upper year students, based on their previous performance. The Dynamic Decision Network (DDN) associated with our MDP model is depicted in Figure 1. We omit the complete list of pre-requisite and elective courses for clarity.

In our model, students must choose two elective courses in each of their last three terms, subject to fulfilling pre-requisite and course area constraints. Each course belongs to one of three areas and has a grade variable taking one of four values $\{G, P, F, N\}$ corresponding to Good, Pass, Fail, and Not Taken. The default value is N. We also have a variable for the CGPA (cumulative grade point average) with two possible values $\{G, P\}$ that correspond to Good and Pass. In addition, book-keeping variables are used to indicate the total number of courses completed, whether each area has been covered, and the number of terms remaining. A state is defined by a particular instantiation of all these variables. Each term corresponds to a time step and the values of the variables may change with each term, giving rise to a different state.

The actions consist of all pairs of courses. The MDP policy depends on the values of all variables mentioned above, and recommends two courses to be taken in the next term.

The transition function was constructed manually. (We will eventually use historical data to estimate it.) It indicates a distribution over values for each variable given the current state. For course grades, the distribution depends on

the CGPA and grades of the pre-requisites. If the pre-requisites for a course are not completed, actions involving that course cannot be taken. Note that the concept of pre-requisites is not explicitly present, however, the conditional probability tables have been encoded to have that effect (i.e., course grade remains N). When a course is passed, the number of completed courses is incremented and whenever an area is covered the value changes accordingly. Finally, after each action, the number of terms passed is incremented.

The reward function is decomposed additively into two components based on the degree requirements. We create two utility variables, *course_reward* and *area_reward*, that have high utility values for states in which 6 courses are passed and three areas are fulfilled. The rewards are only awarded at the end of the third term to avoid multiple accrual if a requirement is completed before the end of the last term. The objective is to maximize the values of these utility variables. Since we are solving a finite-horizon problem with three time steps, we do not discount future rewards.

2.2 Related Work

Explanations for recommender systems have been studied widely [2]. Herlocker [3] described a three-stage process for explanation in recommender systems. First, users can be shown the key data that led to the recommendation, second, a higher-level description of the mathematical process can be provided, and third, a set of claims that can lead towards the conclusion can be presented. Our approach for explanations in MDPs is also motivated by this process with the key difference that choices in MDPs also depend on future states and actions in contrast to recommender systems in which a single decision needs to be explained. Explanations for task processing systems is another related area where the choices also depend on possible future actions. McGuiness et al. [4] identify several templates to present explanations in task processing systems based on predefined workflows. Our approach also uses templates, but we do not (and cannot) use predefined workflows due to the probabilistic nature of MDPs.

Druzdzel [5] first dispelled the notion that it is impossible to generate explanations for probabilistic systems. Chajewska and Halpern [6] combined two previous approaches [7,8] for explanations in probabilistic systems by representing causality using Bayesian Networks and exploiting different links. Lacave and Díez [9] surveyed the existing techniques and their limitation for explanation in Bayesian Networks. Lacave et al. [10] present several approaches to explain graphical models, including Bayesian networks and influence diagrams. Their explanations are geared to users with a background in decision analysis and they present utilities of different actions graphically and numerically. We focus on users without any knowledge of the underlying model or utilities, so we cannot use this approach. The closest to our work is that of Elizalde et al. [11] who present an approach to generating 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 used

in each explanation. The relevant variable is the variable most affected by the action, selected from those that define the value function. This variable 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 present examples of generic, domain-independent templates, as well as a structured mechanism to use additional domain-specific knowledge.

3 Templates for Explanations and their Relative Ordering

We are primarily interested in answering the following two questions:

1. Why has a specific recommendation been made?
2. Why is another particular action not the recommended action?

The first question focuses on the case where the user is interested in understanding the rationale behind choosing a particular action. In response, we need to show the user with benefits of this action, and if necessary, highlight them in comparison to other actions. The second question is comparative and we need to argue that the action chosen by the policy is better. We define a set of templates for explanations that are populated at run-time.

3.1 Templates for Explanations

The policy for an MDP is computed by maximizing the sum of expected discounted rewards (Equation 2). Explanations for MDPs primarily need to explain how this expectation is being maximized by executing a particular action. Our approach is to anticipate the different effects of an action and show the contribution of those effects to the sum of expected rewards.

