Decision Theoretic Instructional Planner for Intelligent Tutoring Systems

Noboru Matsuda[1] and Kurt VanLehn[2] *

[1] Intelligent Systems Program, University of Pittsburgh,
[2] Learning Research and Development Center, University of Pittsburgh
3939 O'Hara Street, Pittsburgh PA 15260

1 Introduction

For many years, ITS researchers have strived to provide better instruction. They have various kinds of student models and expert models, as well as models of student-tutor interactions. However, little research have been conducted on instructional planning, which attempts to make a sequence of instructions optimal for a student. Obviously, the better the planner, the better instructions the ITS would provide.

We have hypothesized that major difficulties of instructional planning come from following three issues:

(a) *Inaccuracy of student mode* – Not only do computers have trouble building a precise student model, but human tutors do, too (Putnam, 1987). Narrow bandwidth of communication is an essential barrier to tutoring. Consequently, ITS may have limited accuracy of student models (see, e.g., VanLehn, 1988).
(b) *Failure in instructions* – Since a student might not understand or misunderstand what the tutor said, an instructional plan may always suffer from being interrupted.
(c) *Unexpected responses* – The more the tutor allows students to take the initiative in communication, the more the chances of encountering unexpected behaviors (e.g., asking a question, entering an answer that was not actually asked to do, etc.), which in turn make the instructional plan unable to continue.

Instructional planning determines a sequence of tutoring actions that maximizes students learning activities, while taking into account above issues. Our approach is to consider instructional planning as a decision theoretic planning problem based on Markov decision processes, which determines a sequence of actions that maximizes the outcome (i.e., rewards) while dealing with the following uncertainties: (Boutilier, Dean & Hanks, 1999)

* Email: mazda@isp.pitt.edu and vanlehn@cs.pitt.edu This research was supported by NSF grant number 9720359 to CIRCLE: Center for Interdisciplinary Research on Constructive Learning Environments. http://www.pitt.edu/~circle

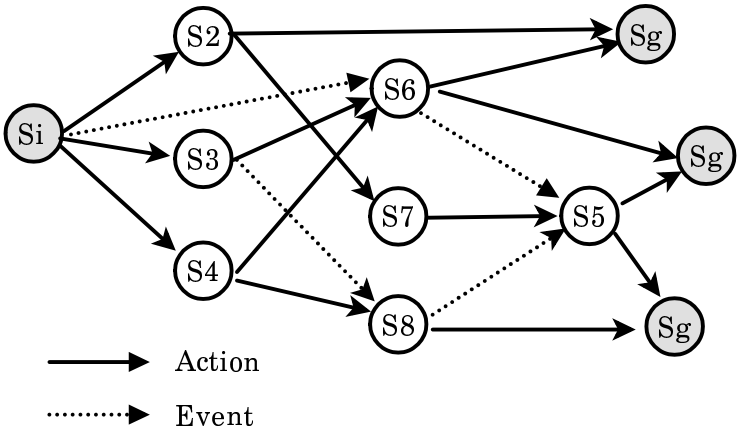

Fig. 1. State transition as a model of instruction.

- Sensory weakness – the planning agent never knows exact state of the world. This corresponds to the inaccuracy of the student model.
- Uncertain outcome – a failure of the action taken affects the plan. This corresponds to the failure in instruction.
- Unexpected event – the world might be suddenly changed due to an event that is not in the plan. This corresponds to unexpected responses.

In order to work out the implications of the decision theoretic approach, we have built a simple decision theoretic planner for intelligent tutoring systems. The planner has been implemented in a simple ITS to teach augmentation skills. An experiment suggests that the decision theoretic planning schema will work fine for an instructional planner in a full-scale system. In the remaining sections, we discuss the issues of instructional planning in detail, and give a basic framework of the instructional planner that implements a model of Markov decision processes. We then show a prototype ITS that is integrated with this instructional planner and discuss its evaluation.

2 Issues on Instructional Planning

2.1 State Transition as a Model of Instructional Interaction

Instructional interaction is modeled by state transition as shown in Fig. 1.

