

A Decision-Theoretic Approach for Selecting Tutorial Discourse Actions

R. Charles Murray
Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA 15260
rmurray@pitt.edu

Kurt VanLehn
LRDC
University of Pittsburgh
Pittsburgh, PA 15260
vanlehn@pitt.edu

Jack Mostow
Project LISTEN
Carnegie Mellon University
Pittsburgh, PA 15213
mostow@cs.cmu.edu

Abstract

We propose a decision-theoretic architecture for selecting tutorial discourse actions. *DT Tutor*, an action selection engine which embodies our approach, uses a dynamic decision network to consider the tutor's objectives and uncertain beliefs in adapting to and managing the changing tutorial state. It predicts the effects of the tutor's discourse actions on the tutorial state, including the student's internal state, and appears to be unique in explicitly predicting the student's next action and its effect on the tutorial state. Based on the probabilities of predicted outcomes and their utilities, DT Tutor selects the tutorial action with maximum expected utility. We illustrate our approach with prototype applications for diverse domains: calculus problem-solving and elementary reading. Feasibility evaluations assess DT Tutor's ability to select optimal actions quickly enough to keep the student engaged.

Introduction

A tutoring system achieves many of its objectives through discourse actions intended to influence the student's internal state. For instance, a tutor might tell the student a fact with the effect of increasing the student's knowledge, thereby enabling the student to perform a problem-solving step. However, the tutor is inevitably uncertain about the student's internal state, as it is unobservable. Compounding the uncertainty, the student's state changes throughout the course of a tutoring session—that is the purpose of tutoring. To glean uncertain information about the student, a tutor must make inferences based on observable student actions and guided by the tutor's beliefs about the situation. The tutor is also likely to be concerned with observable attributes of the tutoring situation, or tutorial state, including the discourse between the tutor and the student and their progress at completing tutorial tasks (e.g., solving problems).

The tutor's actions depend not only on the tutorial state, but also on the tutor's objectives, such as increasing the student's knowledge, helping the student achieve a task, and bolstering the student's affective

state (Lepper et al., 1993). Objectives and priorities may vary by tutor and even for an individual tutor over time. Often, a tutor must strike a balance among multiple competing objectives (Lepper et al., 1993).

This paper describes *DT Tutor*, a decision-theoretic approach for selecting tutorial discourse actions that considers the tutor's uncertain beliefs and multiple objectives in adapting to and managing the changing tutorial state. Prototype applications for diverse domains, calculus problem-solving and elementary reading, illustrate the approach.

1 General Approach

1.1 Belief and Decision Networks

DT Tutor represents the tutor's uncertain beliefs in terms of probability using Bayesian belief networks. A belief network is a directed acyclic graph with *chance nodes* to represent beliefs about attributes and *arcs* between nodes to represent conditional dependence relationships among the beliefs. Beliefs are specified in terms of probability distributions. DT Tutor's chance nodes represent the tutor's beliefs about the tutorial state. For each node with incoming arcs, a conditional probability table specifies the probability distribution for that node conditioned on the possible states of its parents. For nodes without incoming arcs, prior probability distributions are specified.

Each node within a belief network represents an attribute whose value is fixed. For an attribute whose value may change over time (such as a tutorial state attribute), separate nodes can be used to represent each value. Dynamic belief networks do just that. For each time in which the values of attributes may change, a dynamic belief network creates a new *slice*. Each slice is of a set of chance nodes representing attributes at a specific point in time. For tutoring, slices can be chosen to represent the tutorial state after a tutor or student action, when attribute values are likely to change. Nodes may be connected to nodes within the same or earlier slices to represent the fact that an attribute's value may depend on (1) concurrent values of other attributes and (2) earlier values of the same and other attributes.

Decision theory extends probability theory to provide a normative theory of how a rational decision-maker should behave. Quantitative utility values express preferences among possible outcomes of decisions. To decide among alternative actions, the expected utility of each alternative is calculated by taking the sum of the utilities of all possible outcomes weighted by the probabilities of those outcomes occurring. Decision theory holds that a rational agent should choose the alternative with maximum expected utility. A belief network can be extended into a decision network (equivalently, an influence diagram) by adding decision and utility nodes along with appropriate arcs. For DT Tutor, decision nodes represent tutorial action alternatives and utility nodes represent the tutor's preferences among the possible outcomes.

A dynamic decision network (DDN) is like a dynamic belief network except that it also has decision and utility nodes to model decision-making in dynamic situations. The evolution of a DDN can be computed while keeping in memory at most two slices at a time (Huang et al., 1994).

