In G. Gauthier, C. Frasson, & K. VanLehn (Ed.), *Intelligent Tutoring Systems, 5th International Conference, ITS 2000*, pp.153-162. New York: Springer.

DT Tutor: A Decision-Theoretic, Dynamic Approach for Optimal Selection of Tutorial Actions

R. Charles Murray* and Kurt VanLehn

Intelligent Systems Program & Learning Research and Development Center
University of Pittsburgh, Pittsburgh, PA 15260
{rmurray,vanlehn}@pitt.edu

Abstract. *DT Tutor* uses a decision-theoretic approach to select tutorial actions for coached problem solving that are optimal given the tutor's beliefs and objectives. It employs a model of learning to predict the possible outcomes of each action, weighs the utility of each outcome by the tutor's belief that it will occur, and selects the action with highest expected utility. For each tutor and student action, an updated student model is added to a dynamic decision network to reflect the changing student state. The tutor considers multiple objectives, including the student's problem-related knowledge, focus of attention, independence, and morale, as well as action relevance and dialog coherence. Evaluation in a calculus domain shows that DT Tutor can select rational and interesting tutorial actions for real-world-sized problems in satisfactory response time. The tutor does not yet have a suitable user interface, so it has not been evaluated with human students.

1 Introduction

Tutoring systems that coach students as they solve problems often emulate the turn taking observed in human tutorial dialog [7, 15]. Student turns usually consist of entering a solution step or asking for help. The tutor's main task can be seen as simply deciding what action to take on its turn. Tutorial actions include a variety of action types, including positive and negative feedback, hinting, and teaching. The tutor must also decide the action topic, such as a specific problem step or related concept. DT Tutor's task is to select the optimal *type* and *topic* for each tutorial action.

How to select optimal tutorial actions for coached problem solving has been an open question. A significant source of difficulty is that much of the useful information about the student is not directly observable. This information concerns both the student's cognitive and emotional state. Compounding the difficulty, the student's state changes over the course of a tutoring session.

Another complication is that just what constitutes optimal tutoring depends upon the tutorial objectives. A tutor's objectives normally include various student-centered goals and may also include dialog objectives and action type preferences. Furthermore, the tutor may have to balance multiple competing objectives.

DT Tutor uses decision-theoretic methods to select tutorial actions. The remainder of the Introduction describes the basis of our approach, DT Tutor's general architecture, and prior work. Subsequent sections describe DT Tutor in more detail, a preliminary evaluation, future work and conclusions.

* Research supported by ONR's Cognitive Science Division, grant number N0014-98-1-0467.

In G. Gauthier, C. Frasson, & K. VanLehn (Ed.), *Intelligent Tutoring Systems, 5th International Conference, ITS 2000*, pp.153-162. New York: Springer.

1.1 Belief and Decision Networks

Probability has long been the standard for modeling uncertainty in diverse scientific fields. Recent work with belief network (equivalently, Bayesian network) algorithms has made modeling complex domains using probabilistic representations more feasible. Unfortunately, belief network inference is still NP-hard in the worst case. However, many stochastic sampling algorithms have an *anytime* property that allows an approximate result to be obtained at any point in the computation [4].

DT Tutor represents the tutor's beliefs about the student's problem-related knowledge using a belief network obtained directly from a *problem solution graph*, a hierarchical dependency network representing solutions to a problem [2, 11]. Nodes in the graph represent (1) problem steps, and (2) domain rules licensing each step. Problem steps include the givens and every goal and fact along any path towards the solution. We currently model incorrect steps as errors. Arcs represent dependence between nodes. For instance, knowledge of a step depends on knowledge of both its antecedent steps and the rule required to derive it. In belief network form, the nodes represent the tutor's beliefs about problem-related elements of the student's cognitive state and arcs represent conditional dependence between the elements.

Nodes within a belief network represent variables whose values are fixed. However, a student's mental state and the problem solution state change over the course of a tutoring session. To represent variables that change over time, it is more accurate to use a separate node for each time. *Dynamic belief networks* (DBNs) do just that. For each time in which the values of variables may change, a new *slice* is created. Each slice consists of a set of nodes representing values at a specific point in time. Rather than fixed time intervals, slices can be chosen so that each corresponds to the student model after a student or tutor action. Nodes may be connected to nodes within the same or earlier slices to represent the fact that a variable's value may depend on concurrent values of other variables (*synchronic* influences) and on earlier values of the same and other variables (*diachronic* influences). The evolution of a DBN can be represented while keeping in memory at most two slices at a time [10].