- A ***state*** represents tutor's belief about a tutoring situation, which consists of student's knowledge status, an expected response from the student, and some other information carried from the actions applied. Each state is associated with a value (i.e., expected utility) that represent desirability of being in that state.

– An ***action*** is instructional interaction provided to the student. Taking action changes the state, but since the student's response to an action is not deterministic, a given action might lead with different probabilities to several different states.
– An ***event*** is an unexpected action by the student, such as asking a question, which can also cause a state transition.

At the beginning of instruction, the tutor assumes that the student has certain amount of knowledge (S_i in Fig. 1). The goal of the instruction should be also given as a state (S_g). Several goal states might exist when they are equally acceptable.

Instructional planning is considered as a repeated decision making process where, at each state, the tutor must determine an action that transitions the current state into better state.[1] Note that all the possible situations that can be reached in a given tutoring session must be described as *states*.

2.2 Uncertainty in State Transitions

In the state-transition model, the uncertainties mentioned in the previous section are formalized as follows.

Since each state represents a different state of the student model, inaccuracy in student modeling acts like inaccuracy in sensing in robotics. The agent thinks it is in one state when it is really in another. For example, even if a student doesn't understand a particular concept the student model might claim that he does, which in turn makes the tutor believe being in a wrong state.

Since the same instructional action may have different effects, the planner must deal with multiple successor states when an action is done in a state. For example, teaching a concept reaches a state where the student either does or does not understand the concept. One can, however, represent the effect of an instruction as a probability distribution.

Finally, an unexpected response from a student (i.e., the unexpected event) causes a contingent state transition. Student's asking a question is one of the most frequently happened unexpected events.

Since all the uncertainties listed here can be represented as probabilities over the states, it might be good idea to build an instructional planner as a decision theoretic agent that determines an optimal action to take.[2] Applying an instructional action results in a state, so if each state is associated with a certain value that represents the "desirability" of being in such situation, then we can adopt the technique used in the planning literature. This desirability corresponds to the expected utilities. Thus, the optimal action is determined by calculating a maximum expected utility. We discuss an architecture of the planner in the section 3.

[1] Meaning of a "better" state, of course, depends on the tutoring context.
[2] In this paper, we do not discuss uncertainty of the student model. It is an extension for the future work.

2.3 Reactive Planning as State Recapturing

If the student's response is not an expected one, then the tutor simply throws the current plan away and recaptures the state to generate a new plan. This is one of the solutions for reactive planning problem.

There are several approaches to implementing a reactive planner. For instructional planning, calculating the best action only for the current state and ignoring future actions might be the best strategy. The main reasons to prefer this rather conservative approach is that it is quite costly to predict all the possible student's responses due to their uncertainties.

Since a decision theoretic planner reads the policy table to determine an optimal action at every state, it doesn't need to plan future actions at all. After applying an action, however, it is required to examine the resulting state according to student's response. This process is totally domain dependent and might be fairly ad hoc.

2.4 Related works

There exist several papers discussing planning for ITS. One of the earlier works was conducted by Peachey and McCalla (1986) where they proposed partial ordering planner with STRIPS-like operators. The operators correspond to tutoring rules, which state the preconditions and actions to teach a certain concept. The generated plans are partially ordered. They also propose a method to replan the faulty plan. The instructions generated by the planner, however, is quite opportunistic; it tends to teach everything that can be taught. There is no strategic consideration to select an operator.

MacMillan and Sleeman (1987) developed an instructional planner as an application of blackboard model. Although there exists some ambiguity on the generality of the proposed system, their planner has hierarchical structured instruction consisting of problem, strategy, tactic, and focus levels. Those aspects must be taken into account when one design an instructional planner.

Vessileva also developed a reactive planner for ITS (1995). Her planner first produces a plan, executes it as long as possible, and then replans when it eventually encounters unforeseen situations.

One of the most recent works is conducted by Murray and VanLehn (2000). They have used a dynamic decision network to implement turn-taking tutorial actions for coached problem solving. The network called *tutor action cycle networks* consists of (a) deciding/applying a tutorial action, (b) observing student's action (i.e., response), and (c) updating the studnet model based on those two actions.