1.2 General Architecture

The basis of our action selection engine is a DDN formed from dynamically created tutor action cycle networks (TACNs). A TACN consists of three slices, as illustrated in Figure 1. The tutorial state ($State_s$) within each slice is actually a sub-network representing the tutor's beliefs about the tutorial state at a particular point in time (slice)¹. The $T Act_1$ decision node represents the tutorial action decision. The $S Act_2$ chance node represents the student turn following the tutor's action, and the $Util_2$ utility node represents the utility of the resulting tutorial state.

Each TACN is used for a single cycle of tutorial action, where a cycle consists of deciding a tutorial action and carrying it out, observing the next student action, and updating the tutorial state based on these two actions. During the first phase (deciding upon a tutorial action), slice 0 represents the tutor's current beliefs about the tutorial state. Slice 1 represents the tutor's possible actions and predictions about their effects on the tutorial state. Slice 2 represents a prediction about the student's next action and its effect on the tutorial state. The DDN update algorithm calculates which action has maximum expected utility.

In the next phase of the cycle, the tutor executes that action and waits for the student response. When the student's action has been observed, the tutor updates the network based on the observed student action.

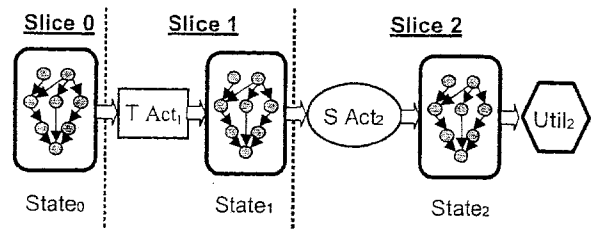


Figure 1. Tutor Action Cycle Network, overview

At this point, the posterior probabilities in $State_2$ represent the tutor's current beliefs. It is now time to select another tutor action, so another TACN is created and the DDN is rolled forward: Posterior probabilities from $State_2$ of the old TACN are copied as prior probabilities to $State_0$ of the new TACN, where they represent the tutor's current beliefs. The old TACN is discarded. The tutor is now ready to begin the next cycle by deciding which action to take next.

In principle, the tutor can look ahead any number of slices without waiting to observe student actions. The tutor simply predicts probability distributions for the next student action and the resulting $State_2$, rolls the DDN forward, predicts the tutor's next action and the following student action, and so on. Thus, the tutor can select an optimal sequence of tutorial actions for a fixed amount of look ahead. However, a large amount of look ahead is computationally expensive.

With this architecture, the tutor not only reacts to past student actions, but also anticipates student actions and their ramifications. Thus it can act to prevent errors and impasses before they occur, just as human tutors often do (Lepper et al., 1993).

2 Application Domains

2.1 Calculus Problem-Solving

The CTD (Calculus Tutor, Decision-Theoretic) prototype was developed for calculus related rates problems (Murray & VanLehn, 2000). This domain was chosen because Singley (1990) developed a tutoring system for it whose interface made most student problem-solving actions, including goal-setting actions, observable. The prototype presumed an extension to the interface to make all problem-solving actions observable. This made it easier to predict the student's next action, since CTD could observe all of the student's prior actions and only had to predict one action instead of combinations of multiple actions. However, predicting the next student action was still not trivial, since calculus related rates problems may have more than one solution path, the steps of which may be executed in multiple orders.

2.2 Project LISTEN's Reading Tutor

RTDT (Reading Tutor, Decision-Theoretic) is a proto-

¹For sub-network and node names, a numeric subscript refers to the slice number. A subscript of s refers to any appropriate slice.

