Dialog Strategies for Adaptive Help Systems

Thorsten Bohnenberger
Department of Computer Science
Saarland University, D-66041 Saarbrücken
bohnenberger@cs.uni-sb.de

Abstract

Many help systems provide assistance independent of their user's current situation. However, there are good reasons that two users in different situations should be assisted differently while dealing with the same problem. We describe a decision-theoretic planning approach by which an adaptive help system can determine its dialog strategy in accordance to the user's current needs. These needs, e.g. the need for a rather comprehensive or a rather concise sequence of instructions, can be derived from a user model. We show how information from the user model can be used to parameterize the decision-theoretic planning process and therewith to generate dialog strategies adaptively.

Keywords: Decision-theoretic planning, user modeling

1 Introduction

Passenger $\mathcal{P}$'s flight is delayed. His business partner is going to pick him up at the airport, therefore $\mathcal{P}$ wants to make a phone call to inform him about his late arrival. He finds a phone which operates on credit card. As he has never used a credit card phone before, he consults his mobile airport assistance system $\mathcal{S}$, running on his PDA. The help system instructs $\mathcal{P}$ step by step how to operate the phone. $\mathcal{P}$ finds $\mathcal{S}$'s comprehensive instructions very convenient and follows them without problems.

Passenger $\mathcal{Q}$ arrives late at the airport. After he overslept in the morning, he hurried out of the house and forgot to turn off the coffee machine. There are just a few minutes remaining before the boarding, but as $\mathcal{Q}$ fears a fire in his flat, he desperately wants to call his neighbor before departure. The only phone $\mathcal{Q}$ can find works on credit card—and just like $\mathcal{P}$, $\mathcal{Q}$ has never used a credit card phone before. His mobile airport assistance system adapts to his needs. $\mathcal{S}$ presents him the information needed to operate the phone in a concise way: it gives several simple instructions in one turn, but at the same time seems to care about not to overload $\mathcal{Q}$ with information. $\mathcal{Q}$ remains concentrated. He makes his call quickly and later reaches the gate just in time.

Unlike e.g. [8], who employs terminological reasoning for the generation of user-adapted plans, or [7], who considers resources within the classical planning paradigm, this article describes how $\mathcal{S}$'s adaptive behavior described above can be achieved by means of *decision-theoretic planning*. We show how the dialog between a user $\mathcal{U}$ and a system $\mathcal{S}$ can be modeled as *Markov decision process (MDP)*, illustrate the system behavior with the example of operating a credit card phone and discuss possible enhancements of $\mathcal{S}$'s adaptive behavior not included in the current implementation. To describe how $\mathcal{S}$'s adaptive behavior relates to the user model, we show how information about *time_pressure* and *cognitive_load* derived from a user model can be used to parameterize the decision-theoretic planning process.

2 Background

[1] describes in detail how the decision-theoretic planning approach can be applied in an abstract situation referring to an experimental environment[1] introduced in [5]. The article shows how *empirical data* derived from the experiment (the probabilities of $\mathcal{U}$ executing instructions correctly and the instruction costs in terms of the time it takes $\mathcal{U}$ to execute instructions) are used to determine $\mathcal{S}$'s instruction strategy (or in terms of decision-theoretic planning: *instruction policy*). On the one hand, $\mathcal{S}$ is supposed to give the instructions comprehensive enough to make $\mathcal{U}$ execute them correctly, on the other hand, $\mathcal{S}$ should avoid to take unnecessarily long to instruct $\mathcal{U}$. In other words, $\mathcal{S}$ is supposed to achieve a trade-off between a very fast but error-prone interaction and a slower but almost certainly correct interaction. For this purpose, $\mathcal{S}$ has the possibilities to give the instructions step by step or to group several instructions in bundles of 2, 3 or 4 instructions.

Some examples of interesting applications for the decision-theoretic computation of interaction strategies are already mentioned in [1]. While [1] and [6] emphasize the applicability of the decision-theoretic approach for human-computer interaction in principle and based on empirical data, the focus of this article is the integration of decision-theoretic planning and user modeling explained with a concrete example in an airport scenario.