Decision theory extends probability theory to provide a normative theory of how a rational decision-maker should behave [13]. Utilities are used to express preferences among possible future states of the world. To decide among alternative actions, the *expected utility* of each alternative is calculated by taking the sum of the utilities of all possible future states of the world that follow from that alternative, weighted by the probabilities of those states occurring. Decision theory holds that a rational agent should choose the alternative with the maximum expected utility. A belief network, which consists entirely of *chance* nodes, can be extended into a decision network (equivalently, an influence diagram) by adding *decision* and *utility* nodes along with appropriate arcs [13]. For tutoring systems, decision nodes could represent tutorial action alternatives, chance nodes could represent possible outcomes of the actions, and utility nodes could represent the tutor's preferences among the possible outcomes.

A *dynamic decision network* (DDN) is like a DBN except that it has decision and utility nodes in addition to chance nodes. DDNs model decisions for situations in which decisions, variables or preferences can change over time. Just as for DBNs, simple algorithms exist to represent the evolution of a DDN while keeping in memory at most two slices at a time [10].

In G. Gauthier, C. Frasson, & K. VanLehn (Ed.), *Intelligent Tutoring Systems, 5th International Conference, ITS 2000*, pp.153-162. New York: Springer.

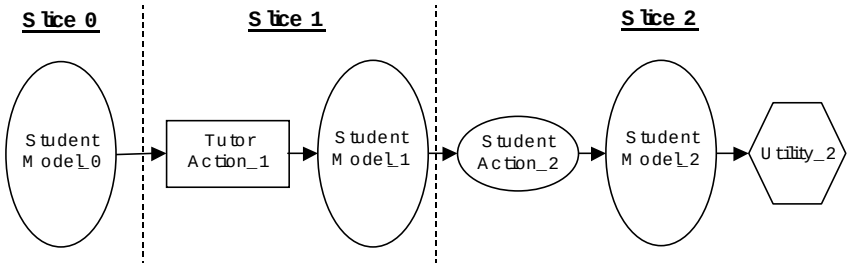

Fig. 1. Tutor action cycle network, high-level overview

1.2 General Architecture

Our basic approach is to use a DDN to implement most of the intelligent, non-user-interface part of DT Tutor. The DDN is formed from dynamically created decision networks. These networks are called *tutor action cycle networks* (TACNs) because they each represent 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 student model based on these two actions.

TACNs consist of three slices, as illustrated in Figure 1. A TACN is used both for deciding the tutor's action and for updating the student model. When deciding the tutor's action, slice 0 represents the tutor's beliefs about the student's current state. Slice 1 represents the tutor's possible actions and the influence of those actions on the tutor's beliefs about the student. Slice 2 represents the student's possible actions and the influence of those actions on the tutor's beliefs about the student. In other words, slice 0 represents the current student state and the other slices represent predictions about the student's state after the tutor's action and after the next student action. Slice 2 also includes the utility model since most of the outcomes in which the tutor might be interested concern the final effects of the tutor's current action.

The DDN update algorithm calculates the action with maximum expected utility. The tutor executes that action and waits for the student to respond. When the tutor has observed the student's action, the student model update phase begins.

The tutor clamps the student's action and updates the network. At this point, the posterior probabilities in *Student Model*$_2$[1] represent the tutor's current beliefs about the student. It is now time for another tutorial action selection, so another TACN is created. Posterior probabilities from *Student Model*$_2$ of the old TACN are copied as prior probabilities to *Student Model*$_0$ of the new TACN. The old TACN is discarded. The tutor is now ready to begin the next phase, deciding what action to take next.

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

---

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

In G. Gauthier, C. Frasson, & K. VanLehn (Ed.), *Intelligent Tutoring Systems, 5th International Conference, ITS 2000*, pp.153-162. New York: Springer.