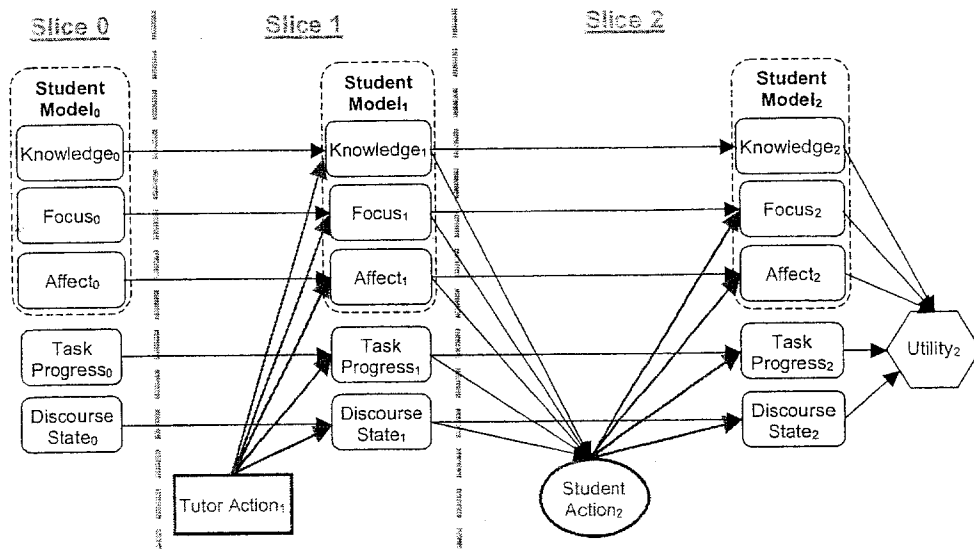


Figure 2. TACN architecture in more detail

type action selection engine for Project LISTEN's Reading Tutor, which helps children with reading skills as they read aloud (Mostow & Aist, 1999). The Reading Tutor has improved the reading of real students in classrooms (Mostow & Aist, in press). It displays one sentence at a time for the student to read, along with a simple animated persona that appears to actively watch and patiently listen. As the student reads, the Reading Tutor uses automated speech recognition to detect when the student may need help. Help is provided using both speech and graphical display actions. To avoid disrupting the flow of reading, the Reading Tutor ignores errors on 36 common function words (e.g., *a*, *the*) that are unlikely to affect comprehension. For the Reading Tutor's corpus, about one-third of the words in a sentence are non-function words, or *content* words.

Reading is a continuous activity in which student turns may consist of multiple reading actions, where each action is an attempt to read a word. Therefore, *RTDT* must predict and respond to multiple student actions per turn. This is in contrast to the episodic nature of the calculus problem solving domain, for which it was assumed that each student turn consists of a single action.

3 Tutor Action Cycle Networks in More Detail

3.1 TACN Components

Figure 2 provides a closer look at the major TACN components and their interrelationships. The *State*, representation in each slice consists of several sub-networks. These include the *Knowledge_s*, *Focus_s*, and *Affect_s* sub-networks which compose the student model, and the *Task Progress_s*, and *Discourse State_s* sub-

networks. Depending on the application, some sub-networks may not be required. For instance, *RTDT* does not model the student's affective state and so its TACNs do not include the *Affect_s* sub-networks. Selected sub-networks are described below.

Arcs between corresponding sub-networks in different time slices represent the stability of attributes over time. For instance, the student's knowledge in slice 1, *Knowledge₁*, is likely to be about the same as the student's knowledge in slice 0, *Knowledge₀*, except as influenced by the tutor's action, *Tutor Action₁*.

3.1.1 Tutor Action₁ Nodes

The purpose of the TACN is to compute the optimal alternative for *Tutor Action₁*, which may consist of one or more decision nodes. For *CTDT*, *Tutor Action₁* consists of two decision nodes, one to specify the *topic* of the tutor action, and another to specify the *action type*. The *topic* is the problem-related focus of the action, such as a problem step or related concept in the target domain. The *type* is the manner in which the topic is addressed, including *prompt*, *hint*, *teach*, *positive* or *negative feedback*, *do* (tell the student how to do a step) and *null* (no tutor action).

For *RTDT*, *Tutor Action₁* is currently a single decision node with multiple topic alternatives and a single action type – *hint* for each word in the sentence or the sentence as a whole – plus *null* (no tutor action).

3.1.2 Student Model Knowledge_s Sub-Network

This sub-network represents the tutor's beliefs about the student's domain knowledge. Each *Knowledge_s* node has possible values *known* and *unknown*. For *CTDT*, the structure of the *Knowledge_s* sub-network is based on a hierarchical dependency network called a

problem solution graph (Conati et al., 1997). Nodes in this graph structure represent the student's knowledge of (1) problem steps, and (2) rules licensing each step. Problem steps include the givens and every goal and fact along any path towards the solution. Arcs represent dependence between nodes. For instance, knowledge of a step depends on knowledge of both its prerequisite steps and the rule required to derive it.

For RTDT, $Knowledge_s$ includes $Know_Word_i_s$ nodes to represent the student's knowledge of how to read each content word i , and a $Know_Sentence_s$ node to represent the student's knowledge of the sentence.

In slice 1, each $Knowledge_1$ node is influenced by the tutor's action. For instance, a tutorial hint about a particular problem step or word increases the probability that the node corresponding to the knowledge element is *known*.