Developing a generic shell to build an instructional planner is another approach. For example, Freedman (1999) designed a generic shell to build a hierarchical task network planner (i.e., hierarchical decomposition).

3 Probabilistic Planing Agent as an Instructional Planner

As discussed in the previous section, the effects of an instructional action can be represented as probability distribution over the successor states. It corresponds to the state transition matrix to model Markov decision processes (MDP) (Boutilier, Dean & Hanks, 1999). Tutoring rules are then represented as follows;

IF ***conditions*** are held in the current state,
THEN taking ***action***
RESULTS in ***states*** $\{S_1, \cdots, S_n\}$
WITH corresponding ***probabilities*** $\{P_1, \cdots, P_n\}$.

We need to define expected utilities for each state. It is well known that one can calculate the utilities by value iteration given that the states have rewards assigned. That is, given a reward function R and a transition matrix $M^a_{i,j}$ that represents a probability of reaching state j by taking an action a at a state i, a maximum utility in the state i is determined by the following formula:

$$U(i) = R(i) + \max_a \sum_j M^a_{i,j} U(j)$$

In the context of ITS, the reward represents an assessment of states with respect to instructional satisfaction. An example of reward function is discussed in section 4.

Once the utilities of states are determined, an optimal action for a state i (i.e., the optimal policy, $P(i)$) is calculated as follows:

$$P(i) = \arg\max_a \sum_j M^a_{i,j} U(j)$$

Since a reward is determined only by the state descriptions, one can assume that the reward for each state will never change throughout the tutoring session. For example, a value of the state where a student knows a particular concept should not be changed no matter when he learns the concept. Thus, once an optimal policy for each state is settled, the ITS can determine the action to be taken by simply looking up a table of the policy.

Fig. 2 shows basic cycle of the instructional planner. The planner reads an optimal action to be taken off the policy table. The executer applies the selected action and passes the evaluator a set of expected responses specified as the resulted states of the action. The evaluator then compares the student's response with the expected responses and updates the student model. In other words, the evaluator determines what exactly the resulted state is.

4 Evaluation on a Prototype System

4.1 The domain — Argumentation

To explore the idea of decision theoretic instructional planner, we have built an ITS to teach argumentation skills. The domain is selected from an existing

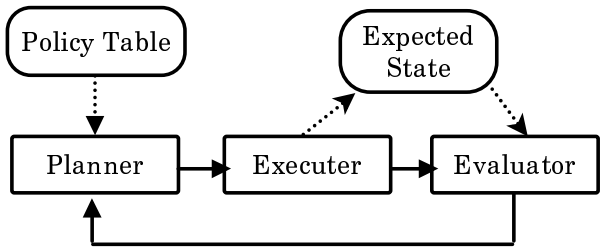

Fig. 2. Plan and execution cycle.

Table 1. An sample problem used for experiment.

Problem	Do you think Santa is real or not?
Hypotheses	Santa is real.
	Santa is not real.
Evidence	Santa signed the tag on my present.
	I sent him my Christmas list at his address.
	Santa can not deliver presents all over the world in one night.
	Santa is too fat to fit in the chimney.
	Reindeer can not fly.

ITS (Goodman, Soller, Linton & Gaimari, 1998). Students are asked to make an argument by stating supports for a given hypothesis with given pieces of evidence. An example of the problem is shown in Table 1.

The student model represents (a) the student knows (or does not know) that both positive and negative evidence are required for the argument, (b) the fact that student knows (or does not know) that to make an argument one must state all the evidence for the positive (negative) hypothesis, and (c) the available evidence. The eight statements shown in Table 2 denote the student model with respect to the problem shown in Table 1.

There are seven responses available for the student. They are shown in Table 3. The student can cite evidence by typing "K." Typing "H" is only one way to request a hint. The remaining commands are responses to questions or comments made by the ITS.

