International Journal of Artificial Intelligence in Education (2001), 12, 124-153

Optimising ITS Behaviour with Bayesian Networks and Decision Theory

Michael Mayo and Antonija Mitrovic *Department of Computer Science, University of Canterbury, Private Bag 4800, Christchurch, New Zealand,*
E-mail:{mmayo, tanja}@cosc.canterbury.ac.nz

Abstract. We propose and demonstrate a methodology for building tractable normative intelligent tutoring systems (ITSs). A normative ITS uses a Bayesian network for long-term student modelling and decision theory to select the next tutorial action. Because normative theories are a general framework for rational behaviour, they can be used to both define and apply learning theories in a rational, and therefore optimal, way. This contrasts to the more traditional approach of using an ad-hoc scheme to implement the learning theory. A key step of the methodology is the induction and the continual adaptation of the Bayesian network student model from student performance data, a step that is distinct from other recent Bayesian net approaches in which the network structure and probabilities are either chosen beforehand by an expert, or by efficiency considerations. The methodology is demonstrated by a description and evaluation of CAPIT, a normative constraint-based tutor for English capitalisation and punctuation. Our evaluation results show that a class using the full normative version of CAPIT learned the domain rules at a faster rate than the class that used a non-normative version of the same system.

INTRODUCTION

Intelligent tutors must operate with incomplete and usually highly uncertain information about their students. Knowledge about the student's current state (the student model) is necessary for assessment, and more importantly, adaptive pedagogical action selection (PAS). Frequently however, the student's interaction time will be insufficient for an accurate student model to be inferred. Even if the student was interacting for the requisite time, the student's state is likely to be changing so rapidly (and there are so many other external influences) that the student model is never likely to be complete or totally correct. To compensate for this dearth of valid information, intelligent tutors should be equipped with strong methods for handling uncertainty. Such methods should ideally be provably optimal, i.e. given observations about the student, the method should be guaranteed to perform optimal PAS.

No guarantees of optimality are made by other methods, like those developed by Artificial Intelligence (AI) scientists. In fact, even for perfectly certain student models, these methods are not provably optimal because the very mechanisms of their reasoning (e.g. production rules) are open to incompleteness and inconsistencies.

To overcome this potential for sub-optimality, we have investigated general theories of rationality (known as normative theories) and applied these to the design of an intelligent tutor. This approach has two advantages. Firstly, such theories are not usually the product of AI alone, but the product of a collaboration of scientists from many different fields over many years. Therefore, they are likely to be more widely tested and accepted than the typical AI theory. Secondly, the entrenchment of the intelligent tutor in a theory of rationality means that its behaviour will be guided by general principles of rational behaviour. This implies optimality.

We propose statistical decision theory (Savage, 1954), encompassing Bayesian probability theory (Bayes, 1763), as a particular theory of rationality suitable for application to intelligent tutoring systems. It is only recently that efficient and effective structures and algorithms for Bayesian reasoning, known as Bayesian networks (Pearl, 1988), have become available.

Note that normative theories are not domain specific, so they do not specify *what* is being reasoned about. In an intelligent tutor, this is the task of the learning theory. The main advantage that normative theories confer is that the learning theory can be defined within a rational framework. In turn, this means that the learning theory is guaranteed to be optimally applied to the student, with respect to the chosen normative theories. This compares favourably to the architecture of the traditional intelligent tutor in which the learning theory is defined using a less rigorous scheme (e.g. heuristic rules) that lack optimality guarantees.

In this paper, we review Bayesian and decision-theoretic approaches in existing intelligent tutors, and propose a number of desirable features that a decision-theoretic tutor should have.

We then define a general methodology for the development of decision-theoretic PAS strategies for intelligent tutors incorporating these desirable features. The methodology emphasises the collection of real-world data for evaluating and comparing different Bayesian network specifications. This "data-centric" approach contrasts to existing approaches to Bayesian network design, such as the "expert-centric" approach whereby a domain-expert directly or indirectly specifies the structure and maybe also the probabilities of the network, as in ANDES' Assessor network (Conati et. al., 1997; Gertner & VanLehn, 2000); and the "efficiency-centric" approach where the network is pre-specified to some degree to optimise either the specification size (e.g. Millán, 2000) or the efficiency of evaluation (e.g. Collins et. al., 1996; Reye, 1998), and the domain knowledge is "fitted" to this limited specification.

The proposed methodology is applied to the development of optimal PAS strategies for CAPIT (Capitalisation And Punctuation Intelligent Tutor). CAPIT is a constraint-based tutor that teaches the basic rules of English punctuation and capitalisation to 8-10 year old schoolchildren (Mayo et. al., 2000). The two decision-theoretic PAS strategies we have developed using this methodology are problem selection and error message selection. The strategies have been evaluated in the classroom and compared to randomised versions of the same strategies.

DECISION THEORY AND BAYESIAN NETWORKS

Decision theory and Bayesian probability theory are both instances of normative theories. A normative system encompasses not only a set of rules, but also the set of logical consequences of those rules (Gärdenfors, 1989). Therefore they can be considered logically complete and consistent. Under the assumption that a rational agent will act logically, normative systems can be thought of as prescriptive models for rational behaviour. This is in direct contrast to descriptive models that attempt to describe either the behaviour of an individual (such as an expert or a teacher) or a group of individuals (e.g. a psychological theory derived from observations), which may be subject to logical inconsistencies or incompleteness.

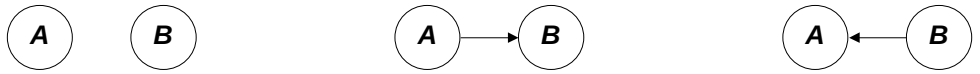
Bayesian probability theory is a set of rules for updating beliefs in an uncertain world. Subjective belief in a proposition such as "variable X is in state x" is represented by the statement $p(X = x) = r$, where $r = 1$ implies that the proposition is certainly true while $r = 0$ implies certain falsehood. A value of r between 1 and 0 indicates the degree of uncertainty between the certain extremes.

To update its beliefs, a Bayesian agent needs to maintain a model of the relationships between its uncertain propositions about the world. Two propositions A and B, for example, are mutually independent if a change in the agent's belief about one proposition does not influence its belief in the other proposition. On the other hand, A and B are dependent if a change in belief about one affects the agent's belief in the other. Dependence is represented by a conditional probability statement such as $p(B|A)$ that defines the agent's "posterior" belief in B given all the possible values of A. It is important to note that this relationship is reversible; given $p(B|A)$, we can always calculate $p(A|B)$ using Bayes' rule (Bayes, 1763):

$$p(A \mid B) = \frac{p(B \mid A)\,p(A)}{p(B)}$$

Once the relationship between the variables is determined, the joint probability distribution can be calculated. The joint probability distribution defines a table of probabilities, one entry for each different combination of values that the variables can jointly take. For example, if *A* and *B* are binary variables, then the joint probability distribution is referred to as *p(A,B)* and consists of four different probability entries. The sum of all the entries must be 1.

Sometimes it is convenient to represent variables and their dependencies as a directed graph, and this notation is called a Bayesian network. Figure 1 illustrates Bayesian networks for all the possible relationships between two variables.



(a) $p(A,B)=p(A)p(B)$ (b) $p(A,B)=p(A)p(B|A)$ (c) $p(A,B)=p(A|B)p(B)$

Figure 1. Graphical notation depicting (a) two mutually independent variables *A* and *B*, (b) two related variables *A* and *B*, and (c) the same two related variables but with the direction of the arc reversed.

Bayesian networks can be used to model the relationships between observed student actions, student internal states, and outcomes. Figure 2 depicts a simple example of this for illustrative purposes. Note that both *Read Textbook* and *Watched Video* are set with certainty to values `YES` and `NO`. This is called instantiation, and implies that these variables have been observed. When an uninstantiated node is queried, its probability distribution must be updated to incorporate all the currently instantiated nodes in the network. Lauritzen and Spiegelhalter (1988) describe an efficient and effective updating algorithm in common use.

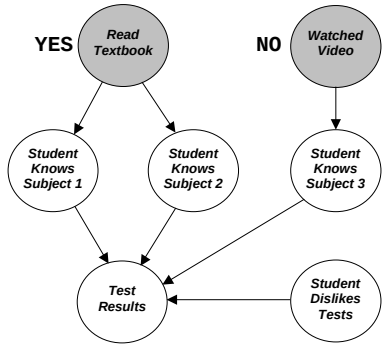


Figure 2. A hypothetical Bayesian network for predicting the performance of a student on a test.

Although the direction of any arc can be reversed, for practical purposes a Bayesian network cannot have any directed cycles. Furthermore, while it is a common convention that the arc directionality corresponds to the direction of causality, it is possible to apply other semantics to arc directionality. For example, Collins et. al. (1996) interpret arcs as emanating from topic to subtopic.

Whereas Bayesian networks are used to update beliefs from initial beliefs and observations, decision theory is a rational means of optimising behaviour by "fusing" uncertain beliefs with preferences. Suppose the agent is faced with the problem of selecting a single action d_i from a set of possible actions d_1, d_2…, d_n. If x is a possible outcome of d_i, then decision theory requires the agent to specify a real-valued preference $U(x, d_i)$ for each possible combination of x and d_i. This is called the utility function. The agent must also be able to estimate the probability of x should it opt for d_i, a value that can be determined from its Bayesian network. The expected utility of d_i is defined, therefore, as the probability-weighted sum of the utilities of each possible outcome less the cost of the action:

$$E[U(x,d_i)] = \left(\sum_x p(x \mid d_i) U(x,d_i) \right) - cost(d_i) \qquad (1)$$

The principle of maximising expected utility says that the agent should select the action d_i that maximises Equation 1.