3.1.3 Student Model Focus, Sub-Network

The $Focus_s$ sub-network represents the student's focus of attention within the current tutorial task. For CTDT, this may be any problem step, so $Focus_s$ has the same problem solution graph structure as $Knowledge_s$. Steps for which prerequisites have been completed but that have not yet been completed themselves are most likely to be in focus. Nodes representing these steps have some distribution over the values *ready* and *in_focus*, where *in_focus* means that the step is in the student's focus of attention. Consistent with a human depth-first problem-solving bias (Newell & Simon, 1972), any such steps that are in the student's current solution path are most likely to be *in_focus*. The probability that an uncompleted step is *in_focus* attenuates with each passing time slice as other problem steps come into focus (Murray & VanLehn, 2000).

For RTDT, $Focus_s$ models the first content word in the student's focus of attention. $Focus_Word_i_s$ nodes for each content word i have possible values *in_focus* and *out_of_focus*. A bias that the student will continue reading with the next word in the sentence is modeled by the $Student_Action_2$ nodes (see below).

In slice 1, each $Focus_1$ node is influenced by the tutor's action. For instance, if the tutor hints about a problem step or word, the corresponding node is likely to be *in_focus*. In slice 2, the student action influences the tutor's beliefs about the student's focus of attention (in $Focus_2$). For instance, if the student experiences an impasse on a problem step or a word, the corresponding node is more likely to be *in_focus*.

3.1.4 Student Action₂ Nodes

DT Tutor appears to be unique in explicitly predicting the student's actions and their effect on the tutorial state. The $Student_Action_2$ nodes represent one or more actions taken on the student's turn. For CTDT, a single student action is assumed. This action is represented by

two nodes, one for the action *topic* and another for the action *type*. The action *topic* may be any problem step and the action *type* may be *correct*, *error*, *impasse*, or *null* (no student action).

For RTDT, the student turn may include multiple reading actions, where each action is an attempt to read a word. Each word i is represented by a $Word_i_2$ node with values *not_read*, *error*, or *correct*, modeling student turns ranging from no productive attempt (all words *not_read*), to all words read correctly (all words *correct*), to any combination in between. In addition, a $Sentence_2$ node models the student's reading of the sentence as a whole as either *fluent* or *disfluent*.

Both CTDT and RTDT probabilistically predict the next student action. For CTDT, $Focus_1$ influences the action *topic*. Given the action *topic*, whether the action *type* will be *correct*, *error* or *impasse* depends on the student's knowledge. Therefore, both the action *topic* and $Knowledge_1$ influence the action *type*.

For RTDT, influences on each $Word_i_2$ node from the corresponding $Focus_Word_i_1$ node probabilistically predict which word the student will attempt first. For any word that the student attempts, an influence from the corresponding $Know_Word_i_1$ node predicts whether the student's reading will be *correct* or in *error*. We assume that if a student reads one word correctly, she is most likely to attempt the next word, and so on, until she gets stuck or makes an error. Therefore, arcs from each node $Word_i_2$ to node $Word_{i+1}_2$, $i = \{1, 2, \dots, n-1\}$ for an n -word sentence, model the influence of reading word i correctly on the likelihood that the student will attempt word $i+1$.

3.1.5 Discourse State, Sub-Network

For CTDT, a $Coherence$ node represents the coherence of the tutor's action in response to the previous student action as either *coherent* or *incoherent*. For instance, *negative feedback* in response to a *correct* student action is *incoherent*. A $Relevance$ node models how well the tutor cooperates with the student's focus of attention by assessing the extent to which the same problem steps are *in_focus* before and after the tutor's action.

For RTDT, $Discourse_State_s$ is simply the number of discourse turns, counted as a measure of success at avoid spending too much time on a sentence

3.1.6 Utility₂ Nodes

$Utility_2$ consists of several utility nodes in a structured utility model representing tutor preferences regarding tutorial state outcomes. Total utility is a weighted sum of the utilities for each tutorial state component (e.g., student knowledge, student affect, task progress, etc.). The tutor's behavior can easily be modified by changing the utilities or their weights. For instance, by assigning a high weight to student knowledge and a low weight to task progress, the tutor focuses on increasing

student knowledge at the expense of task progress.

3.2 Implementation

With input from a problem solution graph (CTDT) or text (RTDT), DT Tutor creates a TACN with default values for prior and conditional probabilities and utilities. Default values are specified by parameter for easy modification. An optional file specifies any prior probability or utility values that differ from the defaults. After creating the initial TACN, DT Tutor recommends tutorial actions, accepts inputs representing tutor and student actions, updates the network, and adds new TACNs to the DDN as appropriate.