Finally, the nine actions taken by ITS are listed in Table 4. The instructional actions are embedded into the tutoring rules. For example, a tutoring rule to "agree with justification" in Table 5 means that

IF the student's previous response is `:NIL`, which means that the student just cited evidence, *and*

Table 2. The student model.

```
good_argument :- make_all_p-support, make_all_n-supprot.
make_all_p-support :- support(e1,hp), support(e2,hp).
make_all_n-support :- support(e3,hn), support(e4,hn),
support(e5,hn).
support(e1,hp).
support(e2,hp).
support(e3,hn).
support(e4,hn).
support(e5,hn).
```

Table 3. Possible student's responses.

linK	Cite evidence.
Hint	Ask "What should I do next?"
Agree	Agree with peer's suggestion
Disagree	Disagree with peer's suggestion
Related	Select "Because *evidence* is related to *hypothesis*."
Not-related	Select "Because *evidence* is not related to *hypothesis*."
reLevant	Select "Because *evidence* is relevant to this problem."
Why	Ask "Why?"
dOne	State that it is done.

Note: The underlined letters in the first column corresponds with the prompts from ITS shown in Table 6.

the student model (<sm>) is not stating that the student either has learned nothing (00000000) or has learned all the concepts (11111111)[3],
THEN prompting "Good, that's right. Why do you think so?",
RESULTS in either the student's response with Related (.4)
or with reLevant (.4)
or to input another support (.1)
or to request a hint (.1)

Given specifications of the student model and the students' responses, one can enumerate all the possible states. Those are simple combinations of a description of student model and a response. In the domain described here, there are 400 possible states.

Once the states are defined, a utility for each state is calculated from the reward function. For this experiment, we used the reward function listed below.

[3] The student model is implemented as eight digits each states if the student knows correspondent concept in Table 2.

Table 4. Instructional actions.

(1) Agree with student's response with justification
(2) Agree with student's response
(3) Disagree with student's response with justification
(4) Disagree with student's response
(5) Request a justification
(6) Explain an evidence
(7) Teach student `make_all_p-support`
(8) Teach `make_all_n-support`
(9) Teach `good_argument`

Table 5. An example of tutoring rule.

```
(:AGREE-WITH-JUSTIFICATION
  (:STATE :sm <sm> :response :NIL)
  (:WHEN (and (not (eq <sm> #b11111111))
              (not (eq <sm> #b00000000)) ))
  (:ACTION "Good, that's right. Why do you think so?")
  (:EFFECT ((single-state :sm <sm> :RESPONSE :S) .4)
           ((single-state :sm <sm> :RESPONSE :REL) .4)
           ((multi-sm-get-one <sm>) .1)
           ((single-state :sm <sm> :response :HINT) .1) ) )
```

- Each instructional step: -0.1
- To make the student learn a concept: 0.125
- Student's correct justification: 2.0 (that is, it is strongly recommended to make the student take self explanation)
- Student's incorrect justification: -100
- To reach a final state: 20
- Student's request for a hint: -1

Note that unlike the tutoring rules, the reward function is fairly subjective. Tutoring rules, on the other hand, are rather domain and problem dependent. They represent average effectiveness of the instructional actions. Thus, replacing the reward function causes different sequence of instructions.

4.2 Experiment

An example of interaction between a student and ITS is shown in Table 6. The lines beginning with '`S:`' are prompts for the student followed by his/her response after '`>`'. The lines beginning with '`C:`' are the feedback to the student. Although,