It is also relevant to mention the foundations of decision theory. Bayesian probability was introduced over 200 years ago, and decision theory was first proposed in 1954 (Savage, 1954). There has been ample time, therefore, for these models to be widely tested. The fact that they are now accepted and applied in a variety of fields is a testament to their rigorous foundations. Intelligent tutors built on these fundamentals, therefore, will be widely accepted, a vision espoused by Everson (1995).

One reason for the neglect of these theories in Artificial Intelligence and related fields, however, was the problem of tractability (Jensen, Lauritzen et al., 1990). In its naïve form, Bayesian probability theory and decision theory are intractable. This led to the development of other schemes, such as fuzzy sets (Zadeh, 1983) and Dempster-Shafer theory (Shafer, 1986), which are not as general as normative theories, but are highly tractable (Horvitz et al., 1988). Since then, however, recent advances in the tractability of the normative reasoning have been made and researchers in these areas are showing renewed interest in normative models.

A noteworthy property of Bayesian networks is that both prior/expert knowledge and data can be seamlessly integrated within a single network. For example, an expert can specify some or all of a Bayesian network, data can be used to learn the rest of it, and then the expert can "fine-tune" the final version. This property is not typical of other representations such as neural networks, and is a significant, natural property of Bayesian networks that will be demonstrated in this paper.

For more technical details, the interested reader is referred to a number of general tutorials on Bayesian networks (D'Ambrosio, 1999; Cowell, 1999). Mislevy & Gitomer (1996) introduce Bayesian networks in the context of intelligent tutors. The learning of Bayesian networks from data is an important component in this paper, and suitable tutorials are provided by Heckerman (1999) and Krause (1998). Decision theory and AI are introduced by Horvitz et. al. (1988) and Russell & Norvig (1995, Ch. 16-17).

NORMATIVE TECHNIQUES IN EXISTING TUTORS

A number of recent intelligent tutors have been proposed with Bayesian network student model. In this section, we review the different approaches to Bayesian student modelling, and then we discuss some of the different applications of the student model to PAS.

Bayesian Network Student Modeling

It is possible to classify Bayesian network student models into three different groups, according to the technique by which they were constructed. *Expert-centric* student models are unrestricted products of domain analysis. That is, an expert specifies either directly or indirectly the complete structure and conditional probabilities of the Bayesian student model, in a manner similar to that with which expert systems are produced. This is the general approach of ANDES (Gertner & VanLehn 2000; Gertner et. al., 1998; Gertner, 1998; Conati et. al., 1997), HYDRIVE (Miselvy & Gitomer, 1996), DT-Tutor (Murry & VanLehn, 2000), and the Bayesian domain model of ADELE (Ganeshan et. al., 2000). One possible disadvantage of this approach is that the resulting models may include so many variables that it becomes infeasible to evaluate the network effectively on-line. For example, tractability testing was an important issue in the initial evaluation of DT-Tutor. *Efficiency-centric* models, on the other hand, work the other way: the model is partially specified or restricted in some way, and domain knowledge is "fitted" to the model. The restrictions are generally chosen to maximise some aspect of efficiency, such as the amount of numeric specification required and/or the evaluation time. This is the methodology of Reye (1998), Murray (1998), Collins et al (1996), Mayo & Mitrovic

(2000), and to a degree, Millán et. al. (2000). In general, restrictions to increase efficiency can introduce incorrect simplifying assumptions about the domain. Finally, the *data-centric* model is a new class of Bayesian student model, introduced and implemented in this paper, in which the structure and conditional probabilities of the network are learned primarily from data. This class of student model dispenses with attempting to model unobserved student states, such as their domain mastery, and instead concentrates to modelling the relationships between observed variables to predict student performance. MANIC (Stern et. al., 1999) is the closest existing system the authors could find to the data-centric approach, but it learns only the probabilities and not the structure of the network, and is therefore more efficiency-centric than data-centric. Work in this area is also described by Beck & Woolf (2000), but Bayesian networks are not used. Figure 3 shows how existing Bayesian network student models fit this classification.

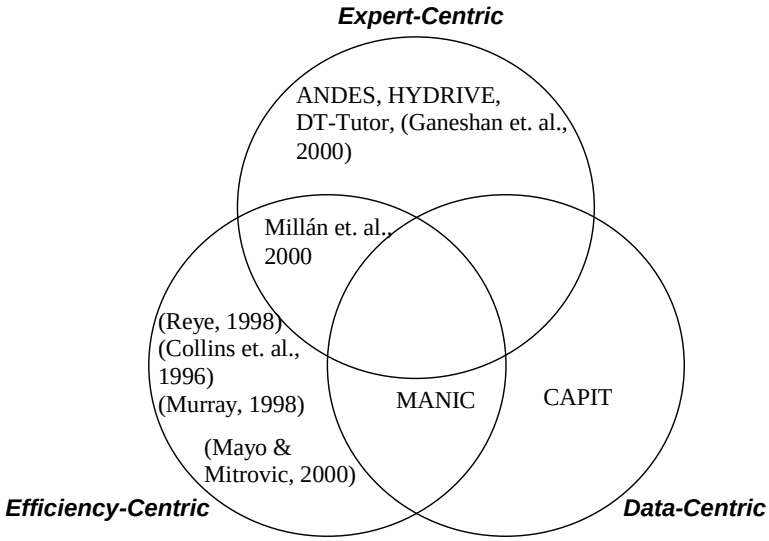


Figure 3. A classification of Bayesian network student models.

Expert-Centric

ANDES (Gertner & VanLehn 2000; Gertner et. al., 1998; Gertner, 1998; Conati et. al., 1997), HYDRIVE (Miselvy & Gitomer, 1996) and DT-Tutor (Murray & VanLehn, 2000), are examples of tutors with large Bayesian networks with structures mostly engineered from complex domain analysis. To match the domains as closely as possible, their networks are not structurally restricted in any way. However, they all have a high proportion of variables representing unobserved, internal student states. A major hurdle for these systems, then, is how conditional probabilities can be elicited or defined for these variables in the absence of data.

ANDES' solution is to use "coarse-grained" conditional probabilities definitions such as noisy-OR and noisy-AND. A noisy-OR variable has a high probability of being true only if at least one of its parents is true, and similarly for noisy-AND variables. In practice, restricting conditional probabilities to noisy-ORs and noisy-ANDs significantly reduces the number of required probabilities and makes the modelling of unobserved variables much simpler because only the structure and node type (noisy-AND or noisy-OR) needs to be specified.

In HYDRIVE, the conditional probabilities are defined subjectively in a "fuzzy-like" fashion. For example, a student's *Strategic Knowledge* takes the vague linguistic values `expert`, `good`, `okay` and `weak`. Tutorial actions and observations of student behaviour modify the probability distribution over these values via conditional probabilities, which were elicited from domain experts.

Finally, DT-Tutor is a generalised domain-independent architecture for student modeling and PAS. Like ANDES and HYDRIVE, it models the student's knowledge, but it goes much further and attempts to model other hidden states such as the student's morale, independence, and focus of attention. A preliminary version of this system has been constructed but no details have been given as yet to how the conditional probabilities will be obtained.

Models that largely represent unobserved, internal student states suffer a major disadvantage: the model structure and/or parameters cannot be adapted on-line to the student. To illustrate, consider a hypothetical very simple Bayesian network with discrete variables, *Observations* and *Student State*. Suppose that the system maintains the conditional probability *p(Student State|Observations)* for computing the posterior probability distribution over the hidden *Student State* from the observable variable *Observations*. This, in a highly abstract form, is how the student models in ANDES and HYDRIVE operate. Now, consider how this model can be adapted to the student. There are two different approaches. The first is to observe the student and instantiate *Observations*, and then update the value of *Student State* from this. This is the standard way in which Bayesian networks are used, but is means that *p(Student State|Observations)* will remain static and that the previous value of *Observations* will be lost. An alternative approach is based on machine learning, and involves modifying *p(Student State|Observations)* itself. If a particular value of *Observations* leads to a particular *Student State*, then *p(Student State|Observations)* is altered to increment slightly the probability that the same *Student State* will be observed again when the same or similar *Observations* are made again in the future.

However, this second approach relies on *Student State* being an observable variable, something that it is not in our simple model. This is an important reason for advocating models that eliminate hidden variables: they are simply more adaptable. Consider an equally simple model defining the relationship between two observable variables, *Observations* and *Next-Observations*. Because the variables are both observable, a conditional probability such as *p(Next-Observations|Observations)* becomes amenable to machine learning, and therefore the system is more adaptable.

Efficiency-Centric

An approach to student modelling using dynamic Bayesian networks (DBNs, Russell & Norvig, 1995, Ch. 17) has been proposed by Reye (1998). Reye's model is a generalisation of the student model used in the ACT Programming Languages Tutor (Corbett & Anderson, 1992; Corbett & Bhatnagar, 1997), and a similar approach was used in the student model of SQL-Tutor (Mayo & Mitrovic, 2000). The idea is to model the student's mastery of a knowledge item over time. The tutor's current belief that the student has mastered the item (M_t) depends on its previous belief (M_{t-1}), the outcome of the student's last attempt at the item (O_{t-1}), and the pedagogical response of the tutor to the last attempt (A_{t-1}). Using dynamic Bayesian networks, not only can the tutor's current beliefs be determined, but also its future beliefs at time *t+1* or beyond, although this is likely to be much more uncertain. This model is depicted in Figure 4 for a single knowledge item.