1.3 Prior Work

Although probabilistic reasoning is become increasingly common in tutoring systems and AI in general, we believe this is the first application of a DDN to tutoring. Probabilistic reasoning is often used in student and user modeling. In particular, Bayesian networks are used in the student models of Andes [2], HYDRIVE [16] and other systems [12]. However, even with a probabilistic student model, other systems select tutorial actions using heuristics instead of decision-theoretic methods. Reye [19] has suggested the use of a decision-theoretic architecture for tutoring systems and the use of dynamic belief networks to model the student's knowledge [20, 21].

2 Detailed Solution

This section describes TACNs in more detail, along with their implementation to form DT Tutor's action selection engine.

2.1 Major TACN Components and Their Interrelationships

Figure 2 provides a closer look at a TACN. The student model includes components to represent the student's knowledge state (sub-network *Knowledge Network*$_n$), the student's problem completion status and focus of attention (sub-network *Relevance Network*$_n$), and the student's emotional state (nodes *Morale*$_n$ and *Independence*$_n$). Tutor and student actions are represented by two nodes each: one for the action *topic* (*Tutor/Student Topic*$_n$) and another for the action *type* (*Tutor/Student Type*$_n$). Tutorial action relevance and coherence are represented by the *Relevance*$_1$ and *Coherence*$_1$ nodes respectively. The utility model (*Utility*$_2$) represents the tutor's preferences.

In Figure 2, the Knowledge and Relevance Networks are shown as large rounded rectangles. Each arc into or out of these sub-networks actually represents multiple arcs to and from various sub-network nodes. For instance, there is a diachronic arc from each *Knowledge Network*$_0$ node to the corresponding *Knowledge Network*$_1$ node.

Student Knowledge Network. For each problem, a Knowledge Network is created from the problem solution graph to represent the tutor's beliefs about the student's problem-related knowledge.

Within the Knowledge Network, each node has possible values *known* and *unknown*. Rule nodes represent the tutor's belief about the student's knowledge of the corresponding rule. Problem step nodes represent the tutor's belief about the student's potential to know the corresponding step given the student's current rule knowledge. At the beginning of a problem, the nodes representing the givens and the problem goal are clamped to *known*, since these are given in the problem statement. The student's potential knowledge of the remaining problem steps depends upon the student's knowledge of the rules required to complete each problem step in turn.

Influences on Knowledge Network nodes include (1) synchronic influences to model the interdependence of the student's problem-related knowledge, and (2) diachronic influences between corresponding nodes in different slices to model the stability of the student's knowledge over time. The tutor can also influence *Knowledge*

In G. Gauthier, C. Frasson, & K. VanLehn (Ed.), *Intelligent Tutoring Systems, 5th International Conference, ITS 2000*, pp.153-162. New York: Springer.

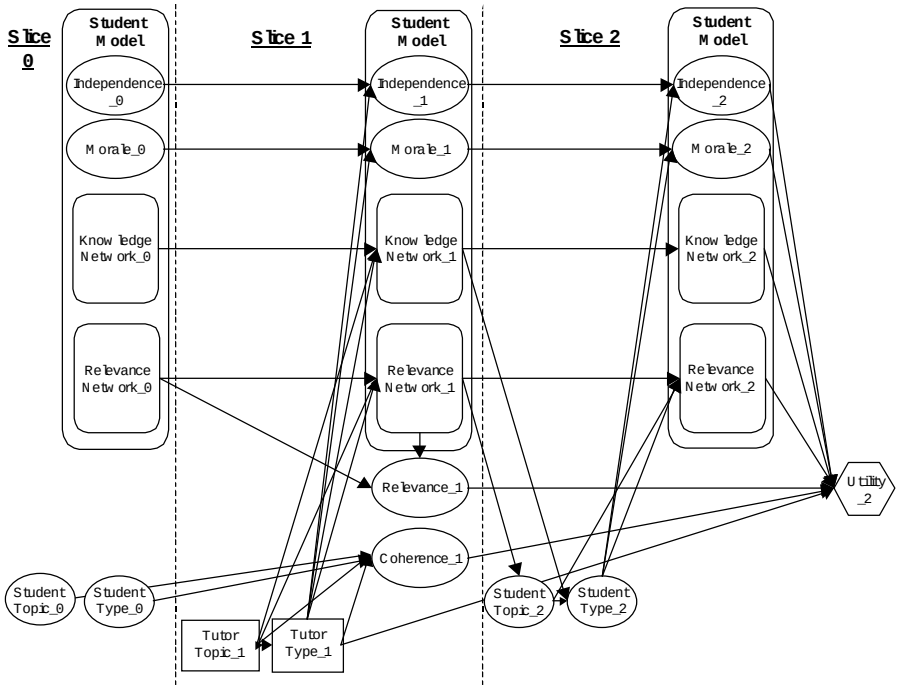

Fig. 2. Tutor action cycle network in more detail

Network$_1$ nodes directly, with an influence depending on the tutor action type. For instance, if the tutor teaches a domain rule, there is normally a greater probability that the rule will become *known* than if the tutor hints about it.