3 Adaptive assistance for using a credit card phone

Operating a credit card phone requires a sequence of actions starting with keeping the credit card ready and lifting the handset, then dialing some preliminary digits before entering credit card information and eventually dialing the desired number. $\mathcal{S}$ could use the eight instructions listed in Table 1 to assist $\mathcal{U}$ during the operating process. However, as the introduction tried to demonstrate, neither giving all of these instructions step by step[2], nor giving all of them in one turn seems to be the appropriate strategy in all situations. Instead, $\mathcal{S}$ should give the instructions in the most convenient way for $\mathcal{U}$'s situation at hand, i.e. it should select the bundling of

[1] We do not repeat the details of the experiment as it has already been described in both [1] and [5].

[2] As $\mathcal{S}$ is an external help system (a system not integrated in the device to operate), $\mathcal{U}$ has to give feedback each time he has finished an instruction—but giving feedback takes time and is annoying if required unnecessarily often.

Table 1: Instructions for operating a credit card phone

1. ”Keep your credit card ready!”
2. ”Lift handset!”
3. ”Dial 0!”
4. ”After the tone, dial 9!”
5. ”After the tone, enter your credit card number!”
6. ”Enter two digits for the month of the expiring date!”
7. ”Enter two digits for the year of the expiring date!”
8. ”After the tone, dial the desired number!”

instructions promising the *maximum expected utility* for $\mathcal{U}$. This can be achieved by modeling the dialog as Markov decision process (see e.g. [11, 4] for a survey).

3.1 Modeling as Markov decision process

A Markov decision process models the stages of a dialog between $\mathcal{S}$ and $\mathcal{U}$ as states connected by stochastic transitions. The transitions describe the system dynamics induced by both $\mathcal{S}$'s and $\mathcal{U}$'s actions. A decision process is called *Markov/Markovian* (or is said to meet the *Markov property*), if for all states, the transitions from one state to another do not depend on the state's history, but only on the state itself. To describe the system dynamics of the instruction process, i.e. the interaction between $\mathcal{S}$ and $\mathcal{U}$, we need to define the features of the states and how they are changed by following the transitions for corresponding actions. Figure 1 illustrates how we construct the MDP for the interaction between $\mathcal{S}$ and $\mathcal{U}$.

Each state consists of 4 features: (1) N_TO_GIVE—the number of instructions $\mathcal{S}$ still has to give, (2) N_IN_BUNDLE—the current length of the instruction bundle, (3) N_IN_WM—the number of instructions $\mathcal{U}$ currently keeps in working memory and (4) CORRECT_PERFORMANCE?—the information if $\mathcal{U}$ has avoided making any errors so far. The action GIVE_INSTRUCTION generates a transition leading to a successor state in which N_TO_GIVE is decreased by 1, while N_IN_BUNDLE and N_IN_WM are increased by 1. If $\mathcal{U}$ has avoided making any errors so far, the action WAIT_FOR_EXECUTION generates transitions to two successor states. In both successor states N_IN_WM is decreased by 1, moreover N_IN_BUNDLE is set to 0, iff at the same time N_IN_WM is decreased from 1 to 0. The successor states only differ in the feature CORRECT_PERFORMANCE? One state is reached with probability $p_{correct}$ (the probability that $\mathcal{U}$ executes the instruction correctly), the other with $p_{error} = 1 - p_{correct}$. From a state in which the feature CORRECT_PERFORMANCE? is already negative, WAIT_FOR_EXECUTION leads to the successor state in which CORRECT_PERFORMANCE? is negative with probability $p_{error} = 1$. The transitions for both actions GIVE_INSTRUCTION and WAIT_FOR_EXECUTION are annotated with costs—the time it takes $\mathcal{S}$ to give an instruction or $\mathcal{U}$ to execute an instruction, respectively.