We call states with high rewards "preferred" states. For instance, we have a reward associated with completing all course areas, and users (students) understand that this is a good thing. Note that the reward may depend only on a subset of variables, such as the area variables. Hence, there may be a set of preferred states (e.g., all possible combinations of passed courses that satisfy the areas) associated with a reward. We use the term preferred scenario to refer to a set of preferred states. A preferred scenario is defined by assigning values to a subset of the variables in the MDP. The expected reward is the sum of products of the probability of each reachable state with its reward. To explain why an action yields the highest expected utility, it may be sufficient to point out that a preferred scenario is reached with high probability. This is especially convincing when there are no other actions that reach this preferred scenario with high probability.

We can categorize actions as leading to a preferred scenario with a single high reward or multiple high rewards. For instance, one action may lead to

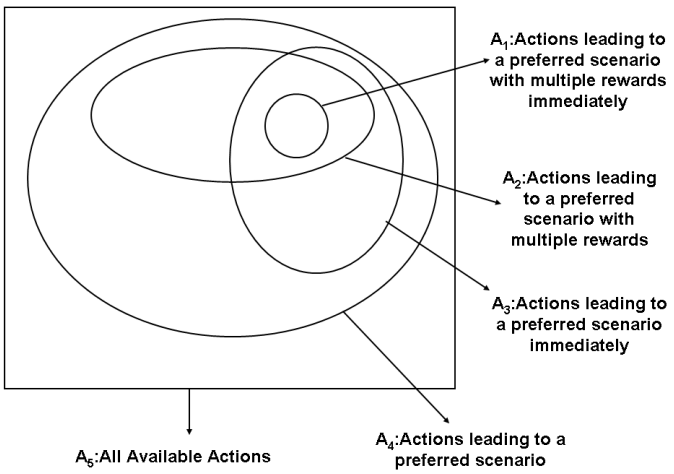

Fig. 2. Different sets of actions for explanations

completing area requirements, but another action may lead to completing both area and course requirements. Furthermore, we can distinguish between actions that are likely to do this immediately and actions that do it only in the future. The actions that lead immediately to a preferred scenario will have a lower discount factor applied to their expected reward. Even though rewards are not discounted in our course advising scenario, we still argue that users are more likely to understand how an action can lead immediately to a preferred state.

We show the relationship among the different sets of actions in Figure 2 using a Venn diagram. The set of available actions, A_5, is defined as actions that can be executed in the current state. In the course advising domain, we exclude from this set all actions with courses whose pre-requisites have not been satisfied. All other sets are subsets of A_5. Below we describe templates in which preferred scenarios and their probabilities are filled in at run-time and these sets are used to select a template.

If there are no other actions in A_5 apart from the chosen action, then we use Template 1. (A_5 should not be empty unless the MDP is ill-formed, which we ignore.) The underlined expressions are replaced by actual values at run-time.

- **Template 1:** "Action *ActionName* is the only action available in state(s) with: $Var_1 = Val_1, Var_2 = Val_2, ...$"

For action sets A_1 to A_4, we define three explanation templates shown below.

- **Template 2:** "Action *ActionName* is the only action that can lead (immediately) to a preferred scenario with: $Var_1 = Val_1, Var_2 = Val_2, ...$ with a high probability (>50%)"

- **Template 3:** "Action *ActionName* leads (immediately) to a preferred scenario with: $Var_1 = Val_1, Var_2 = Val_2, ...$ with highest probability: $P\%$"
- **Template 4:** "Action *ActionName* is one of the actions that can lead (immediately) to a preferred scenario with: $Var_1 = Val_1, Var_2 = Val_2, ...$ with a high probability (>50%)"

The word "immediately" is placed in parentheses as it is only used for explanations for sets A_1 and A_3. For explanations from sets A_1 and A_2, the list of variable assignments will generally be longer, reflecting all the conditions that must be met to earn multiple high rewards.

An action can yield the highest expected reward without necessarily reaching a state of high reward with high probability, so it is possible that none of the proposed templates can be used. We can reduce this possibility by also considering whether an action minimizes the likelihood of entering an undesirable state and using Templates 2, 3 or 4. Undesirable states are those with low rewards, such as failing a course. The process of populating these templates is similar, except that the probability computed is that of not reaching an undesirable state.

Sample explanations for the second question, in which the student asks why a certain action was not taken, can be answered by inverting one of the above explanations. For instance, it could be that it does not maximize the likelihood of reaching a preferred scenario, or it is not a possible action at this stage, or if this action is taken it is likely to move towards an undesirable state with a high probability.