Both of DT Tutor's applications are prototypes for testing the viability and generality of the approach. CTDT does not yet have a graphical user interface, and RTDT has not been integrated with the Reading Tutor. Therefore, simulated student input was used for feasibility evaluations.

4 Technical Feasibility Evaluations

4.1 Response Time

One of the major challenges facing probabilistic systems for real-world domains is tractability (Cooper, 1990). Therefore, we tested whether applications based on DT Tutor can respond in real-time. All tests were performed on a 667-MHz Pentium III PC with 128-MB of RAM. Using Cooper's (1988) algorithm for decision network inference with belief network algorithms, we tested with three algorithms: an exact clustering algorithm (Huang & Darwiche, 1996) and two approximate algorithms, likelihood sampling and heuristic importance (Shachter & Peot, 1989), with 1,000 samples each. The times reported are the mean over 10 trials. The times for the approximate algorithms were extremely close, with neither holding an advantage in all cases, so they are not distinguished below.

For CTDT, only the approximate algorithms had reasonable response times for both problems tested: 1.5 seconds for a 5-step problem and 2.1 seconds for an 11-step problem.

For the Reading Tutor's corpus of readings, sentence length ranges from approximately 5 to 20 words as reading level progresses from kindergarten through fifth grade, with approximately two-thirds content words. We tested response times for preemptive help on sentences with 2 to 14 content words. Our response time goal was 0.5 seconds or less. For all three algorithms, response times for sentences with up to 7 content words were less than 0.5 seconds, ranging from 0.04 seconds for 2 content words to .49 seconds for 7 content words. Response times for the exact algorithm blew up starting at 10 content words with a time of 12.48 seconds. Response times for the approximate

algorithms remained promising (as explained below) for up to 12 content words, ranging from .59 seconds for 8 content words to 3.14 seconds for 12 content words. However, response times for even the approximate algorithms blew up at 13 content words with times of 23-26 seconds. Therefore, response time for preemptive help was satisfactory for students at lower reading levels, did not meet the goal for longer sentences (starting at 8 content words or approximately 12 words total), and was entirely unsatisfactory even with the approximate algorithms for the longest sentences (13-14 content words or approximately 20 words total).

For decision-making purposes, it is sufficient to correctly rank the optimal alternative. When only the rank of the optimal alternative was considered, the approximate algorithms were correct on every trial in the tests above. While this result cannot be guaranteed, it may make little practical difference if the alternative selected has an expected utility that is close to the maximum value. Moreover, many sampling algorithms have an anytime property that allows an approximate result to be obtained at any point in the computation (Cousins et al., 1993), so accuracy can continue to improve until a response is needed. For RTDT, response times for corrective feedback should generally be faster because RTDT does not consider helping with words that have already been read correctly. In any case, faster response times can be expected as computer hardware and probabilistic reasoning algorithms continue to improve. Therefore, the response times reported above for the approximate algorithms show promise that DT Tutor applications for real-world domains will be able to respond accurately enough within satisfactory response time. To handle the more challenging cases (such as the longest sentences faced by RTDT) in the near-term, application-specific adjustments may be required – e.g., abstraction in the knowledge representation within TACN components.

4.2 Action Selections

DT Tutor's decision-theoretic representation guarantees that its decisions will be optimal given the belief structure and objectives that it embodies. Nevertheless, the first step in evaluating a tutoring system is to verify that its behavior is consistent with strong intuitions about the pedagogical value of tutorial actions in specific situations, and this is a prerequisite for testing with human subjects. The space of network structures, probability and utility values, in combination with all possible student actions, is infinite, so the most we can do is sample from this space.

First, we used default parameters to initialize TACNs with intuitively plausible probability and utility values. Next, we simulated student action inputs while perturbing probability and utility values to probe dimensions of the situation space. Following are some

significant behaviors:

- Proactively attempts to prevent student failures
- Avoids helping unless the student appears to need it.
- Adapts to the student's apparent focus of attention.
- Tends to favor increasing student knowledge over direct help with task steps, since increasing the student's knowledge facilitates task progress anyway.
- Considers the student's affective state (CTDT).
- Prioritizes tutorial topics based on utility.