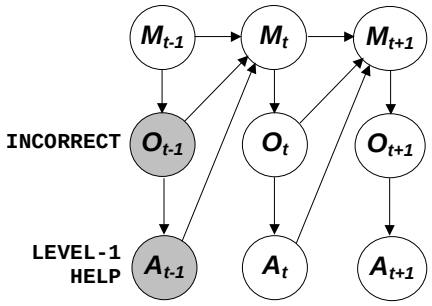


Figure 4. A DBN modelling the mastery of the student on a single knowledge item equivalent Reye's approach. At time *t-1*, the student failed an attempt at the item and so the tutor provided remedial help.

One problem with this approach is that the complexity-reducing assumption that mastery of a knowledge item is probabilistically independent of the mastery of any other items is unrealistic. Suppose, for example, that the knowledge items are "high level" such as concepts or topics. Then we would expect the mastery of some items to be dependent on the mastery of

items that are pre- and co-requisites. This is a basic assumption of many systems and the rationale behind many approaches to course sequencing, e.g. Brusilovsky (2000). Alternatively, the knowledge items could be "low level" such as constraints (Ohlsson, 1994; Mitrovic & Ohlsson, 1999; Mayo et. al., 2000). Clearly, we would expect many dependencies between constraint mastery based on factors such as syntactic and/or semantic relatedness. We demonstrate later that in the punctuation domain, a model with dependencies between items makes better predictions of student performance than a simpler model similar to Figure 4.

There are Bayesian student models that allow some dependencies to be expressed whilst remaining efficiency-centric. They are the singly-connected hierarchical structures described by Murray (1998), Collins (1996), and Stern et. al. (1999). A singly-connected network has the property that for every pair of nodes in the network, there is one and only one path between the nodes. Bayesian networks with a singly-connected topology evaluate in linear time (Pearl, 1988; Murray, 1999), and while they can express dependence between knowledge items, the singly-connected assumption means that certain types of dependence (namely, undirected loops) cannot be represented. This is clearly a strong restriction, because all of the expert-centric models described above contain undirected loops in their Bayesian networks.

The problems of single-connectedness are illustrated by MANIC (Stern et. al., 1999), which attempts to learn the probabilities (but not the structure) of its hierarchy from observations of the student. MANIC's hierarchical structure explicitly assumes that its variables are independent of each other given their mutual parent variable. Unfortunately, the data acquired from students directly contradicted this and Stern et. al. were forced to compensate by introducing several ad-hoc "fixes" to the network, such as "merging" dependent nodes and deleting irrelevant nodes. This jeopardised its normative status. A clear solution to this problem would have been to drop the restriction that the network was a hierarchy, although this would have led to a more complex model and the necessity for more complex learning algorithms.

Interestingly, Millán et. al. (2000) recently proposed an architecture that is to a degree both expert- and efficiency-centric. Their Bayesian network is selected to optimise the amount of numeric specification required, and to achieve this, the directionality of the arcs between groups of variables is fixed. The variables are also limited to binary states with specific semantics. However, the topology of the network does not have to be singly-connected which makes it quite flexible.

Data-Centric

This is the approach whereby both the structure and conditional probabilities of the network are learned from data collected from real-world evaluations of the tutor. There are a number of benefits of this approach. Firstly, because the model is induced from actual data, its predictive performance can easily be evaluated by testing the network on data that was not used to train it. Secondly, data-centric models can be expected to be much smaller than the typical expert-centric model because the latter represents both observed and hidden variables, while the former models only observable variables.

This is not to say that data-centric models may not contain hidden variables. If many observable variables are "compressed" in some way into a single hidden variable, then the resulting student model with hidden variables will in fact be smaller than the original model without hidden variables. However, there are some difficult issues to deal with. For example, how is the compression to be performed and more importantly, will the resulting hidden variables be consistent with the original observable variables? In a Bayesian system, it follows that probabilistic inference should be used to deduce the probability distribution over the new hidden variables from the original observable variables. If this is not done, the new compressed model will be probabilistically inconsistent with the original model. However, the additional computation required to maintain this consistency may well offset the space-saving that the compression afforded, rendering the whole process futile. On the other hand, there are theoretically sound methods of compressing Bayesian networks by introducing hidden variables (Heckerman, 1999). The advantage of adding hidden variables to a Bayesian network is that, if the hidden variables are defined carefully, then the number of arcs in the network can be

reduced as a result, without a significant corresponding decrease in the accuracy of the network. If such an algorithm were applied to a Bayesian network student model however, there is no guarantee that the hidden variables will be semantically meaningful (i.e., they might not correspond to states such as concept mastery).

Pedagogical Action Selection

Given a Bayesian student model, the next issue is how to use the model to optimise the pedagogical actions of the intelligent tutor. Unfortunately, only a handful of papers describe how their Bayesian student models are actually applied to a task other than assessment. Of those that do, there seem to be three general approaches: *alternative* strategies, *diagnostic* strategies, and *decision-theoretic pedagogical* strategies. The three classes and the systems that fall into them are given in Table 1.

Table 1. Decision-making with the student model.

Alternative	Diagnostic	Decision-Theoretic
ANDES Ganeshan et. al., 2000 SQL-TUTOR	Millán et. al., 2000 Collins et. al., 1996	DT-TUTOR CAPIT

Alternative strategies

Alternative strategies optionally take the posterior probabilities of the Bayesian network and use them as the input to some heuristic decision rule. To illustrate, ANDES selects hints for the student based on the solution path that the student is following to solve the current problem (Gertner et. al., 1998). However, the student's solution path is by no means certain (e.g. the student could be on paths *A*, *B* or *C* with posterior probabilities *p(A)*, *p(B)*, and *p(C)*), and therefore the system uses the heuristic of assuming that the most probable solution path (e.g. *A*, assuming *p(A)>p(B)* and *p(A)>p(C)*) *is* the student's solution path. However, this is a sub-optimal heuristic as demonstrated by a simple counter-example. Suppose the optimal hint for solution path *A* is H_1, but the optimal hint for both paths *B* and *C* is H_2. Then if it is the case that $p(B) + p(C) > p(A)$, hint H_2 will be optimal, but the heuristic rule will incorrectly select hint H_1. ANDES also has heuristic decision procedures disconnected entirely from the student model. For example, a simple matching heuristic is used to generate feedback on incorrect equation entries (Gertner, 1998).

Another system using heuristic decision procedures is ADELE (Ganeshan et. al., 2000). ADELE has a Bayesian network model of the domain knowledge, but it uses a heuristic based on focus-of-attention to select the node in the network about which to provide a hint. Decision-theoretic processes were considered but abandoned because they were considered too inefficient.

Finally, SQL-Tutor uses a heuristic for problem selection (Mayo & Mitrovic, 2000). The main rationale for this was that, like ADELE, the computation required for exact decision-theoretic computation (which would have involved more than 500 constraints) made direct application of decision theory intractable. The heuristic used was based on Vigotsky's Zone of Proximal Development (Vigotsky, 1978), and did tend to select problems of an appropriate complexity level efficiently. However, this approach is not guaranteed to select the optimal problem.

Diagnostic Strategies

This is the approach of Millán et. al. (2000), which expands on the strategy suggested by Collins et. al. (1996). The basic idea is to select actions whose outcomes are likely to maximise the posterior precision of some node in the network. For example, Millán et. al.'s domain is test question selection, and questions are selected to maximise the system's certainty that the student

has mastered the domain concepts. This strategy has limited applicability outside of diagnostic tests.

Decision-Theoretic Pedagogical Strategies

Decision-theoretic strategies are utilised in both DT-Tutor (Murray & VanLehn, 2000) and CAPIT (this paper). Both systems select tutorial actions that maximise expected utility (Equation 1). While diagnosis is obviously an important component of expected utility maximisation, it is only a secondary component. The primary consideration of an expected utility calculation is the likely outcomes of the action, and their pedagogical utility. For example, in CAPIT as shall be described, the expected utility of an action (e.g. problem selection) depends on the likely outcomes of the action (e.g. how many errors are made). In DT-Tutor, the action's impact on many different factors related to the student (e.g. their morale, etc) has an influence on expected utility. Diagnosis, therefore, is only required to the extent that it discriminates between alternate actions. The key difference between the two systems is, as Figure 3 depicts, DT-Tutor has a static, expert-centric student model whereas CAPIT has a data-centric student model that can adapt on-line. This impacts on action selection because the crucial $p(x|d_i)$ component of Equation 1 is evaluated using the Bayesian model.

Summary: Desirable Features

To summarise, there are a number of desirable features of decision-theoretic tutors. The first obvious desirable feature is to select pedagogical actions according to pure decision-theoretic principles rather than heuristics. This, combined with a Bayesian student model, means that the system will be fully normative and therefore its behaviour will be optimal. Secondly, the data-centric approach has two key advantages: the specification size of the network is smaller, and its predictive performance can be readily evaluated. This data-centric approach is therefore an attractive approach. The approach of MANIC (Stern et. al., 1999), in which the probabilities of the Bayesian network are initialised from population data (the "population student model") and subsequently adapted on-line to the current student, was a first step in this direction. A natural extension to MANIC's approach is to abandon the assumption that the student model is a hierarchy, and instead learn its structure as well as its conditional probabilities from data. The methodology presented in the next section shows how to develop just such a system with these desirable features.