3.2 Ordering of Explanations

It is quite possible that many templates can be populated for a particular recommendation. Thus, it is necessary to select an ordering mechanism. We first consider Template 1, since it gives a simple and sufficient explanation (no other action available) when applicable. Then, we focus on explanations with Templates 2, 3 and 4 based on A_1 to A_4. We first consider whether the chosen action is a member of A_1 since it is a subset of all other action sets and earning multiple high rewards immediately can be more convincing than the explanations with respect to other action sets. If not, we present an explanation from A_2 since earning multiple high rewards at some point in the future should be more convincing than a single reward. Similarly, if this is impossible, we try to present an explanation from A_3 (single immediate high reward) and then A_4 (single future high reward). For each set A_1 to A_4, we first consider Template 2, then Template 3 and finally Template 4.

3.3 Automatic Generation of Explanations

In this section, we describe algorithms to generate explanations automatically, using the above templates. We do not require any extra information apart from what is available in the encoded MDP. However, we do need a mechanism to

compute the sets shown in Figure 2. We also need a technique to populate the various templates once we are given these sets.

Actions unavailable to the system are generally represented by assigning a reward of $-\infty$. We can find all actions that do not lead to a reward of $-\infty$ for the utility variables and add them to A_5.

We can compute a set PS_1 of preferred scenarios ps from the dependencies of the reward variables, by identifying variable assignments with high rewards. Furthermore, by combining preferred scenarios that are consistent, we obtain a new set PS_2 of preferred scenarios with multiple rewards. A pair of preferred scenarios are consistent if the common variables in their variable assignments have identical values. The combination of two or more consistent preferred scenarios yields a new preferred scenario by merging their variable assignments. Thus, $|PS_2|$ can at most be $2^{|PS_1|}$ when all $ps \in PS_1$ are consistent.

To compute sets A_1 and A_3, we compute each $Pr\left(ps|s,a\right)$ for the current state s and each preferred scenario ps defined by some variable assignment. As mentioned earlier, if all state variables are assigned a value, then ps is a single state, otherwise it is a set of states. When ps is a single state, then its probability is the product of the conditional probabilities of each individual variable assignment that defines ps. When ps is a set of states, the aggregate probability is computed by variable elimination [12], which efficiently sums out the unassigned variables from the product of the conditional probabilities of each individual variable assignment.

To compute sets A_2 and A_4, we need to compute the probability of reaching a preferred state using the available actions. For a set of preferred states ps, let $L\left(ps|s,a\right)$ represent the cumulative probability of ultimately reaching any state in ps if action a is executed in the current state s, and the MDP policy is executed thereon. This is computed by solving the following recurrence relation.

$$L\left(ps|s,a\right) = Pr\left(ps|s,a\right) + \sum_{s' \notin ps} \gamma Pr\left(s'|s,a\right) L\left(ps|s', \pi\left(s'\right)\right) \quad \forall s,a$$

In practice, we use the variable elimination algorithm to perform each iteration of the recurrence efficiently (details are beyond the scope of this paper). Since the course advisor domain only has 3 terms, we do not discount the probabilities (i.e., $\gamma = 1$) and we only need to traverse 3 levels in the recurrence. For infinite horizon problems, the recurrence can be terminated when convergence is achieved (due to the discount factor) or when the probability of reaching a preferred state exceeds some threshold.

Given these sets, we need to determine which template should be used for each. For Template 1, we only need to check that $|A_5| = 1$. Template 2 is used if $|A_i| = 1 \ \forall i \in \{1 \ldots 4\}$ and the chosen action belongs to it. Let $vars(ps)$ denote the set of variables that determine the set of preferred states ps. Also let v_i denote the value required of the i^{th} variable in ps. Now Template 3 is used if $a_{chosen} = arg\,max_a\, Pr\left(ps|s,a\right)$ holds true. We substitute the probability with the cumulative probability $L\left(ps|s,a\right)$ for sets A_2 and A_4. If the chosen action belongs to a set but neither of Templates 2 or 3 is possible, we use Template 4.

4 Using Domain-Specific Constructs in Explanations

The templates mentioned in previous sections are completely domain-independent. Domain-specific constructs cannot be leveraged to convey information that is not encoded explicitly in the MDP. For instance, we know that failing a course is an undesirable state even though our model does not have a low utility value associated with all transitions resulting in it. The MDP is indirectly aware that getting an F in a course is undesirable since the course requirement can never be completed after a course has been failed.

The above discussion points towards the possibility of enriching explanations by using concepts from the domain. However, this requires additional effort on the part of a domain expert. In this section, we present a mechanism to assist the designer by defining an ontology for MDP explanations that encodes extra information so that it can be used in our templates.