5 Related Work

Tutoring is a type of practical, mixed-initiative interaction. By mixed-initiative interaction, we mean collaborative interaction between participants characterized by dynamic negotiation of *initiative*, or control of the interaction (e.g., Allen, 1999; Horvitz, 1999b). Mixed-initiative interaction includes a variety of action types, including but not limited to conversation (Horvitz, 1999b). Tutoring involves *practical dialogue*, in which the participants pursue specific goals or tasks (Allen et al., 2001). Dialogue models can use any communication protocol and are independent of natural language (Allen, 1999). Besides natural language, computer-supported communication channels include graphical user interfaces and menu-based systems (Allen, 1999). In addition to research specific to tutoring, a good deal of research has addressed practical, mixed-initiative interaction more generally. This research can be characterized along a number of dimensions.

5.1 Decision-Theoretic Action Selection

One broad dimension on which DT Tutor has focused is decision-theoretic action selection, taking into account both the system's uncertainty about the user and the system's objectives. The sub-dimensions discussed below correspond to various methods used in decision-theoretic modeling. Like DT Tutor, the systems of Horvitz and colleagues (e.g., Horvitz et al., 1998; Horvitz & Paek, 1999; Paek & Horvitz, 2000) model the state of the collaboration, including the user's state, with connected sets of Bayesian models, and employ decision theory for optimal action-selection. The systems described in, e.g., (Horvitz & Paek, 1999; Paek & Horvitz, 2000) also use value-of-information to guide user queries and observation selection. DT Tutor does not query the user or select observations and so it does not consider value-of-information. Traum (e.g., 1999) computes the expected utility of various grounding alternatives, and Walker (1998) uses decision-theoretic methods to combine multiple measures of dialogue performance.

Many tutoring systems use Bayesian networks for student modeling (e.g., Conati et al., 1997), but almost all of them still make decisions heuristically. DT Tu-

tor's CTDT prototype follows (Conati et al., 1997) and others in basing its probabilistic Knowledge Networks on problem solution graphs. CAPIT (Mayo & Mitrovic, 2001, to appear) is apparently the only published tutoring system other than DT Tutor to use decision theory for action selection. CAPIT bases its decisions on a single objective and ignores the student's internal state to focus on observable variables. DT Tutor considers multiple objectives, including objectives related to a rich model of the student's internal state.

A few systems adapt probabilities online (e.g., Albrecht et al., 1997; Horvitz, 1999a; Mayo & Mitrovic, 2001, to appear), which DT Tutor does not do. Horvitz (1999a) also adapts "context-dependent" utilities online. Instead of adapting utilities according to context, DT Tutor adapts overall *expected* utility using fixed utilities based on the outcomes of chance nodes which model the underlying (changing) context. While DT Tutor's approach is computationally equivalent, it more accurately models the components of expected utility in the many situations where it is the context (represented by chance nodes) and not the set of preferences (represented by utility nodes) that is changing.

A number of probabilistic approaches have been tried to model the temporal evolution of the collaboration, including dynamic and single-stage network representations (e.g., Horvitz et al., 1998). (Albrecht et al., 1997) uses dynamic belief networks for user modeling. Very few if any systems other than DT Tutor use a dynamic decision network to combine consideration of uncertainty, objectives, and the changing state within a unified paradigm. DT Tutor also appears to be unique in proactively looking ahead to probabilistically predict the user's next action and its effect on the collaboration.

5.2 Attributes Modeled

Systems vary a great deal as to which attributes of the mixed-initiative interaction are modeled. For instance, the systems of (Allen et al., 2001; Allen et al., 2001, to appear; Horvitz & Paek, 1999; Paek & Horvitz, 2000) handle the interaction from end-to-end, from recognition of speech and other inputs to generation of speech and other outputs. Other systems (e.g., Albrecht et al., 1997; Horvitz et al., 1998) make do with "keyhole" observations of user actions obtained from external application program facilities. Collagen (Rich & Sidner, 1998) observes application program actions by both the user and an external system "agent," as well as user selections from a system-specified communication menu. DT Tutor leaves input recognition and detailed output generation to components external to the basic system architecture. DT Tutor's RTDT prototype is designed to combine DT Tutor's action selection engine with the graphical user interface and speech recognition and generation facilities of Project LISTEN's Reading Tutor for end-to-end handling of interaction

with the student.

While some systems focus on modeling detailed discourse phenomena such as grounding (e.g., Paek & Horvitz, 2000; Traum, 1999), DT Tutor focuses on interaction at the problem-solving or task level, including reasoning about the user's task-related goals and knowledge. Systems such as those described in (Allen et al., 2001, to appear; Conati et al., 1997; Horvitz & Paek, 1999; Rich & Sidner, 1998) also incorporate reasoning about interaction at the problem-solving level. The initial versions of the systems described in (Horvitz et al., 1998) avoid detailed modeling of the task state. Like DT Tutor, the systems of (Allen et al., 2001, to appear; Horvitz & Paek, 1999; Rich & Sidner, 1998) provide a domain-independent general architecture which can be tailored to specific task domains.