Student Relevance Network. Like the Knowledge Network, the Relevance Network is a belief network created from the problem solution graph. It represents the tutor's beliefs about the student's problem solving progress and problem-related focus of attention. The representation of the student's focus of attention is used to select relevant tutorial actions and to predict the topic of the student's next action.

Rule nodes have possible values *relevant* and *irrelevant*, where *relevant* means that the rule is part of the student's focus of attention. If the tutor addresses a rule directly, the rule becomes *relevant* with a probability dependent on the tutor action type. For instance, teaching a rule is more likely than hinting to make a rule *relevant*.

Problem step nodes have possible values *complete*, *not ready*, *ready*, and *relevant*. Completed step nodes have the value *complete*. If a step node is not *complete* and if it has any synchronic step node parents that are not *complete*, its value is *not ready*, meaning that it is not ready to be completed until its parent steps have been completed (but this does not preclude the student from completing it anyway – e.g., by guessing). Otherwise, a step node has some distribution over the values *ready* and *relevant*. Such nodes represent the frontier of the student's work within the problem space, possible next steps and thus potentially the focus of the student's attention. The value *ready* means that a step is ready to be completed since all of its parent steps have been

In G. Gauthier, C. Frasson, & K. VanLehn (Ed.), *Intelligent Tutoring Systems, 5th International Conference, ITS 2000*, pp.153-162. New York: Springer.

completed. The value *relevant* means that not only is the step ready to be completed, but it is also part of the student's focus of attention. For instance, a step is likely to be *relevant* if the tutor addresses it (e.g., with a *hint*) or if the student makes an error on it. Steps that are *relevant* at some point in time become a little less *relevant* with each passing time slice. This is to model *relevance aging*: steps that were *relevant* slowly become less *relevant* as the student moves on to other topics.

When there are multiple steps that could be *relevant* by virtue of being the next uncompleted step along a solution path, DT Tutor assumes a depth-first bias to decide how likely the various steps are to be part of the student's focus of attention: When applying a rule produces multiple *ready* steps, students usually prefer to pick one and complete work on it before starting work on another. Such a bias corresponds to a depth-first traversal of the problem solution graph and is consistent with both activation-based theories of human working memory [1] and observations of human problem solvers [18]. However, a depth-first bias is not absolute. At any given step, there is some probability that a student will not continue depth-first.

To model depth-first bias, when a step first becomes *ready* or *relevant* because the last of its outstanding parent steps has become *complete*, that step becomes *relevant* with high probability. This is because with a depth-first bias, having just completed the step's parent, the student is likely to continue working with the step itself. Relevance aging helps to model another aspect of depth-first bias: preferring to continue with more recently raised steps. When the student completes or abandons a portion of the solution path, steps that were recently highly *relevant* but that are still not *complete* have had less relevance aging than steps that were highly *relevant* in the more distant past, so the more recently raised steps remain more *relevant*.

Student Emotional State. Human tutors consider the student's emotional or motivational state in deciding how to respond [5, 14]. Concern for student morale is likely to be one reason why tutors tend to give negative feedback subtly, to play up student successes and downplay student failures, etc., while maximizing the student's feeling of independence is likely to be one reason why tutors tend not to intervene unless the student needs help, to minimize the significance of the tutor's help, etc. [15]. Such behaviors cannot be explained in terms of concern for the student's knowledge and the problem solving state alone.

The student's emotional state is modeled with the $Morale_n$ and $Independence_n$ nodes. Each of these nodes has possible values *level 0* through *level 4*, with higher levels representing greater morale or independence. Both tutor and student actions influence these nodes with an influence dependent on the action type. In addition, diachronic influences model the stability of the student's emotional state over time.