The domain-specific information is encoded in the ontology using three basic concepts, namely variables, actions, and scenarios. Each variable has a name, a current value, and can optionally have different properties associated with it. Each action also has a name, a field to indicate its availability given the current values of all other variables, and optional information about what variables this action may affect. A scenario is described by a list of variables and their values, with a field *isDesirable* indicating whether it is desired or undesired.

The ontology is bootstrapped with the names of the courses and possible actions. It is also populated with domain-specific scenarios during the initialization step. Note that the optimal value function could also be used to determine which states are desirable/undesirable, but the user may not appreciate why high/low value states are necessarily desirable/undesirable. This would also require scenarios that specify complete variable assignments. Using the ontology, a domain expert can ensure that the scenarios have a small subset of MDP variables and scenarios that will be well-understood by users. For example, we can create a scenario with the course as a variable and give F as its value. We set the *isDesirable* field of this scenario to -1. The negative sign indicates that it is undesirable and the magnitude indicates the level of desirability of the scenario. We can create such assignments in the ontology for every course so that the system knows that getting an F in any course is undesirable.

We can augment the sets of preferred scenarios extracted from the MDP with this new set. We still favor explanations involving preferred scenarios extracted from the MDP because the MDP policy is determined solely by them. Since preferred scenarios derived from the ontology do not have explicit rewards, they only provide an intuition regarding good/bad states.

We also perform inference using the ontology to enhance our explanations. Consider the following example. Let each course be a variable having a name and its value the grade obtained. We associate a property *hasPreReq* with it. We define a new property *hasPreReqComp* that indicates whether all pre-requisites for a course have been completed. This property is computed by assigning Y as default value and then using the following rule.

$$\forall c \left(\exists p : hasPreReq\,(c,p) \bigwedge \left(hasVal\,(p,\text{'F'}) \bigvee hasVal\,(p,\text{'N'}) \right) \right) \Rightarrow$$

$$hasPreReqComp\,(c,\text{'N'})$$

We also define a property *hasCourse* for each action that indicates the course appearing in the action. Now we use the following rule to find available actions such that the courses appearing in them have $hasPreReqComp =$ 'Y'. We only need to consider actions having $isAvailable =$ 'Y' while generating explanations.

$$\forall a \left(\exists c : \left(hasCourse\,(a,c) \bigwedge hasPreReqComp\,(c,\text{'Y'}) \right) \right) \Rightarrow isAvailable\,(a,\text{'Y'})$$

Using our ontology, it is clear that we can represent three different types of domain knowledge. First, we can indicate various preferred scenarios. Second, we can include concepts specific to a domain, such as pre-requisites. Third, we can provide additional rules to prune the set of available actions.

5 Implementation Details and Sample Explanations

5.1 Course-Selection Advising Model and Implementation

Our course-selection MDP has 4 pre-requisite courses: *cs240*, *cs241*, *cs251*, and *cs350*, and 7 elective courses: *cs343* and *cs454* (pre-req *cs350*), *cs457* (pre-req *cs343*), *cs445* and *cs448* (pre-req *cs240*), *cs370* (pre-reqs *cs251* and *cs343*), and *cs372* (pre-req *cs370*). It has $\binom{7}{2} = 21$ possible actions, with each action representing a pair of courses. The first three electives belong to *area1*, the next two to *area2* and the last two to *area3*. We also populate the ontology with the course names, all possible actions and then list preferred scenarios. We identify 9 preferred scenarios, 1 for each area with value *Y* indicating that it is covered, and 6 for the number of courses being greater than 0. Later we generate all combinations of 4 or fewer preferred scenarios to get sets of desired states with multiple rewards. We cannot combine 5 or more scenarios because *courseCount* can only have a single value. We solve the MDP to yield its optimal policy π^* which is stored for future use. Whenever a student asks for advice, her current grades are entered into the model and the policy is consulted for the recommended action $a = \pi^*(s)$. If the student asks for an explanation, we then populate the ontology with the current state and run our rules to get available actions. The explanations are generated using these available actions and the preferred scenarios defined earlier. Note that if a student is already in a preferred scenario, for instance if the course requirements are already complete, then that preferred scenario is not included in our explanations.