DT Tutor uses a detailed model of the user to customize its actions. Systems such as those described in (Conati et al., 1997; Horvitz et al., 1998; Langley et al., 1999) incorporate user models for similar reasons. The systems of (Paek & Horvitz, 2000; Traum, 1999) do without a user model for modeling grounding interactions, while (Allen et al., 2001) does without a user model by customizing responses based on discourse obligations and task demands.

A model of the user's focus of attention can be critical to providing timely and appropriate assistance (Horvitz, 1999b). DT Tutor models the user's focus of attention within a task domain using cues from both the discourse context and the domain structure, as do the systems of (Allen et al., 2001; Horvitz et al., 1998; Rich & Sidner, 1998). Some of the research by Horvitz and colleagues (e.g., Paek & Horvitz, 2000) uses a coarser definition of focus of attention, at the level of which task the user is attending to.

There is widespread agreement that multiple considerations are relevant to selecting system actions, including discourse characteristics (e.g., Allen, 1999; Allen et al., 2001, to appear; Rich & Sidner, 1998; Traum, 1999; Walker et al., 1998), task demands (Allen, 1999; Rich & Sidner, 1998), and multiple sources of evidence about user needs (Horvitz et al., 1998). DT Tutor uses decision theory to balance multiple considerations, as do the other decision-theoretic systems discussed above. The remaining research is mostly silent about how to prioritize considerations.

For practical discourse, the system may have its own goals apart from the needs of the user, such as accomplishing a particular task. In many systems, consideration of the system's goals is hard-wired and implicit (Allen et al., 2001). DT Tutor selects actions in service of its own goals and priorities (many of which concern the student's state), which are explicitly specified in terms of utility. With this explicit representation, DT Tutor's behavior can easily be modified by simply changing utilities or their weights. Of the research dis-

cussed above, only the work by Allen and colleagues (e.g. Allen et al., 2001) appears to explicitly consider the system's goals in addition to the user's goals.

6 Future Work

We are currently selecting a domain for fleshing DT Tutor out into a complete tutoring system, either by adding a user interface or by embedding it within an existing system. Our next major milestone will be testing its efficacy with students.

Efficiently obtaining more accurate probability and utility values is a priority. However, precise numbers may not always be necessary. For instance, diagnosis (say, of the student's knowledge) in Bayesian systems is often surprisingly insensitive to imprecision in specification of probabilities (Henrion et al., 1996). For a decision system, it is sufficient to correctly rank the optimal decision alternative, regardless of the exact expected utility. Moreover, if the actual expected utilities of two or more alternatives are very close, it may make little practical difference which one is selected.

Conclusions

This work has shown that a decision-theoretic approach can be used to select tutorial discourse actions that are optimal, given the tutor's beliefs and objectives. DT Tutor's architecture balances tradeoffs among multiple competing objectives and handles uncertainty about the changing tutorial state in a theoretically rigorous manner. Discourse actions are selected both for their direct effects on the tutorial state, including the student's internal state, and their indirect effects on the subsequent student turn and the resulting tutorial state. The rich tutorial state representation may include any number of attributes at various levels of detail, including the discourse state, task progress, and the student's knowledge, focus of attention, and affective state. Response time remains a challenge, but testing with approximate algorithms shows promise that applications for diverse real-world domains will be able to respond with satisfactory accuracy and speed. This approach might be applied more generally to other types of practical, mixed-initiative interaction.

Acknowledgments

This research was sponsored by the Cognitive Science Division of the Office of Naval Research under grant N00014-98-1-0467. Our decision-theoretic inference is performed by the SMILE reasoning engine for graphical probabilistic models contributed to the community by the Decision Systems Laboratory of the University of Pittsburgh (<http://www.sis.pitt.edu/~dsl>). We thank the reviewers for helpful suggestions, including calling our attention to some important related work.