Table 6. An example of dialogue.

```
> (its:run)

C: Now, what do you think?

S: [linK Agree Disagree Related Not-related reLevant Hint Why dOne]
S: Your choice? [K,A,D,R,N,L,H,W,O] > k
S: Evidence? [(E1 E2 E3 E4 E5)] > e1

               State: (10000 :NIL)
               Take action-id: :AGREE-WITH-JUSTIFICATION-2
               Expecting:
                         (((16 :S)) 0.4)
                         (((16 :REL)) 0.4)
                         (((24 :NIL) (20 :NIL) (18 :NIL) (17 :NIL)) 0.1)
                         (((16 :HINT)) 0.1)

C: Correct.  Tell me why you think so?

S: [linK Agree Disagree Related Not-related reLevant Hint Why dOne]
S: Your choice? [K,A,D,R,N,L,H,W,O] > k
S: Evidence? [(E2 E3 E4 E5)] > e2

               State: (11000 :NIL)
               Take action-id: :AGREE-WITH-JUSTIFICATION
               Expecting:
                         (((24 :S)) 0.4)
                         (((24 :REL)) 0.4)
                         (((88 :NIL) (28 :NIL) (26 :NIL) (25 :NIL)) 0.1)
                         (((24 :HINT)) 0.1)

C: Good, that's right.  Why do you think so?

S: [linK Agree Disagree Related Not-related reLevant Hint Why dOne]
S: Your choice? [K,A,D,R,N,L,H,W,O] > o

               State: (11000 :DONE)
               Take action-id: :DISAGREE-WITH-DONE
               Expecting:
                         (((88 :NIL) (28 :NIL) (26 :NIL) (25 :NIL)) 0.7)
                         (((24 :HINT)) 0.3)

C: I do not think so.

S: [linK Agree Disagree Related Not-related reLevant Hint Why dOne]
S: Your choice? [K,A,D,R,N,L,H,W,O] > h

               State: (11000 :HINT)
               Take action-id: :EXPLANATION-E5
               Expecting:
                         (((25 :NIL)) 0.9)
                         (((88 :NIL) (28 :NIL) (26 :NIL)) 0.1)

C: Consider the link [e5-hn]

S: [linK Agree Disagree Related Not-related reLevant Hint Why dOne]
S: Your choice? [K,A,D,R,N,L,H,W,O] > w

               State: (11000 :WHY)
               Take action-id: :N-LINKS-FOR-WHY-INCOMPLETE
               Expecting:
                         (((88 :NIL) (28 :NIL) (26 :NIL) (25 :NIL)) 1.0)

C: Because you do not make all n-links
```

Table 7. An example of dialogue. (continued)

```
S: [linK Agree Disagree Related Not-related reLevant Hint Why dOne]
S: Your choice? [K,A,D,R,N,L,H,W,O] > o

              State: (11001 :DONE)
              Take action-id: :DISAGREE-WITH-DONE
              Expecting:
                        (((89 :NIL) (29 :NIL) (27 :NIL)) 0.7)
                        (((25 :HINT)) 0.3)

C: I do not think so.

S: [linK Agree Disagree Related Not-related reLevant Hint Why dOne]
S: Your choice? [K,A,D,R,N,L,H,W,O] > h

              State: (11001 :HINT)
              Take action-id: :EXPLANATION-E4
              Expecting:
                        (((27 :NIL)) 0.9)
                        (((89 :NIL) (29 :NIL)) 0.1)

C: Consider the link [e4-hn]

S: [linK Agree Disagree Related Not-related reLevant Hint Why dOne]
S: Your choice? [K,A,D,R,N,L,H,W,O] > k
S: Evidence? [(E3 E4)] > e4

              State: (11011 :NIL)
              Take action-id: :AGREE-WITH-JUSTIFICATION
              Expecting:
                        (((27 :S)) 0.4)
                        (((27 :REL)) 0.4)
                        (((91 :NIL) (31 :NIL)) 0.1)
                        (((27 :HINT)) 0.1)

C: Good, that's right.  Why do you think so?

S: [linK Agree Disagree Related Not-related reLevant Hint Why dOne]
S: Your choice? [K,A,D,R,N,L,H,W,O] > k
S: Evidence? [(E3)] > e3

              State: (11111 :NIL)
              Take action-id: :JUSTIFICATION
              Expecting:
                        (((31 :S)) 0.3)
                        (((31 :REL)) 0.3)
                        (((95 :NIL) (63 :NIL)) 0.3)
                        (((31 :HINT)) 0.1)

C: Why do you think so?

S: [linK Agree Disagree Related Not-related reLevant Hint Why dOne]
S: Your choice? [K,A,D,R,N,L,H,W,O] > l

              State: (11111 :REL)
              Take action-id: :AGREE
              Expecting:
                        (((95 :NIL) (63 :NIL)) 0.3)
                        (((31 :HINT)) 0.7)

C: that's right.  Go on!

S: [linK Agree Disagree Related Not-related reLevant Hint Why dOne]
S: Your choice? [K,A,D,R,N,L,H,W,O] >
```

the prototype system does not have enough tutoring rules to thoroughly teach the subject in the example domain, one might observe, in the trace of sessions, several advantages of the instructional planner.