A METHODOLOGY FOR DEVELOPING RATIONAL PAS STRATEGIES

In this section, one approach to building a normative intelligent tutoring system is described. While this is the general approach used to build CAPIT, it is by no means the only approach. The approach is described as a five-step methodology. The main point to note is that in the first step, a version of the tutor with no intelligent decision-making capabilities is deployed in a classroom. The point of this is to collect data describing the behaviour of students in the domain. All the student actions and system responses are logged, and then machine learning techniques are then used to induce a Bayesian network model from this data. In turn, the Bayesian network model is the basis for the decision-theoretic PAS strategies. Table 2 illustrates the process, although like any engineering process, the order of the steps is by no means fixed. The steps are now discussed in more detail.

Table 2. The five-step methodology for designing decision-theoretic PAS strategies.

1	Randomised Data Collection
2	Model Generation
3	Decision-Theoretic Strategy Implementation
4	On-line Adaptation
5	Evaluation

The first step is randomised data collection, in which an almost fully-functional version of the tutor is tested in a classroom representative of the intended population of users of the system. The only difference between this and the final version of the tutor is the PAS strategy: this version's PAS strategy is random. That is, given a set of alternatives (such as unsolved problems), the tutor makes the selection completely randomly. All actions should be logged as records of form <*State, Action, Outcome*>, where *State* is a description of some state prior to the action selection (e.g. the state of the student model, or the recent history of the student, or a combination thereof), *Action* is the pedagogical action that is randomly selected (e.g. the next problem), and *Outcome* is the observed outcome(s) of the action (e.g., correct or incorrect). Because PAS selection is random, the data should be uniformly spread over all the possible actions.

The next step is model induction, the construction of a Bayesian network for predicting student performance (the outcomes) given a state and an action. The data from Step 1 serves as the source from which the model is induced. At this stage, prior and expert knowledge can be added to the network. This can be done either before learning by adding dependencies and probabilities between the variables, or after learning, by fine-tuning the induced network. There is at least one Bayesian network learning algorithm that can cope with this type of prior knowledge (Cheng et. al., 1998). However, the rationale for any decision at this stage should be to enhance predictive performance. Also, because the network is being learned from data, the variables can only represent observations.

The third step is implementation of the decision-theoretic strategy. This is an encoding of Equation 1 to design a procedure that selects pedagogical actions that maximise expected utility. Equation 1 has two main components, the utility function $U(x,d)$, and the conditional probabilities, $p(x|d)$, for each possible outcome x of each potential next action d. The Bayesian network constructed in the previous step is used to provide the outcome probabilities, $p(x|d)$. However, the utility function is not yet defined. In fact, it is at this point that learning theories are incorporated into the system. The utility function essentially defines a learning theory for a particular task. To illustrate, if d represents a possible next problem and x is the number of errors the student makes when attempting the problem, then $U(x,d)$ can be defined to be maximal for some optimal number of errors. This is exactly the strategy used in one of the decision strategies in CAPIT, and will be discussed in more detail later.

Step four involves implementing an on-line Bayesian network learning algorithm. In Step 2, the Bayesian network is constructed from population data. Stern et. al. (1999) refer to this as a "population student model". However, as data is acquired directly from the current student, the population data should be gradually discounted. Additionally, as the student state changes over time, older data acquired from the student will need to be discounted as well. While there are a number of existing algorithms for Bayesian network induction from data, there is little in the way of *on-line* Bayesian network induction algorithms. Furthermore, the online learning algorithms that do exist (e.g. Heckerman, 1999; Bauer et. al., 1998) make the assumption that the data-source is essentially static and unchanging over time, in direct contrast to an actual student whose state changes constantly. In our application of the methodology to CAPIT, therefore, we describe a modification to an existing algorithm for on-line conditional probability learning, and avoid the much more difficult problem of updating a network's structure on-line.

Finally, the fifth step is an evaluation of the decision-theoretic PAS strategy. This is necessary to ensure that the decision-theoretic strategies actually provide a pedagogical benefit for the extra computational effort they require. One strategy that requires virtually no computational effort is the randomised PAS strategy implemented for Step 1. We therefore advocate evaluating decision-theoretic and randomised PAS in a controlled experiment in which one group of students (the control group) use a version of the tutor with the original, randomised strategy, and the second group of students (the experimental group) use tutor version with the decision-theoretic strategy. We have performed this evaluation with CAPIT.

CAPIT: AN INTELLIGENT TUTOR FOR CAPITALISATION AND PUNCTUATION

CAPIT (Mayo et. al. 2000) is an intelligent tutor implemented in Visual Basic 6. It runs on any 32-bit Windows platform, and use the MSBN API provided by Microsoft for its Bayesian networks implementation (http://research.microsoft.com/msbn).

It is also the second intelligent tutor to implement Ohlsson's Constraint-Based Modelling (CBM) (Ohlsson, 1994), the other being a tutor for the SQL database language (Mitrovic & Ohlsson, 1999). CBM was proposed in part because of the intractability of modelling approaches that try to infer students' mental processes from problem solving steps, and in part because Ohlsson believes that diagnostic information is most readily available in the problem states that the student arrives at. It is also computationally highly efficient.

A CBM tutor represents domain knowledge as a set of constraints of the form $<C_r, C_s>$ where C_r is the *relevance condition* and C_s is the *satisfaction condition*. The constraints define which problem states are consistent, and which are not. A constraint is relevant to a problem if its C_r is true. All constraints that are relevant to a problem state must also be satisfied for the problem state to be correct. Otherwise, the problem state is incorrect and feedback can be given depending on which relevant constraints had their satisfaction condition violated.

Traditional capitalisation and punctuation exercises for children tend to fall into one of two categories (Bouwer, 1998): *completion* (the student must punctuate and capitalise a fully lowercase, unpunctuated piece of text), and *check-and-correct* (the student needs to check for errors, if any, and correct them). CAPIT poses problems of the first class, the completion exercise. If the child makes a mistake, an error message is displayed. For example, Table 3 depicts one of the shorter problems in the system, a student's incorrect attempt at punctuating and capitalising it, and the tutor's correct solution.

Table 3. (a) A problem, (b) a student's incorrect solution, and (c) the correct solution.

(a) the driver said it will rain
(b) The driver said, "it will rain".
(c) The driver said, "It will rain."

There are two errors in the student's solution in Table 3: the direct speech does not start with a capital letter, and the period is outside the quotation marks. Currently, CAPIT displays only one error message at a time, and the student is expected to correct the error (and any others) and resubmit the problem before any more feedback is displayed. If the student submitted this solution, a feedback message such as *The full stop should be within the quotation marks! Hint: look at the word* rain *in your solution* would be displayed. Error messages are typically short and relate to only a single mistake, but if the student wants more detailed information, she/he can click *Why?* to be shown further explanatory material.

The current version of CAPIT contains 45 problems and 25 constraints. The problems are relevant to the constraints in roughly equal proportions, although a small number of constraints (such as capitalisation of sentences) are relevant to all the problems. The constraints cover the following parts of the domain:

- Capitalisation of sentences.
- Capitalisation of the names of both people and places.
- Ending sentences with periods.
- Contracting *is* and *not* using apostrophes (e.g *haven't* is a contraction of *have not*).
- Denoting ownership using apostrophes (e.g. *John's dog*).
- Separating clauses using commas.
- Separating list items using commas (e.g. *apples, oranges, lemons and pears*).
- Denoting direct speech with quotation marks.

- The correct punctuation of the possessive pronoun *its*.

While the domain coverage is not complete, it is adequate to make CAPIT an intelligent tutoring system with practical application. In our evaluation study, classes used the tutor over a period of one month. While some students quickly mastered all the rules, most of them failed to master all the rules by the end of the evaluation.

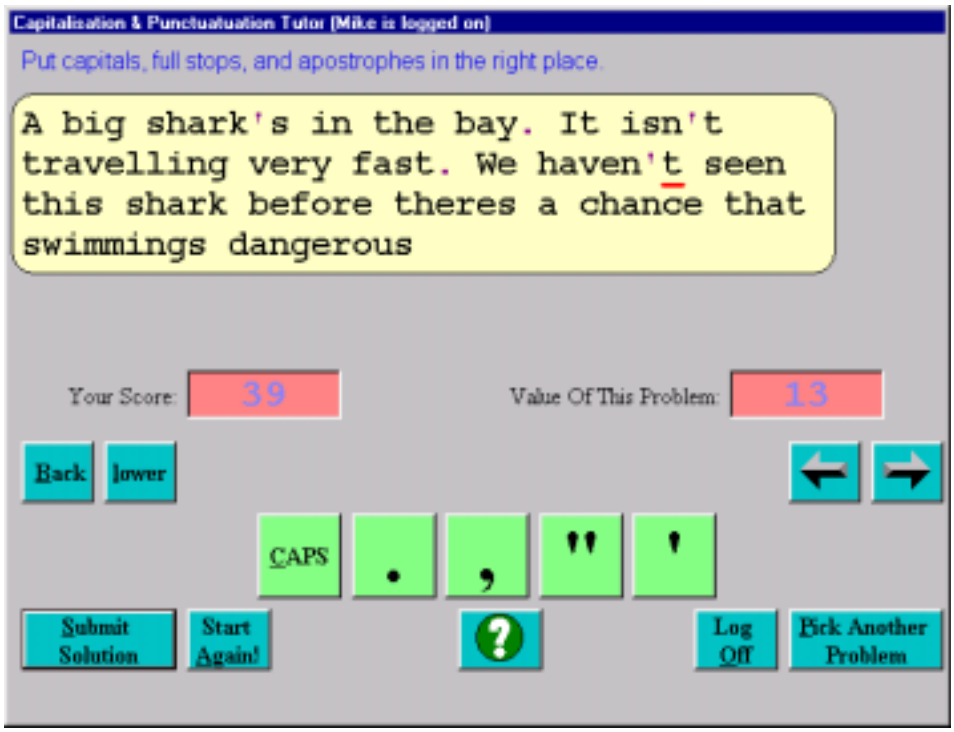


Figure 5. CAPIT's main user interface.