We make the following basic assumption, which is underpinned by the empirical data of the experimental study: the probability that $\mathcal{U}$ executes an instruction incorrectly depends significantly on how long $\mathcal{U}$ had to memorize the instruction before executing it. To illustrate this: a person who has never used a credit card phone before, and who is presented the complete sequence of eight instructions in one turn, might easily forget a step during the execution or

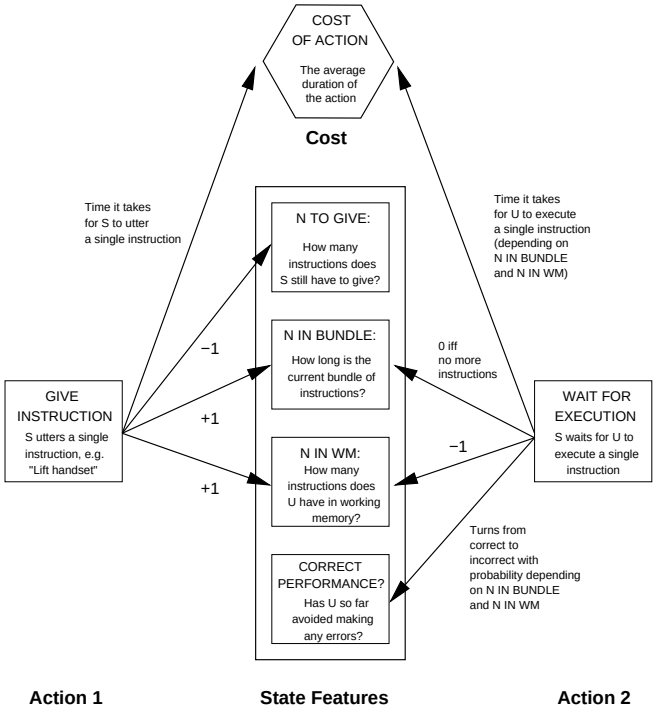

Figure 1: Construction of the MDP modeling the interaction between $\mathcal{S}$ and $\mathcal{U}$

execute instructions in the wrong order. Moreover, some steps are inherently more error-prone than others, and some take longer execution time than others (e.g. dialing a single digit vs. entering a complete credit card number). Such instruction dependent differences in the probabilities and costs are not taken into account in the model for the experimental scenario described in [1], as all instructions in the experiment had similar complexity.

3.2 Parameterization by the user model

Our overall goal is to make the system adapt its behavior to $\mathcal{U}$'s current situation, i.e. to make the system behavior dependent on the user model, which $\mathcal{S}$ maintains during the interaction with $\mathcal{U}$. A common technique used to model a user are *Bayesian networks* [12]. Two variables, which have major influence on how information should be presented to $\mathcal{U}$, are TIME_PRESSURE and COGNITIVE_LOAD [13]. Inference concerning these variables can currently be drawn both from symptoms occurring during speech input [10] (e.g. articulation rate, distractions, filled and unfilled pauses etc.) and symptoms occurring during manual input [9] (e.g. scrolling speed, tapping close to an icon, hard or soft tapping on the display etc.). In the near future, also information gained by bio-sensors (e.g. pulse rate, surface energy of the skin, blood pressure etc.) will be considered. The question is—if we have estimations of $\mathcal{U}$'s TIME_PRESSURE and COGNITIVE_LOAD—how should they influence the planning process, and hence the resulting dialog strategy?

If $\mathcal{U}$ is under time pressure, $\mathcal{S}$ should clearly present instructions concisely, i.e. $\mathcal{S}$ should rather choose bigger bundles and rely on $\mathcal{U}$ to memorize and execute them correctly. In terms

of the dialog modeled as MDP, this means that the importance of reaching a goal state in which $\mathcal{U}$ will have avoided to make any error, is rather low. Low importance of reaching an error-free goal state corresponds to assigning a rather low reward to the goal states. Accordingly, assigning higher reward to the goal states means attaching higher importance to $\mathcal{U}$ following the instructions error-free. And in fact higher rewards yield dialog strategies using small bundles or giving instructions step by step. Therefore, the reward R assigned to the goal states of the MDP can be determined according to the formula:

$$R = R_{low} time\_pressure \ + \ R_{high}(1 - time\_pressure) \tag{1}$$

with $time\_pressure \in [0, 1]$ corresponding to the variable TIME_PRESSURE of the Bayesian network, and R_{low}, R_{high} appropriate values for low and high rewards of goal states, respectively.