First, look at the first tutor action, where the ITS took the action `AGREE-WITH-JUSTIFICATION-2` and responded "*Correct. Tell me why you think so?*" As a matter of fact, in the prototype system, when a student inputs a piece of evidence, the tutor almost always takes this action (because it's optimal, of course). Note, however, the exception at the ninth turn, where the ITS decided to perform the `JUSTIFICATION` action. This is because at that state (i.e., `(11111 :NIL)`), the expected utilities of the states where the student gets some new concept are greater than the ones where the student responds to the tutor's question. This is because `AGREE-WITH-JUSTIFICATION-2` puts more preference (.4) on the states where the student input some response, while `JUSTIFICATION` puts the same preference (.3) on the states where the student inputs some response or makes a support. If one wishes to implement the same behavior as described above with operator based planner, he/she might be forced to specify much more complicated conditional part (i.e., LHS of the operator) to state the same policy. That is, for example, the closer to a goal state the current state is, the less value there is for making students reflect on their answer.

Second, since the ITS looks up the policy table at every state, even if the student does not properly respond to its comment, the session proceeds without break, as long as student's response does not conflict with the conversation context (see a discontinuity at the third turn). So, the MDP based instructional planner is reactive in a sense that it always takes the best action in current situation.

5 Conclusion

Although the example domain used in the prototype system is a simple one, several distinctive features of MDP based decision theoretic planner for ITS are observed. The fact that each individual instructional action is defined as an operator in a state transition network explains why the decision theoretic planner is robust. Furthermore, since the optimal policy is determined by considering the rewards over all the states, the entire performance of the planner is rational in a sense that its decision is always globally optimal.

The challenges concern the reward function and a state expansion. Since they are both dominated by the number of factors used to describe the states, the more complicated the state descriptions, the more difficult to define the reward function, and the larger the state space. Indeed, even using our simple binary representation of student knowledge (1=knows the rule; 0=does not know the rule), the state space doubles with every new rule.

In this paper, the problem of faulty student model is not examined. The idea is that if one can define a probability distribution over the space of the student model, then an optimal policy at state i ($O(i)$) is determined as follows;

$$O(i) = \arg\max_{P} S_i^j \cdot P(j)$$

where S_i^j is a probability of student's actual knowledge state is at the state j when the student model says that he/she is in the state i.

References

Boutilier, C., Dean, T. & Hanks, S. (1999). Decision theoretic planning: Structural assimptions and computational leverage. *Journal of Artificial Intelligence Research*, *11*, 1–94.

Freedman, R. (1999). Atlas: A plan manager for mixed-initiative, multimodal dialogue. In *AAAI'99 Workshop on Mixed-Initiative Intelligence*. Orlando.

Goodman, B., Soller, A., Linton, F. & Gaimari, R. (1998). Encouraging student reflection and articulation using a learning companion. *Int. J. of Artificial Intelligence in Education*, *9*, 3–4.

MacMillan, S. A. & Sleeman, D. H. (1987). An architecture for a self-improving instructional planner for intelligent tutoring systems. *Computer Intelligence*, *3*, 17–27.

Murray, R. C. & VanLehn, K. (2000). DT Tutor: A decision-theoretic, dynamic approach for optimal selection of tutorial actions. In *Proceedings of ITS 2000*. Berlin: Springer-Verlag.

Peachey, D. R. & McCalla, G. I. (1986). Using planning techniques in intelligent tutoring systems. *Int. J. Man-Machine Studies*, *24*, 77–98.

Putnam, R. T. (1987). Structuring and adjusting content for students: A study of live and simulated tutoring of addition. *American Educational Research Journal*, *24(1)*, 13–48.

VanLehn, K. (1988). Student modeling. In M. C. Polson & J. J. Richardson (Eds.), *Foundations of Intelligent Tutoring Systems*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Vassileva, J. (1995). Reactive instructional planning to support intelligent teaching stragegies. In *Proc. of AI-ED* (pp. 334–342).