CAPIT's main user interface, showing a partially completed problem, is depicted in Figure 5. Brief instructions relevant to the current problem are clearly displayed at the top of the main interface. This reduces the cognitive load by enabling the learner to focus on the current goals at any time without needing to remember them. Immediately below the instructions, and clearly highlighted, is the current problem. In this area, the child interacts with the system by moving the cursor using keyboard or mouse, capitalising letters, and inserting punctuation marks. The child can provide input either by pointing and clicking the mouse, or by pressing intuitive key combinations such as *Shift-M* to capitalise the letter *m*. By requiring the cursor to be positioned at the point where the capital letter or punctuation mark is to go, the child's ability to locate errors as well as correct them is tested.

Motivation is provided in two ways. Firstly, whenever a correct solution is submitted, some points are added to the child's score. The number of points added is equal to the number of punctuation marks and capital letters in the solution that was just submitted. Secondly, whenever a correct answer is submitted, an animation is displayed. These simple strategies were found to be highly effective motivators for children in the target age group of 8-10 year olds.

Figure 6 shows the architecture of CAPIT. The student model comprises a record of the outcome of the previous attempt at each constraint[1] (the short-term model) and the current configuration of the Bayesian student model (the long-term model). The student modeller is a pattern matcher that takes the student's solution to a problem and determines which constraints are violated. It then passes the violated constraints (if any) to the pedagogical module. The pedagogical module is the core component of the system. It performs two significant PAS tasks:

[1] An "attempt at a constraint" in this context means an attempt at a problem whose solution is relevant to the constraint.

firstly, given the violated constraints, it selects the single violated constraint about which feedback should be given; secondly, when *Pick Another Problem* is clicked, or when the student solves the current problem, the pedagogical module selects the most appropriate next problem for the student. The current version of the pedagogical module can perform PAS in two ways: randomly, or using decision theory. More details of the decision-theoretic strategies are given later.

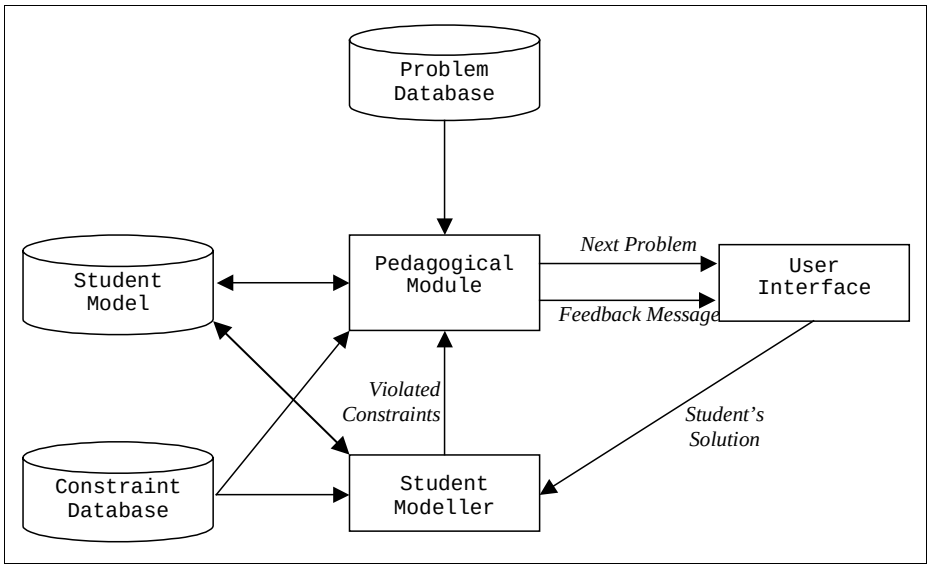


Figure 6. The architecture of CAPIT.

Problems in CAPIT are represented as arrays of words. Each word in the problem representation is properly punctuated and capitalised, and the tutor generates the initial problem text by removing the punctuation marks and turning all capital letters into lowercase. Each word also has one or more *tags* associated with it. The tags specify the semantic and/or grammatical classes of a word, to the degree that it is relevant for punctuation and capitalisation. For example, Table 4 is the tutor's internal representation of a short problem. Each word in this problem has one, two or three tags. The tag `DEFAULT` indicates that a word does not need to be punctuated or capitalised (although the student may incorrectly attempt to do so), such as *driver* and *will* in the example. Because `DEFAULT` completely specifies the capitalisation and punctuation requirements, `DEFAULT` words do not require any other tags. Other tags such as `L-CASE` indicate that a word does not need to be capitalised, but says nothing about the punctuation requirements (and vice-versa for the tag `NO-PUNC`). Other types of words need more specific tags. For example, *The* is the first word in the sentence and therefore carries the tag `SENTENCE-START`. Similarly, *rain* is the last word in both the sentence and the direct speech. This fact is reflected by one of its tags, `DIRECT-QUOTE-ENDING-SENTENCE`. A longer example, which is more representative of the complexity of the 45 problems in the database, is given in Table 5.

Table 4. Problem representation for *The driver said, "It will rain."*

The	`SENTENCE-START,NO-PUNC`
driver	`DEFAULT`
said,	`WORD-PRECEDING-DIRECT-QUOTE,L-CASE, ONE-PUNC`
"It	`DIRECT-QUOTE-START,ONE-PUNC`
will	`DEFAULT`
rain."	`DIRECT-QUOTE-ENDING-SENTENCE,L-CASE, TWO-PUNC`

Table 5. Representation of a more complex problem.

There's	SENTENCE-START,IS-CONTRACTION,ONE-PUNC
a	DEFAULT
bee	DEFAULT
buzzing	DEFAULT
past	DEFAULT
me.	SENTENCE-END,ONE-PUNC,L-CASE
It's	SENTENCE-START,IS-CONTRACTION,ONE-PUNC
taking	DEFAULT
its	ITS-POSSESSIVE-PRONOUN,NO-PUNC,L-CASE
honey	DEFAULT
back	DEFAULT
to	DEFAULT
its	ITS-POSSESSIVE-PRONOUN,NO-PUNC,L-CASE
hive.	SENTENCE-END,ONE-PUNC,L-CASE
I	SENTENCE-START,NO-PUNC
hope	DEFAULT
it	DEFAULT
knows	DEFAULT
its	ITS-POSSESSIVE-PRONOUN,NO-PUNC,L-CASE
way	DEFAULT
home.	SENTENCE-END,ONE-PUNC,L-CASE

Table 6. Examples of constraints.

	C_r	C_s	Msg
(a)	{DEFAULT, L-CASE}	^[a-z0-9%SYMBOLSET%]*$	This word doesn't need any capital letters!
(b)	{NAME-OF-PERSON}	^[%SYMBOLSET%]*[A-Z0-9]	Each word in a person's name should start with a capital!
(c)	{DIRECT-QUOTE-ENDING-SENTENCE}	[^%SYMBOLSET%]+((\.+"+)|"+|\.+)?$	The full stop should be within the quotation marks!
(d)	{ITS-POSSESSIVE-PRONOUN}	[^']s$	No apostrophe is required in its!

The constraints used in CAPIT comprise three parts: namely, the relevance condition, C_r, which is a set of tags; the satisfaction condition, C_s, which is a regular expression; and associated explanatory material. Table 6 gives examples of four out of the 25 constraints that range from the very general to the very specific. The error message field of each constraint shows only the hint that is displayed when the constraint is violated; in addition, most constraints have an associated page of textual explanation that is displayed when the student clicks the *Why?* button.

The tags of each word determine which constraints are relevant to the problem. If at least one word from the problem has a tag that is also in a constraint's C_r, then that constraint is relevant to the problem. For example, constraint (a) from Table 6 is relevant to the problem in Table 4 because several words have the tags DEFAULT and L-CASE. Similarly, constraint (c) is relevant to the same problem because the last word has the tag DIRECT-QUOTE-ENDING-SENTENCE. Constraints (b) and (d) are not relevant.

If a constraint is relevant to a word, then its satisfaction condition, C_s, is evaluated against that word. The satisfaction condition is a regular expression, which is a language for pattern matching. The main difference between the expressions used in the tutor and standard regular expressions is the presence of %SYMBOLSET% in the C_s. %SYMBOLSET% stands for a string of all the punctuation marks that the tutor knows about. For example, the current version of CAPIT deals with commas, periods, quotation marks and apostrophes. Therefore the C_s condition of constraint (a), ^[a-z0-9%SYMBOLSET%]*$, becomes the standard regular expression ^[a-z0-9'",.]*$ before being matched to a student's solution.

Briefly, a regular expression like `^[a-z0-9’”,.]*$` defines a pattern that can be matched to a string. The symbol `^` defines the start of the string and `$` defines the end of the string. The square brackets `[]` define a set of characters, while a negative character set (which matches any characters not in the set) is defined by square brackets with a `^` inside the brackets, e.g. `[^a-z]`. The symbol `*` usually follows a character or pattern and means “zero or more repetitions of the previous pattern”. Thus, the `^[a-z0-9’”,.]*$` is matched by any string containing zero or more characters that are lower case letters, numbers or punctuation marks. A word containing an upper case letter does not match the expression, and therefore the constraint would be violated. Constraint (d) has a much simpler C_s, `[^']s$`. This regular expression is simply matched by all strings ending in *s* that do not have an apostrophe in the penultimate position.