If $\mathcal{U}$ shows symptoms of cognitive load, he is more likely to execute instructions incorrectly than if he would be fully concentrated. This means, COGNITIVE_LOAD directly influences the probabilities assigned to the transitions of the MDP. If the probability that $\mathcal{U}$ executes instructions correctly equaled 1 no matter what the bundling size is, $\mathcal{S}$ could always choose the biggest bundling size: this would promise the lowest cost in terms of time needed to execute the instructions and to give feedback. But, as we said before, the size of the bundle matters. Therefore, we assume that the probability that $\mathcal{U}$ executes an instruction correctly depends on both how long $\mathcal{U}$ had to memorize the instruction (i.e. the instruction's position in a bundle) and $\mathcal{U}$'s current cognitive load. In fact, this is also underpinned by the empirical data derived from the experiment mentioned in section 2: subjects who are distracted by a secondary task are more likely to execute an instruction incorrectly than others. This can be summarized by the following formula for $\mathcal{U}$'s probability $p_{i,n}$ to execute the ith instruction in a bundle of size n incorrectly:

$$p_{i,n} = p_{i,n,low}(1 - cognitive\_load) + \ p_{i,n,high} cognitive\_load \tag{2}$$

with $cognitive\_load \in [0, 1]$ corresponding to the variable COGNITIVE_LOAD of the Bayesian network, $p_{i,n,low}$ and $p_{i,n,high}$ appropriate values for error probabilities if $\mathcal{U}$'s cognitive load is low or high, respectively.

3.3 System behavior

We have integrated and tested the approach described above in a prototypical airport assistance system, which allows the specification of symptoms for time pressure and cognitive load occurring during $\mathcal{U}$'s interaction with $\mathcal{S}$. In its start configuration, let $\mathcal{S}$ expect a probability of 0.5 for both variables TIME_PRESSURE and COGNITIVE_LOAD. $\mathcal{S}$'s instruction policy for this configuration is to give the first four instructions in pairs. Consider then the following three possible progressions of the interaction: (1) $\mathcal{U}$ speaks with average speed and shows symptoms such as distractions or filled pauses in his speech. $\mathcal{S}$ increases its expectation for COGNITIVE_LOAD while its expectation for TIME_PRESSURE remains nearly constant. After $\mathcal{U}$ has executed the first two instruction pairs, $\mathcal{S}$ changes its instruction policy and gives the remaining four instructions step by step. (2) $\mathcal{U}$ speaks very quickly, yet, does not show symptoms such as distractions or filled pauses in his speech. $\mathcal{S}$ increases its expectation for TIME_PRESSURE and at the same

time decreases its expectation for COGNITIVE_LOAD. After $\mathcal{U}$ has executed the first two instruction pairs, $\mathcal{S}$ changes its instruction policy and gives the remaining four instructions all in one turn. (3) $\mathcal{U}$ speaks very quick and show symptoms such as distractions or distractions or filled pauses in his speech. $\mathcal{S}$ increases its expectations for both TIME_PRESSURE and COGNITIVE_LOAD. After $\mathcal{U}$ has executed the first two instruction pairs, $\mathcal{S}$ sticks with its instruction policy an gives the remaining four instructions in pairs as well.

These are only three examples of how changes in the user model influence $\mathcal{S}$'s instruction strategy. Yet, $\mathcal{S}$ can adapt its instruction strategy to any given user model configuration in this way. This yields a much more precise mapping from user model configurations to dialog strategies than e.g. using threshold values and a simple decision tree.