Tutor Action Nodes. The $Tutor\ Type_1$ alternatives include fairly fine distinctions to model some of the subtlety that human tutors exhibit when working with students. These alternatives include *prompt*, *hint*, *teach*, *positive feedback*, *negative feedback*, *do* (do a step for the student), and *null* (no tutor action).

$Tutor\ Topic_1$ can be any problem-related topic, so there is an alternative for each rule or step node in the problem solution graph. The value *null* is also supported to model (1) a tutor action with a type but no topic (e.g., general *positive feedback*), and (2) no tutor action at all, in which case $Tutor\ Type_1$ is *null* as well.

In G. Gauthier, C. Frasson, & K. VanLehn (Ed.), *Intelligent Tutoring Systems, 5th International Conference, ITS 2000*, pp.153-162. New York: Springer.

Student Action Nodes. First, the values of the student action nodes in slice 0 are simply the values of the student action nodes in slice 2 of the previous TACN, except for the very first TACN, in which they both have the value *null*.

Student Topic$_n$ can be any step in the problem solution graph. It can also be *null* to model either no student action at all (in which case *Student Type*$_n$ is *null* as well) or a student action with a *null* topic (e.g., a general *impasse* or an *error* which the tutor cannot interpret as an attempt at a particular step). *Student Topic*$_2$ is influenced by the relevance of the steps that are most likely to be the topic of the student's next action – i.e., by *Relevance Network*$_1$ step nodes that are *ready* or *relevant.*

Student Type$_n$ has possible values *correct, error, impasse*, and *null*. *Impasse* means that the student does not know what action to take – for instance, when the student asks for help. *Null* is used to model no student action. *Student Type*$_2$ is influenced both by the student action topic and by the student's knowledge of that topic – i.e., by *Student Topic*$_2$ and by the *Knowledge Network*$_1$ step nodes.

Utility Model. Node *Utility*$_2$ is actually a structured utility model consisting of several nodes to represent tutor preferences regarding the following outcomes:

1. Student rule knowledge in slice 2 (rule nodes in *Knowledge Network*$_2$)
2. Student problem step progress in slice 2 (step nodes in *Relevance Network*$_2$)
3. Student independence in slice 2 (*Independence*$_2$)
4. Student morale in slice 2 (*Morale*$_2$)
5. Tutor action type in slice 1 (*Tutor Type*$_1$)
6. Tutor action relevance in slice 1 (*Relevance*$_1$)
7. Tutor action coherence in slice 1 (*Coherence*$_1$)

2.2 Implementation

DT Tutor was implemented using software developed at the Decision Systems Laboratory, University of Pittsburgh: GeNIe, a development environment for graphical models, and SMILE©, a platform independent library of C++ classes for reasoning with graphical probabilistic models. From the problem solution graph structure, DT Tutor creates a TACN with default values for node outcomes, prior probabilities, conditional probabilities, and utilities. An optional file can be loaded to specify any prior probability or utility values that differ from the default values. After creating the initial TACN, DT Tutor recommends tutorial actions, accepts inputs representing actual tutor and student actions, updates the network, and adds new TACNs to the DDN as appropriate. We have not yet developed a suitable graphical interface for students, so a simple text interface was created for evaluation.

While DT Tutor will work with most any problem solution graph, for the initial implementation we selected the domain of calculus related rates problems for two reasons. First, the number of steps per problem is non-trivial without being too large, so results obtained should be generalizable to other real world domains. Second, Singley [23] developed an interface for this domain with the purpose of reifying goal structures. We assume an extension to Singley's interface that makes all problem solving actions observable. This makes it easier to determine the student's current location in the problem solution space and thus to model the student's current focus of attention and predict the student's next action.

In G. Gauthier, C. Frasson, & K. VanLehn (Ed.), *Intelligent Tutoring Systems, 5th International Conference, ITS 2000*, pp.153-162. New York: Springer.

3 Evaluation

The goals for evaluation were to evaluate whether DT Tutor's approach can be used to select actions within reasonable space and time that are not only optimal but that also correspond to some of the more interesting behaviors of human tutors.

3.1 Evaluate Tractability

One of the major challenges facing Bayesian models for real world domains is tractability in terms of both space and time. A number of measures were taken to reduce space requirements [see 17] which were then considered tractable since the tutor was able to successfully perform the tests described below.