The constraints in CAPIT tend to fall somewhere on a continuum between general and specific. *General* constraints apply to many different words because they are relevant to general tags such as `DEFAULT` and `L-CASE`. As a result, the feedback is more general and may not address the specific misconceptions that led to the error. Constraint (a) from Table 4 is one example of this class. *Specific* constraints are satisfied only by specific punctuation/capitalisation patterns, and feedback can be more specific in this instance. For example, constraint (b) is satisfied only when a word with the tag `NAME-OF-PERSON` starts with a capital letter or digit (excluding any punctuation marks at the beginning). Constraint (c) is violated only when the student punctuates a word that ends both a direct quote and a sentence incorrectly, with the quotation mark preceding the period (e.g. see Table 1(b)). In this case, the tutor can tell the student specifically to reverse the order of the punctuation marks. In all other cases, the constraint is satisfied. Similarly, (d) is violated only when the student tries to add an apostrophe before the *s* in the possessive pronoun *its*, and is satisfied otherwise.

DECISION-THEORETIC PAS IN CAPIT

Decision problems conducive to our methodology include next problem selection, error message selection, topic selection, selective highlighting/hiding of text, and timing of interventions to give help. We decided to develop decision-theoretic strategies for the first two of these tasks, problem selection and error message selection. In this section, we describe how the general methodology was applied to develop these strategies for CAPIT.

Step 1: Randomised Data Collection

Initial data for Step 1 of the decision-theoretic PAS strategy development was acquired from a data acquisition deployment of CAPIT at Westburn School, Christchurch, New Zealand in March 2000 (Mayo et. al., 2000). A version of CAPIT was used in which problems and error messages were selected randomly. The problems came from the pool of all unsolved problems, and the error message was selected from the set of constraints violated on the current attempt (recall that there is one error message per constraint). The evaluation study consisted of four 30-45 minute sessions. Details of each problem attempt and error message displayed were logged. Subsequent analysis revealed the following averages. Each student made 89 attempts at 28 different problems. 21 of the problems were eventually solved, and 7 abandoned. Students violated an average of 181 constraints during the sessions, of which feedback was given on 68. A total of 3300 records of the form <*State, Action, Outcome*> were acquired during this step, where *State* is a record of the outcome of the last attempt at each constraint (which may include attempts at previous problems, if for example, a constraint was relevant to the last problem but is not relevant to the current problem), *Action* is the problem that was selected randomly, and *Outcome* is a record of the constraints that were violated and satisfied following the problem attempt.

Step 2: Model Selection

The data acquired from the evaluation study was used to generate the best Bayesian network for long-term student modelling. The selection criterion was the ability of the network to predict student performance on constraints. An issue at this point was whether to use a model in which the constraints were independent of each other, as in Reye's model, or whether to allow (more realistically) any dependencies between constraints to be learned from the data. This decision is quite significant because a model in which constraints are assumed independent can be formulated with only four variables, whereas a model in which any dependencies between constraints are allowed is much more complex and must consist of at least twice the number of variables as there are constraints. Figures 7 and 8 illustrate the competing "small" and the "large" specifications. In both diagrams, L_i represents the outcome of the last attempt at the ith constraint, and can take values S (satisfied), V (violated), VFB (violated with feedback), or NR (has not been relevant before). N_i is the predicted outcome of the next attempt, whose values are {S, V, NR}. Note that in accordance with the "desirable features" discussed earlier, neither networks explicitly model unobserved student states.

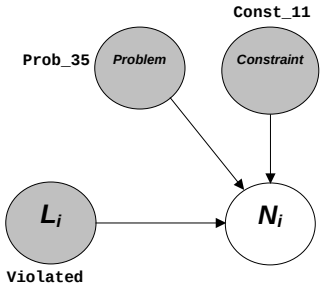


Figure 7. The structure of the small Bayesian network for predicting the outcome of the next attempt at the ith constraint. In this example, the network is predicting the outcome of the next attempt at constraint 11 which is relevant to the current problem, 35, and was previously violated.

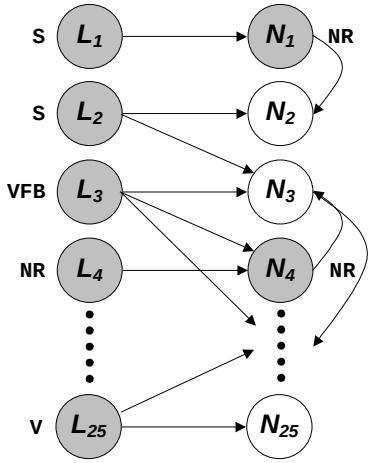


Figure 8. The structure of the large Bayesian network specification after learning. The student has previously satisfied constraints 1 and 2, violated constraints 3 and 25 (receiving feedback on 3), and has not yet attempted constraint 4. The network is currently configured to predict this student's performance on a problem whose relevant constraints include 2, 3 and 25.

A number of variants of the large network were considered. In each case, an algorithm proposed by Cheng et. al. (1998) for structural learning by mutual entropy maximisation was utilised to learn a Bayesian network from the data collected in Step 1. The conditional probabilities were estimated using the standard Dirichlet priors approach (Heckerman, 1999) which is described in more detail later. An important component of the structural learning

algorithm is the minimum threshold. This essentially determines the minimum amount of mutual information required between two variables before an arc can connect them. For these experiments, minimum thresholds of 4, 6 and 10 were selected (initial experiments showed that a threshold below 4 resulted in a network far too large for on-line evaluation). Another parameter that we wanted to investigate was the addition of prior knowledge: does it enhance predictive performance? The "obvious" prior knowledge to add is an arc from L_i to N_i, for each constraint i, indicating that at the very least, the outcome of the next attempt at a constraint is partly dependent on the outcome of the previous attempt. We thus formulated six specifications for large networks: *Large(4)*, *Large(6)* and *Large(10)* being the specifications without prior knowledge, and *PLarge(4)*, *PLarge(6)* and *PLarge(10)* being specifications with prior knowledge. For each of the large specifications, the L_i nodes were fixed as root nodes, to reflect the fact that they come before the N_i nodes in temporal order. Thus, there are two types of arc that can be learnt from the data for the large networks: arcs from *L* layer to the *N* layer, and arcs within the *N* layer.

The data was then divided into training and test datasets. Approximately 20% of the records were selected randomly into the test dataset. The remaining 80% were kept in the training set, and used to train one large network for each of the 6 specifications. A simpler network equivalent to Figure 7 (*Small*) was also trained from this data, although in this case the structure was already specified and only the conditional probability $p(N_i|L_i,Problem,Constraint)$ had to be learned. This entire process of training was repeated three times (j=1..3), each time with a different randomly generated training/test dataset divisions. The total number of different networks that were generated, therefore, was 21.

The first question to answer was whether or not the larger networks were better predictors of student performance than the small ones on the test data. Each of the 6 large networks generated from the jth training set was compared to the *Small* network generated from the jth training set in the following way. For each problem attempt in the jth test set, the large and small networks were given the values for $L_1..L_{25}$. The large networks had their $P_1..P_{25}$ nodes instantiated to NR for each constraint not relevant to the attempt's problem. The values of the remaining uninstantiated nodes in $P_1..P_{25}$ were then predicted. For the large networks, this required one evaluation of the network, and for *Small*, one evaluation per relevant constraint was necessary. The standard junction tree algorithm for Bayesian network inference was used (Lauritzen & Spiegelhalter, 1988). Then, for each P_i representing a relevant constraint, the predicted value of P_i was compared to the actual value of P_i. The total number of correct predictions was counted. A correct prediction was deemed to occur if the predicted outcome with maximum probability matched the actual outcome. To clarify what was done further, the output of each comparison was essentially a table of tuples of the form $<i, L_j(i), S_j(i), T_j(i)>$, where i=1..n_j is the attempt (n_j is the number of attempts in the jth test set), $L_j(i)$ is the number of correct predictions given by the large network, $S_j(i)$ is the number of correct predictions given by the small network, and $T_j(i)$ is the maximum number of correct predictions (simply the total number of relevant constraints on that attempt).

Table 7. The coefficients of determination characterising the number of correct predictions as a function of the total number of relevant constraints, for each network and training/test dataset.

	$TestData_1$ (n=639)	**$TestData_2$ (n=608)**	**$TestData_3$ (n=686)**
PLarge(10)	0.7649	0.7434	0.7638
PLarge(6)	0.7562	0.7385	0.7615
PLarge(4)	0.7101	0.7208	0.7417
Large(10)	0.7446	0.7464	0.7495
Large(6)	0.749	0.7347	0.7511
Large(4)	0.7117	0.7236	0.7387
Small	0.7312	0.7086	0.7314

The coefficient of determination (r^2) was calculated for each network by taking the number of correctly predicted constraints as a function of the number of relevant constraints. The results are summarised in Table 7. The r^2 values of the large networks are higher than those of the small networks, suggesting that the large specifications are better.

Next, we tested to see if the large networks were statistically significantly better predictors of student performance than the small networks. Note that for each of the 18 comparisons, the number of correct predictions made by the small and large networks are paired. That is, for each attempt i=1..n_j in each of the 18 tests, both an $S_j(i)$ and a $L_j(i)$ were generated, both of which can be considered stochastic functions of i. Therefore, the samples are pair-wise dependent. A paired-difference experiment (McClave & Benson, 1991, pp. 421-7) was used to test for significant differences.

Table 8 shows that the *PLarge(10)*, *PLarge(6)* and *Large(6)* specifications all produce Bayesian networks that are statistically significantly better predictors of student performance than the networks produced by the *Small* specification. The other specifications each had at least one comparison where no statistically significant difference was found (indicated by "Accept H_0"). For these tests, a high t value indicates greater significance. For all the networks, the outcomes in the second test set were much more difficult to predict than those of the first and third sets, resulting in lower t values. The exception to this is *Large(10)*, which happened perform barely well on the second test set but not the first and third. From these results, we were able to rule out *Small* as a worthwhile specification to continue with.