References

- Albrecht, D. W., Zukerman, I., Nicholson, A. E., & Bud, A. (1997). Towards a Bayesian model for keyhole plan recognition in large domains. *6th International Conference on User Modeling*, pp. 365-376.
- Allen, J. F. (1999). Mixed-initiative interaction. *IEEE Intelligent Systems*, September/October, pp. 14-16.
- Allen, J., Ferguson, G., & Stent, A. (2001). An architecture for more realistic conversational systems. *Intelligent User Interfaces 2001 (IUI-01)*, 1-8.
- Allen, J. F., Byron, D. K., Dzikovska, M., Ferguson, G., Galescu, L., & Stent, A. (2001, to appear). Towards conversational human-computer interaction. *AI Magazine*.
- Conati, C., Gertner, A., VanLehn, K., & Druzdzel, M. (1997). On-line student modeling for coached problem solving using Bayesian networks. *6th International Conference on User Modeling*, pp. 231-242.
- Cooper, G. F. (1988). A method for using belief networks as influence diagrams. In *Workshop on Uncertainty in Artificial Intelligence*, pp. 55-63.
- Cooper, G. F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence* 42, 393-405.
- Cousins, S. B., Chen, W., & Frisse, M. E. (1993). A tutorial introduction to stochastic simulation algorithms for belief networks. *Artificial Intelligence in Medicine* 5, 315-340.
- Henrion, M., Pradhan, M., Del Favero, B., Huang, K., Provan, G., & O'Rorke, P. (1996). Why is diagnosis in belief networks insensitive to imprecision in probabilities? *Twelfth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 307-314.
- Horvitz, E. (1999a). Principles of mixed-initiative user interfaces. In *CHI '99, ACM SIGCHI Conference on Human Factors in Computing Systems*.
- Horvitz, E. (1999b). Uncertainty, action, and interaction: In pursuit of mixed-initiative computing. *IEEE Intelligent Systems*, September/October, pp. 17-20.
- Horvitz, E., Breese, J., Heckerman, D., Hovel, D., & Rommelse, K. (1998). The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Fourteenth Conference on Uncertainty in Artificial Intelligence*, pp. 256-265.
- Horvitz, E., & Paek, T. (1999). A computational architecture for conversation. In *Seventh International Conference on User Modeling*, pp. 201-210.
- Huang, C., & Darwiche, A. (1996). Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning* 15, 225-263.
- Huang, T., Koller, D., Malik, J., Ogasawara, G., Rao, B., Russell, S., & Weber, J. (1994). Automated symbolic traffic scene analysis using belief networks. In *Twelfth National Conference on Artificial Intelligence*, pp. 966-972.
- Langley, P., Thompson, C., Elio, R., & Haddadi, A. (1999). An adaptive conversational interface for destination advice. In *Third International Workshop on Cooperative Information Agents*.
- Lepper, M. R., Woolverton, M., Mumme, D. L., & Gurtner, J.-L. (1993). Motivational techniques of expert human tutors: Lessons for the design of computer-based tutors. *Computers as Cognitive Tools*, pp. 75-105. Lawrence Erlbaum Associates.
- Mayo, M., & Mitrovic, A. (2001, to appear). Optimising ITS behaviour with Bayesian networks and decision theory. *International Journal of Artificial Intelligence in Education* 12.
- Mostow, J., & Aist, G. (1999). Giving help and praise in a reading tutor with imperfect listening -- because automated speech recognition means never being able to say you're certain. *CALICO Journal* 16(3).
- Mostow, J., & Aist, G. (in press). Evaluating tutors that listen: An overview of Project LISTEN. In *Smart Machines in Education: The coming revolution in educational technology*. MIT/AAAI Press.
- Murray, R. C., & VanLehn, K. (2000). DT Tutor: A dynamic, decision-theoretic approach for optimal selection of tutorial actions. *Intelligent Tutoring Systems, 5th International Conference*, pp. 153-162.
- Newell, A., & Simon, H. A. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall, Inc.
- Paek, T., & Horvitz, E. (2000). Conversation as action under uncertainty. In *16th Conference on Uncertainty in Artificial Intelligence*.
- Rich, C., & Sidner, C. L. (1998). COLLAGEN: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction* 8(3/4).
- Shachter, R., & Peot, M. (1989). Simulation approaches to general probabilistic inference on belief networks. In *Fifth Annual Conference on Uncertainty in Artificial Intelligence*, Vol. 5, pp. 221-231.
- Singley, M. K. (1990). The reification of goal structures in a calculus tutor: Effects on problem solving performance. *Interactive Learning Environments* 1, 102-123.
- Traum, D. R. (1999). Computational models of grounding in collaborative systems. *Working Papers of the AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems*, 124-131.
- Walker, M. A., Litman, D. J., Kamm, C. A., & Abella, A. (1998). Evaluating spoken dialogue agents with PARADISE: Two case studies. *Computer Speech and Language* 12(3), 317-347.