3.4 Variations and enhancements

Initially, we assumed that $\mathcal{S}$ plans the complete sequence of eight instructions in one turn. If we do not expect any user to be able to memorize and follow all eight instructions correctly in one turn, then it is reasonable to partition the instruction sequence. A reasonable partition for the eight instructions at hand is to consider two sets of four instructions—as we did in the previous section. Apart from complexity reasons, partitioning the instruction sequence has another advantage: to plan the second set of instructions, $\mathcal{S}$ can consider symptoms for time pressure and cognitive load which $\mathcal{U}$ showed during the execution of the first set. Other partitions, dependent of the total number of instructions or relations of content among each other, are possible.

So far, $\mathcal{S}$ only determines the bundling sizes for a fixed sequence of actions. Allowing $\mathcal{S}$ to decide if an instruction can be omitted completely, is a reasonable variation of adaptivity. E.g. the instruction "Lift handset!" can probably often be left out for a user under time pressure without posing a problem. Although $\mathcal{U}$ could lift the handset at the wrong stage of the interaction, $\mathcal{U}$ would at least lift the handset at some point—and with a little luck, it might just be the right one. For the MDP this means, that the states with the transitions for "Lift handset!" additionally gets transitions for "Dial 0", which skip the states in which the "Lift handset!"-transitions end. Obviously, the cost of the interaction is reduced by skipping the "Lift handset!"-transition, but the probability to reach the next state error-free is reduced as well. Clearly, this enhancement fits well into the decision-theoretic model.

The augmentation of the dialog with different presentation modes is the last enhancement to be discussed. Consider e.g. the instruction "Enter two digits for the month of the expiring date!" $\mathcal{S}$ can present the content of this instruction in a very detailed version (e.g. "There is an expiring date indicated on your credit card. It is written like this: first there are two digits for the month, then there is a slash, and then there are another two digits for the year. Please enter the two digits for the month!") or it could choose a short version (e.g. "Enter expiring date: month!"). In [2, 3] a similar approach using different presentation modes is applied for the planning of navigation recommendations. In fact, an instruction policy can be considered as a means to help $\mathcal{U}$ to navigate through the stages of a dialog. In the MDP, additional presentation modes are mirrored by additional transitions between the states. Obviously, there are different costs for $\mathcal{S}$ giving an instruction and different probabilities for $\mathcal{U}$ executing an instruction correctly associated with different presentation modes. As in the case of omitting instructions, this enhancement fits well into the decision-theoretic model.

4 Discussion

We tried to motivate the application of decision-theoretic planning for the achievement of adaptive behavior of a help system in terms of its dialog strategy. A concrete example was chosen to illustrate the added value of the adaptive behavior. Cases like those of passenger $\mathcal{P}$ and $\mathcal{Q}$ in the introduction can properly be dealt with, as we experienced when applying the approach in the prototypical airport assistance system of the project READY[3]. Yet, it is also a matter of fine tuning the system to achieve the desired system behavior within the complete spectrum of situations, which can occur. This problem amplifies, as we could not use empirically derived or learned probabilities for the scenario, but had to use estimates. Learning these probabilities would clearly be a neat but also costly procedure to ground the decision-theoretic planning approach.

A very concrete problem occurred, when we tried to enhance the modeling by allowing the system to repeat instructions, if the user has not understood or could not execute an instruction correctly. Although it is not a problem to enhance the MDP such that it can deal with repetitions, the question is: how should the system as a whole react to such kind of feedback? Although the instruction policy would in principle be able to repeat instructions, the fact that $\mathcal{U}$ is not able to follow the instructions as determined by the current policy indicates, that $\mathcal{S}$ should rather recompute the complete policy with adjusted parameters from the user model. It is still an open question what the best strategy in such situations might be.