Table 8. Results of two-tailed paired difference experiments comparing each large network against *Small*. H_0 is the hypothesis that there is no difference in the mean number of correct predictions made by both networks. A positive t value indicates that the large network is better than *Small*. The rejection region for all the datasets is approximately ±2.58 for 99% confidence, and ±1.96 for 95% confidence.

	$TestData_1$ (n=639)	**$TestData_2$ (n=608)**	**$TestData_3$ (n=686)**
	Small	**Small**	**Small**
PLarge(10)	t=5.75, α=0.01	t=3.95, α=0.01	t=5.46, α=0.01
PLarge(6)	t=5.17, α=0.01	t=3.30, α=0.01	t=5.06, α=0.01
PLarge(4)	Accept H_0	Accept H_0	t=2.1, α=0.05
Large(10)	Accept H_0	t=1.98, α=0.05	Accept H_0
Large(6)	t=4.01, α=0.01	t=2.78, α=0.01	t=3.93, α=0.01
Large(4)	Accept H_0	Accept H_0	t=2.05, α=0.05

The next task was to determine the most accurate large network. In particular, does the selection of the minimal threshold or the addition of prior knowledge result in improved performance? For this analysis, the three best large networks (*PLarge(10)*, *PLarge(6)* and *Large(6)*) were compared. No statistically significant difference was found between *PLarge(10)* and *PLarge(6)* on any of the training/testing dataset divisions. However, significant differences were found between *PLarge(10)* and *Large(6)* as Table 9 shows.

Table 9. Results of two-tailed paired difference experiments comparing *PLarge(10)* against *Large(6)*. H_0 is the hypothesis that there is no difference in the mean number of correct predictions of made by both networks. A positive t value indicates that the *PLarge(10)* is better than *Large(6)*. The rejection region for all the datasets is approximately ±2.58 for 99% confidence, and ±1.96 for 95% confidence.

	$TestData_1$ (n=639)	**$TestData_2$ (n=608)**	**$TestData_3$ (n=686)**
	Large(6)	**Large(6)**	**Large(6)**
Plarge(10)	t=3.13, α=0.01	Accept H_0	t=2.08, α=0.05

To conclude step 2, *PLarge(10)* was selected as the best specification to proceed with because the t values for *PLarge(10)* were, on average, greater than those of *PLarge(6)* in Table 8. The training and testing datasets were combined into a single dataset and a Bayesian network with the *PLarge(10)* specification was learned. One of the desirable features discussed earlier was to take advantage of the unique ability of Bayesian network to integrate prior knowledge and data; this has been shown to improve predictive accuracy here.

Step 3: Decision-Theoretic Strategy

Step 3 is the implementation of decision-theoretic PAS strategies. The key task here is to define a utility function U(x,d) specific to the PAS strategy that can be substituted into Equation 1 to yield a task-specific expected utility function. For CAPIT, we were interested in two tasks: next problem selection, and error message selection following an attempt in which multiple constraints are violated.

The value of the next problem $d \in$ {Problem_1, …, Problem_45} is determined by predicting the student's performance on the problem with the Bayesian network. An appropriate problem can be considered to fall into the zone of proximal development, defined by Vigotsky (1978) as "the distance between the actual development level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance or collaboration of more capable peers". We interpret this as stating that a student should be given a problem slightly above their current level but not so difficult as to be discouraging. This principle implies that utility should be maximised for problems where one or two errors are likely (reflecting a challenging problem), but minimised for problems whose outcome is no errors (being too easy) or several errors (being too hard). This utility function is defined in Table 10.

Table 10. The utility function for problem selection. Utility is maximised for problems resulting in one or two errors only.

x	U(x,d)
No-Errors	0.0
1-Error	1.0
2-Errors	1.0
3+Errors	0.0

The cost of all problems is assumed to be zero. Let us also assume that ξ comprises the student history (i.e. the instantiations of $L_1..L_{25}$) and the instantiations of N_i to NR for those constraints not relevant to d. Substituting into Equation 1 yields the expected utility of problem d:

$$\begin{aligned} E[U(x,d)\,|\,\xi] \;=\;& p(\texttt{No-Errors}|\,d,\xi)\,U(\texttt{No-Errors},d) \\ & + p(\texttt{1-Error}|\,d,\xi)\,U(\texttt{1-Error},d) \\ & + p(\texttt{2-Errors}|\,d,\xi)\,U(\texttt{2-Errors},d) \\ & + p(\texttt{3+Errors}|\,d,\xi)\,U(\texttt{3+Errors},d) - 0 \\ =\;& p(\texttt{1-Error}|\,d,\xi) + p(\texttt{2-Errors}|\,d,\xi) \end{aligned}$$

Now we need to calculate p(1-Error| d,ξ) and p(2-Errors| d,ξ) from the Bayesian network. This is not straightforward because the predicted outcomes $N_1..N_{25}$ are not necessarily mutually or conditionally independent. In fact, the best way to deal with this computation is to extend the Bayesian network itself at runtime by adding a deterministic function *NumErrors* $\in$ {No-Errors, 1-Error, 2-Errors, 3+Errors} to the network, which is dependent on the relevant constraints only. The function simply counts the number of its parents that are violated, but because the parents of *NumErrors* are uncertain, the uncertainty is transferred to *NumErrors* by the Bayesian network inference algorithm (Lauritzen & Speigelhalter, 1988) in the correct way. The addition of *NumErrors* to the example large network is illustrated in Figure 9. The probabilities of Equation 1 can now be determined by querying the posterior distribution over the *NumErrors* variable.

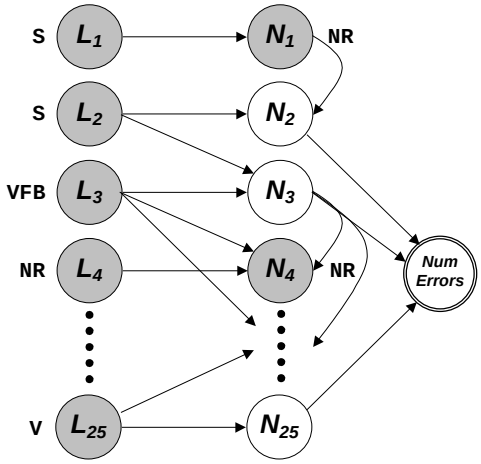


Figure 9. The same large network as depicted in Figure 8, but with *NumErrors* added as a child of all the N_i nodes representing relevant constraints.

The strategy for decision-theoretic error message selection is slightly different. In this case, $d \in \{\mathsf{FB_i} \mid$ Constraint i was relevant and violated on the last attempt$\}$ where $\mathsf{FB_i}$ is the decision to give feedback on the ith constraint. It is assumed that an error message about a constraint can influence the outcome of the next attempt at the constraint, resulting in a satisfaction (desired) or a violation (not desired). Table 11 characterises this as a utility function

Table 11. The utility function for feedback selection.

x	U(x,d)
N_i=V	0.0
N_i=S	1.0

Because the system gives feedback on only one violated constraint per attempt, the probabilities of these outcomes can be read directly from the network by "pretending" that feedback was given on the ith constraint. That is, we instantiate L_i to VFB instead of V and query N_i to obtain $p(N_i=\mathsf{V}|d, \xi)$ and $p(N_i=\mathsf{S}|d, \xi)$. However, the cost of each feedback message cannot be assumed to be zero, because each constraint will have a different probability of being satisfied without feedback, anyway. This probability of satisfaction without feedback can be considered the "opportunity cost" of giving feedback on the ith constraint, which is therefore defined as:

$$cost(d) = p(N_i=\mathsf{S}|\xi)$$

Substituting these values into Equation 1 yields the expected utility for feedback:

$$\begin{aligned} E[U(x,d)\mid\xi] &= \big(p(N_i=\mathsf{S}\mid d,\xi)U(N_i=\mathsf{S}) + p(N_i=\mathsf{V}\mid d,\xi)U(N_i=\mathsf{V})\big) - p(N_i=\mathsf{S}\mid\xi) \\ &= p(N_i=\mathsf{S}\mid d,\xi) - p(N_i=\mathsf{S}\mid\xi) \end{aligned}$$

The expected utility of an error message is therefore the posterior gain in probability of the constraint being satisfied that the message results in.

This step illustrates the framing of two simple learning theories as utility functions. To capture the more general notions such as a curriculum, other learning theories could be represented as more complex utility functions and thus integrated into the normative framework.

Step 4: On-line adaptation

The next challenge was implementing an on-line learning algorithm so that the Bayesian student model would adapt to the student. Heckerman (1999) shows how to calculate conditional probabilities for a Bayesian network from data. Let $X=x_k|Pa_x=pa_x$ represent an observation of variable X in state x_k when its parents Pa_x are in configuration pa_x. A normal Bayesian network maintains, for each possible $X=x_k|Pa_x=pa_x$, a single conditional probability $p(X=x_k|Pa_x=pa_x)$. The Dirichlet priors approach treats $p(X=x_k|Pa_x=pa_x)$ itself as an uncertain variable, and calculates its expected value from the data. It turns out that by assuming that the probability distribution over $p(X=x_k|Pa_x=pa_x)$ is a Dirichlet distribution, the expected value corresponds to the frequency. Suppose $X=x_k|\ Pa_x=pa_x$ has been observed α_k times while $Pa_x=pa_x$ has been observed α times. Obviously $\alpha_k \leq \alpha$. Then, it shown by Heckerman (1999) that the expected value of $p(X=x_k|Pa_x=pa_x)$ is:

$$E[p(X=x_k|Pa_x=pa_x)] = \frac{\alpha_k}{\alpha}$$

If $Pa_x=pa_x$ is observed a further N times, while N_k further observations of $X=x_k|pa_x$ are made, the expected value of the conditional probability updates simply to:

$$E[p(X=x_k|Pa_x=pa_x)|\text{observations}] = \frac{\alpha_k + N_k}{\alpha + N}$$

The parameters α and α_k are known as sufficient statistics, because they are adequate to define a Bayesian network; once they have been calculated, the rest of the training data can be discarded.