It is important to provide real-time responses in order to keep the student engaged. With an early version of the Andes physics tutor [2], students tolerated response times of up to 40 seconds. However, considering both (1) the variety of domains for which ITSs might be constructed, and (2) ever-improving computer hardware and algorithms for evaluating probabilistic models, no exact response time requirement can be determined. Rather, it is more important to begin to evaluate how such systems will scale.

Test results are shown in Table 1. Both of the approximate algorithms using 1,000 samples returned responses for both problems well within the tolerated limit, as did the exact algorithm for the smaller of the two problems. Response times for the approximate algorithms grew linearly in the number of samples and in the number of nodes. The approximate algorithms using 10,000 samples and the exact algorithm on the larger problem did not return responses quickly enough, and in any case, faster response times are desirable. Fortunately, a number of speedups are feasible, as discussed in [17]. In addition, the *anytime* property of the approximate algorithms could be used to continually improve results until a response is required. For many applications, including this one, it is sufficient to correctly rank the optimal decision alternative. When only the rank of the optimal decision alternative was considered, the approximate algorithms using 1,000 samples were correct on every trial.

Table 1. Action selection response times

Algorithm	Response time mean (range)	
	Problem A[a]	Problem B[b]
Exact: Clustering [9]	108 (107-109)	11 (11-12)
Approximate:		
Likelihood Sampling [22]		
1,000 samples	12 (12-13)	8 (7-8)
10,000 samples	106 (104-110)	64 (62-66)
Heuristic Importance [22]		
1,000 samples	12 (12-13)	8 (7-8)
10,000 samples	104 (101-106)	64 (60-66)

Note. Response time is the number of seconds required to determine the optimal tutorial action. Mean and range are over 10 trials. All tests were performed on a 200MHz Pentium MMX PC with 64MB of RAM. The algorithms were tested with Cooper's [3] algorithm for decision network inference using belief network algorithms.

[a] 10-step problem, 185-node TACN. [b] 5-step problem, 123-node TACN.

In G. Gauthier, C. Frasson, & K. VanLehn (Ed.), *Intelligent Tutoring Systems, 5th International Conference, ITS 2000*, pp.153-162. New York: Springer.

3.2 Evaluate Tutorial Action Selections

DT Tutor's decision-theoretic representation guarantees that its decisions will be optimal given the beliefs and objectives it embodies. Therefore, besides a sanity check of the implementation, the purpose of this part of the evaluation was to find out whether DT Tutor's approach and choices about which outcomes and objectives to model were sufficient to endow the tutor with some of the more interesting capabilities of human tutors. While detailed results would be too lengthy to report here [see 17], testing showed that DT Tutor is indeed capable of selecting rational tutorial actions that correspond in interesting ways to the behavior of human tutors. Notable behaviors included the following:

- DT Tutor did not provide help when it believed the student did not need it. Human tutors foster their students' independence by letting them work autonomously [14].
- When the student was likely to need help, DT Tutor often intervened *before* the student could experience failure. Human tutors often provide help proactively rather than waiting for a student error or impasse [14, 15].
- As the student moved around the problem space, DT Tutor adapted to support the student's current line of reasoning, assuming a depth-first topic bias.
- All other things being equal, DT Tutor preferred to address rules rather than problem-specific steps. Effective human tutoring is correlated with teaching generalizations that go beyond the immediate problem-solving context [24].
- DT Tutor considered the effects of its actions on the student's emotional state as well as the student's knowledge state. Human tutors consider both as well [14].
- DT Tutor prioritized its actions based on current beliefs and objectives. Likewise, human tutors prioritize their actions based on the student's needs and tend not to waste time addressing topics that the student does not need to know [15].

4 Future Work and Conclusions

We still need to develop a graphical interface or embed DT Tutor's action selection engine within an existing ITS and evaluate it with human students. Efficiently obtaining more accurate probability and utility values would be beneficial as well. However, an encouraging result from prior research is that Bayesian systems are often surprisingly insensitive to imprecision in specification of numerical probabilities [8] and may be accurate enough to infer the correct *decision* even if some of their assumptions are violated [6], so that precise numbers may not always be necessary.