Acknowledgments

The research described in this paper was funded by 'Deutsche Forschungsgemeinschaft' DFG within their collaborative research program 378 on "resource-adaptive cognitive processes". The experiment, mentioned in section 2 was developed and conducted in collaboration with Leonie March and Ralf Rummer of the Department of Psychology of the Saarland University. The decision-theoretic planning approach was devised and integrated into the READY prototype in collaboration with Anthony Jameson.

References

[1] Thorsten Bohnenberger. How can an intelligent dialog system plan ahead? In Martin E. Müller, editor, *ABIS-00, Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen*, Osnabrück, Germany, 2000. Universität Onsabrück. Available from http://w5.cs.uni-sb.de/∼bohne/publications/Bohnenberger00.ps.gz.

[2] Thorsten Bohnenberger and Andreas Butz. Decision-theoretic planning of navigation instructions. In *Workshop notes of the ECAI 2000 workshop on Artificial Intelligence in Mobile Systems, August 22, 2000, Berlin, Germany*. ECCAI, 2000. Available from http://w5.cs.uni-sb.de/∼bohne/publications/BohnenbergerButz00.ps.gz.

[3] **RE**source **A**dapted **D**ialog s**Y**stem; http://w5.cs.uni-sb.de/∼ready/

[3] Thorsten Bohnenberger and Anthony Jameson. When policies are better than plans: Decision-theoretic planning of recommendation sequences. In James Lester, editor, *IUI 2001: International Conference on Intelligent User Interfaces*, pages 21–24. ACM, New York, 2001. Available from http://dfki.de/∼jameson/abs/BohnenbergerJ01.html.

[4] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Artificial Intelligence Research*, 11, 1999.

[5] Barbara Großmann-Hutter and Christian Müller. Experimentelle Untersuchung von Spracheingaben unter kognitiver Belastung zur Benutzermodellbildung [An experimental investigation of speech input under cognitive load for user modeling]. In Tania Jörding, editor, *ABIS-99, Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen*. Otto-von-Guericke University, Magdeburg, Germany, 1999. Available from http://dfki.de/∼jameson/abs/Grossmann-HutterM99.html.

[6] Anthony Jameson, Barbara Großmann-Hutter, Leonie March, Ralf Rummer, Thorsten Bohnenberger, and Frank Wittig. When actions have consequences: Empirically based decision making for intelligent user interfaces. *Knowledge-Based Systems*, 14:75–92, 2001. Available from http://dfki.de/∼jameson/abs/JamesonGMRBW01.html.

[7] Jana Koehler. Planning under resource constraints. In *European Conference on Artificial Intelligence, ECAI-98*, pages 489–493. ECCAI, 1998. Available from http://citeseer.nj.nec.com/koehler98planning.html.

[8] Detlef Küpper. Plan processing in user models. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 447–448. Springer Wien New York, Vienna, New York, 1997.

[9] K. Lindmark and D. Heckmann. Interpreting symptoms of time pressure and cognitive load in manual input. In Martin E. Müller, editor, *ABIS-00, Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen*, Osnabrück, Germany, 2000. Universität Onsabrück. Available from http://fsinfo.cs.uni-sb.de/∼dominik/papers/ABIS2000.ps.

[10] Christian Müller, Barbara Großmann-Hutter, Anthony Jameson, Ralf Rummer, and Frank Wittig. Recognizing time pressure and cognitive load on the basis of speech: An experimental study. In Julita Vassileva, Piotr Gmytrasiewicz, and Mathias Bauer, editors, *UM2001, User Modeling: Proceedings of the Eighth International Conference*. Springer, Berlin, 2001. Available from http://dfki.de/∼jameson/abs/MuellerGJ+01.html.

[11] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[12] Ralph Schäfer. *Benutzermodellierung mit dynamischen Bayes'schen Netzen als Grundlage adaptiver Dialogsysteme*. PhD thesis, Universität des Saarlandes, Germany, 1999.

[13] Frank Wittig. Empirisch basierte Benutzermodellierung mit Bayes'schen Netzen: Strukturelle Aspekte. In *ABIS-01, Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen*, Dortmund, Germany, 2001. Universität Dortmund. To appear.