5.2 Sample Explanations

Consider a student who has not completed any electives and who asks for advice from the system. For this student, all elective courses have value 'N', and the book-keeping variables have their initial values. The policy returns *act_ cs343cs348* as the optimal action. The explanation provided for this action is:

– Action *act_ cs343cs448* leads to a preferred scenario with: $area1 = Y$, $area2 = Y$, $area3 = Y$, $courseCount = 6$, $termCount = 3$ with highest probability: *73.51%*

The above explanation is generated using Template 3 for set A_2 by only using the preferred scenarios extracted from the MDP. It reflects the likelihood (73.51%) of fulfilling both degree requirements (area and course.) Five other templates are also populated for this state. The exact number of templates that can be populated depends on the structure of the model. For this problem, we can reach almost all preferred scenarios from the initial state, so a large number of templates can be populated. Thus, if we include preferred scenarios defined in the ontology, 450+ templates can be populated. A sample explanation using a preferred scenario from the ontology is:

– Action *act_ cs343cs448* leads to a preferred scenario with: $area1 = Y$, $area2 = Y$, $courseCount = 2$ immediately with highest probability: *90.25%*

Assume that the student disregarded the above advice, and instead took and passed two courses *cs454* and *cs448* with grades G and P respectively. The new state of the MDP has modified values for *courseCount*, *area1*, *area2* and *termCount*. If the student seeks a recommendation now, the recommended action is *act_ cs343cs370*. A sample explanation using preferred scenarios from the ontology is:

– Action *act_ cs343cs370* leads to a preferred scenario with: $area1 = Y$, $area2 = Y$, $area3 = Y$ immediately with highest probability: *95.00%*

This explanation shows that with this action the student is very likely (*95%*) to cover an extra area (*area3*) immediately. The number of populated templates using preferred scenarios from ontology decreases to 40 at this point. Let us assume that the student accepts this recommendation and gets grade G in both courses. Now *area3*, *termCount*, and *courseCount* also change and the new recommended action is *act_ cs457cs445*. The number of templates populated is much fewer (only 9) at this stage.

We evaluated the number of templates populated for other states of the MDP and the same trend was observed; initially we have more explanations available, but as the variables are modified from their default values the number

decreases. This matches our intuition, since the grades of courses do not change, so as the variable assignments increase, the number of preferred scenarios that can be reached decreases, and we populate fewer templates. This behaviour may vary for other domains and models depending on their structure. It may be argued that fewer templates indicates a stronger explanation because they are only explaining the relevant aspects of the choice. We are currently investigating techniques for determining the optimal number of explanations, which taken together, can justify a particular choice completely.

6 Conclusion and Future Work

We presented a mechanism to generate explanations for an action chosen by a policy computed by solving an MDP. We identified a set of templates that can be used for any domain in which the MDP is being used. Our approach does not require any additional effort from the MDP designer. We intend to explore alternate explanations for situations when none or too many of our templates can explain the choice of an optimal action. We also plan to develop an interactive mechanism in which users can request further justification using follow-up questions. Finally, we also plan to seek feedback from real-world users on the effectiveness of explanations presented through our templates.

References

1. Boutilier, C., Dean, T., Hanks, S.: Decision theoretic planning: Structural assumptions and computational leverage. JAIR **11** (1999) 1–94
2. Tintarev, N., Masthoff, J.: A survey of explanations in recommender systems. In: ICDE Workshop on Recommender Systems & Intelligent User Interfaces. (2007)
3. Herlocker, J.: Explanations in recommender systems. In: CHI' 99 Workshop on Interacting with Recommender Systems. (1999)
4. McGuinness, D.L., Glass, A., Wolverton, M., da Silva, P.P.: Explaining task processing in cognitive assistants that learn. In: Proceedings of AAAI 2007 Spring Symposium on Interaction Challenges for Intelligent Assistants. (2007)
5. Druzdzel, M.: Explanation in probabilistic systems: Is it feasible? Will it work? In: Fifth International Workshop on Inteligent Information Systems. (1996)
6. Chajewska, U., Halpern, J.: Defining explanation in probabilistic systems. In: Thirteenth Conference on Uncertainty in Artificial Intelligence. (1997)
7. Gärdenfors, P.: Knowledge in Flux: Modeling the Dynamics of Epistemic States. MIT Press (1988)
8. Pearl, J.: Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann (1998)
9. Lacave, C., Díez, F.J.: A review of explanation methods for Bayesian networks. The Knowledge Engineering Review **17** (2002) 107–127
10. Lacave, C., Luque, M., Díez, F.: Explanation of Bayesian networks and influence diagrams in Elvira. IEEE Transactions on Systems, Man, and Cybernetics **37**(4) (2007) 952–965
11. Elizalde, F., Sucar, E., Reyes, A., deBuen, P.: An MDP approach for explanation generation. In: Workshop on Explanation-Aware Computing with AAAI. (2007)
12. Zhang, N., Poole, D.: A simple approach to Bayesian network computation. In: Canadian Artificial Intelligence. (1994) 171–178