This research has shown that a decision-theoretic approach can indeed be used to select tutorial actions that are optimal, given the tutor's beliefs and objectives, for real-world-sized problems in satisfactory response time. The DDN representation handles uncertainty about the student in a theoretically rigorous manner, balances tradeoffs among multiple objectives, automatically adapts to changes in beliefs or objectives, and increases the accuracy of the information upon which the tutor's decisions are based. By modeling not only the student's problem-related knowledge but also the student's focus of attention and emotional state, DT Tutor can select actions that correspond to some of the more interesting behaviors of human tutors.

In G. Gauthier, C. Frasson, & K. VanLehn (Ed.), *Intelligent Tutoring Systems, 5th International Conference, ITS 2000*, pp.153-162. New York: Springer.

References

1. Anderson, J. R. (1993). *Rules of the Mind*. Lawrence Erlbaum Associates.
2. 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.
3. Cooper, G. F. (1988). A method for using belief networks as influence diagrams. *Workshop on Uncertainty in Artificial Intelligence*, pp. 55-63.
4. Cousins, S. B., Chen, W., & Frisse, M. E. (1993). A tutorial introduction to stochastic simulation algorithms for belief networks. *AI in Medicine 5*, pp. 315-340.
5. del Soldato, T., & du Boulay, B. (1995). Implementation of motivational tactics in tutoring systems. *Journal of Artificial Intelligence in Education 6*(*4*), pp. 337-378.
6. Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning 29*, pp. 103-130.
7. Graesser, A. C., Person, N. K., & Magliano, J. P. (1995). Collaborative dialogue patterns in naturalistic one-to-one tutoring. *Applied Cognitive Psychology 9*, pp. 495-522.
8. 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*.
9. Huang, C., & Darwiche, A. (1996). Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning 15*, pp. 225-263.
10. Huang, T., Koller, D., Malik, J., Ogasawara, G., Rao, B., Russell, S., & Weber, J. (1994). Automated symbolic traffic scene analysis using belief networks. *Twelfth National Conference on Artificial Intelligence*, pp. 966-972.
11. Huber, M. J., Durfee, E. H., & Wellman, M. P. (1994). The automated mapping of plans for plan recognition. *Tenth Conference on Uncertainty in Artificial Intelligence*, pp. 344-350.
12. Jameson, A. (1996). Numerical uncertainty management in user and student modeling: An overview of systems and issues. *User Modeling and User-Adapted Interaction 5*(*3-4*), pp. 103-251.
13. Keeney, R., & Raiffa, H. (1976). *Decisions with Multiple Objectives*. Wiley.
14. 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.
15. Merrill, D. C., Reiser, B. J., Merrill, S. K., & Landes, S. (1995). Tutoring: Guided learning by doing. *Cognition and Instruction 13*(*3*), pp. 315-372.
16. Mislevy, R. J., & Gitomer, D. H. (1996). The role of probability-based inference in an intelligent tutoring system. *User Modeling and User-Adapted Interaction 5*(*3-4*).
17. Murray, R. C. (1999). A dynamic, decision-theoretic model of tutorial action selection. Unpublished MS Thesis, University of Pittsburgh. http://www.isp.pitt.edu/~chas/
18. Newell, A., & Simon, H. A. (1972). *Human Problem Solving*. Prentice-Hall, Inc.
19. Reye, J. (1995). A goal-centred architecture for intelligent tutoring systems. *World Conference on Artificial Intelligence in Education*, pp. 307-314.
20. Reye, J. (1996). A belief net backbone for student modeling. *Intelligent Tutoring Systems, Third International Conference*, pp. 596-604.
21. Reye, J. (1998). Two-phase updating of student models based on dynamic belief networks. *Fourth International Conference on Intelligent Tutoring Systems*, pp. 274-283.
22. Shachter, R. D., & Peot, M. A. (1990). Simulation approaches to general probabilistic inference on belief networks. *Uncertainty in Artificial Intelligence*, pp. 221-231.
23. Singley, M. K. (1990). The reification of goal structures in a calculus tutor: Effects on problem solving performance. *Interactive Learning Environments 1*, pp. 102-123.
24. VanLehn, K., Siler, S., Murray, C., Yamauchi, T., & Baggett, W. B. (in press). Human tutoring: Why do only some events cause learning? *Cognition and Instruction*.