The problem with this algorithm is that it does not take into account the temporal ordering of the cases, and therefore there is no way to "bias" the conditional probabilities towards to most recent cases. This is an incorrect assumption for an intelligent tutor to make because the student's state is expected to change constantly. The system therefore needs a way of gradually discounting the effects of old data. However, the effect of the standard approach would be that as α gets increasingly large, the influence of new cases on the conditional probabilities decreases. To illustrate, CAPIT's population student model was learned from records of approximately 3300 problem attempts. The average student is likely to make only 50-100 problem attempts. Therefore, the standard Dirichlet priors approach would not be expected to adapt the network's parameters to the student to the desired extent.

Fortunately, the standard approach can be modified to prefer more recent observations. Our solution is to reduce α to a value such that the effect of new cases becomes significant. Let that value be α_{MAX}. The sufficient statistics α and α_k can now be replaced by two new statistics, α' and α_k', defined as:

$$\alpha' = \alpha_{MAX},\ \alpha_k' = \alpha_{MAX}(\alpha_k/\alpha)$$

The lower the constant α_{MAX}, the more significance new cases will have on the conditional probabilities. In CAPIT, the conditional probabilities are updated after every attempt (so N=1). The update rule, therefore, simplifies to:

$$E[p(X=x_k|pa_x)|\text{One observation of } X=x_k \text{ when } Pa_x=pa_x] = \frac{\alpha_k'+1}{\alpha'+1}$$

$$E[p(X=x_k|pa_x)|\ j \neq k, \text{One observation of } X=x_j \text{ when } Pa_x=pa_x] = \frac{\alpha_k'}{\alpha'+1}$$

A value for α_{MAX} was chosen by trials with simulated students. Two students were simulated: a "good" student who got every problem correct, and a "bad" student who made numerous mistakes and frequently abandoned problems. A domain expert analysed the sequence of problems that was selected for each student. It was found that when $\alpha_{MAX} > 5$, the system was not quick enough to present challenging problems to the good student even after several problems were solved correctly in a single attempt. This occurred simply because the conditional probabilities did not update fast enough. For the bad student, simple problems were repeatedly selected regardless of the value of α_{MAX}. A value of $\alpha_{MAX} = 5$ was therefore selected.

Step 5 Evaluation

Two evaluations were performed; a simple, informal evaluation to ensure that the system was behaving reasonably, followed by an extensive classroom evaluation with school students.

Simulated Students Evaluation

The first step in the evaluation was an informal observation of the behaviour of the decision-theoretic version of CAPIT. The observations were noted during the trial run with the simulated good and bad students. The system always started with the easiest problem, which involved merely dividing the text into sentences and inserting capital letters at the start, and periods at the end, of each sentence. For the good student, further problems typically introduced new constraints one at a time until a certain point was reached (probably when the posterior probability of the student satisfying the most common constraints was sufficiently high), after which more difficult problems (e.g. direct speech problems) introducing several new constraints at a time were selected. This is similar to a human tutor assessing a good student's capabilities initially with easier problems, before moving more directly to challenging problems. For the bad student who repeatedly made mistakes and abandoned problems, the tutor appeared to repeatedly select problems from a pool of 3-4 easier problems but never selected the same problem twice consecutively. Again, a similar strategy to that of a human tutor returning to previously abandoned problems while maintaining some variety. Note that problems in this system do not have explicit levels – all unsolved problems are available to be selected at any one time. Feedback selection was also observed. In the extreme case of a bad student who repeatedly submitted the same (incorrect) solution with multiple violated constraints, the selection of feedback messages seemed to cycle from the most to the least specific constraints, and back again, with each attempt. Again, there is no explicit rule programmed into the tutor to make it do this.

Classroom Evaluation

Three classes of 9-10 year olds at Ilam School in Christchurch, New Zealand, participated in a four-week evaluation of CAPIT. The first class (Group A) did not use the tutor at all. The purpose of this group was to provide a baseline for comparing the pre and post test results of students that did use a tutor in the domain with those that did not. The second class (Group B) used the initial version of the tutor with randomised problem and error message selection, and the third class (Group C) used the full version of the tutor with decision-theoretic PAS and the adaptive Bayesian student model. The groups using the tutor, B and C, had one 45-minute session per week for the duration of the study, and they worked in the same pairs each week. (Working in pairs was necessary because of the limited availability of computers.) Every interaction was logged. Pre and post tests were also completed, with students completing the tests in the same pairs.

Some significant attributes of the performance of Groups B and C during the evaluation are summarised in Table 12. Pairs of students in Group C used CAPIT for approximately 34 minutes more on average than those in Group B, which was a result of a teacher cutting one of the sessions short. As a result, Group C made many more attempts, and asked for more *Why?* explanations, than Group B. The average time per attempt for both groups is approximately 43 seconds. However, despite the additional interaction time, Group C attempted and solved less problems than Group B, and abandoned more problems. This is probably due to Group C being a less-able class than Group B, an observation that was confirmed by the teachers. In hindsight, pairs of students should have been assigned to groups randomly rather than by class. An interesting discrepancy is the mean number of attempts per solved problem. Group C performed better here, perhaps suggesting that the feedback messages in their case were better adapted.

Table 12. Averages describing the behaviour of Groups B and C's.

	Group B	**Group C**
Number of pairs	16	14
Ave. interaction time per pair (mins)	80.9	115
Ave. # attempts	109.7	167.3
Ave. # solved problems per pair	29	22
Ave. # attempted problems per pair	34	30
Ave. # attempts per solved problem	5.8	5.5
Ave. # expl. asked for per pair	10.3	18

Further analysis was performed at the level of the individual constraints. Figure 10 gives the average number of times each constraint was relevant per user. This reflects the higher number of attempts made by Group C, and highlights the constraints that are common to most of the problems, for example, constraint 4 (a sentence must start with a capital letter) and constraint 7 (a sentence must end with a period). This table can be compared to Figure 11, the frequency with which constraints were violated when relevant. It shows that Group C violated proportionately more constraints that Group B, which corresponds with the averages in Table 11. However, it is interesting to note that the constraints defining the correct punctuation of direct speech (constraints 17-25) were violated proportionately less by Group C. This suggests that the decision-theoretic problem sequencing placed problems involving direct speech (which are more difficult) later in the sequence, after the student had mastered the other constraints, therefore allowing them to focus on learning direct speech punctuation.

The pre- and post-tests were comparable (and challenging) and consisted of eight completion exercises similar to those presented by CAPIT, but done manually with pencil-and-paper. Students worked in their assigned pairs to complete the test. The score for each test was calculated by subtracting the number of punctuation and capitalisation errors from the number of punctuation marks and capital letters required for a perfectly correct solution; it was thus possible for a pre- or post-test to have a negative score (fortunately none of the students were that bad). The mean scores and standard deviations (the Y error bars) are shown in Figure 12. The mean pretest score for Group C is almost 10% lower than that of Group B. Both Group B and C show an improvement in mean test scores, although the improvement is more marked for Group C. Group A, the class that did not use the tutor, actually regressed.

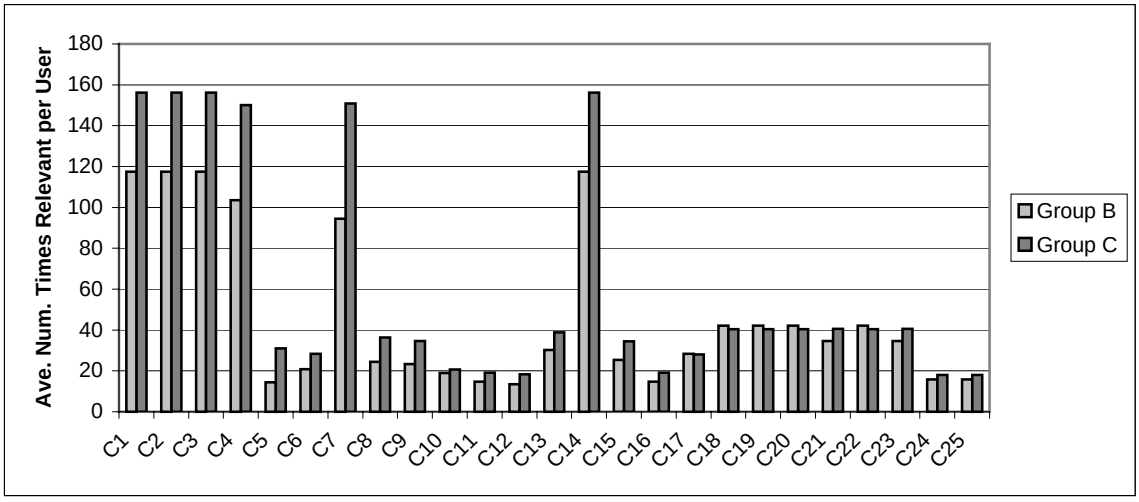


Figure 10. The frequency of constraint relevance to selected problems.

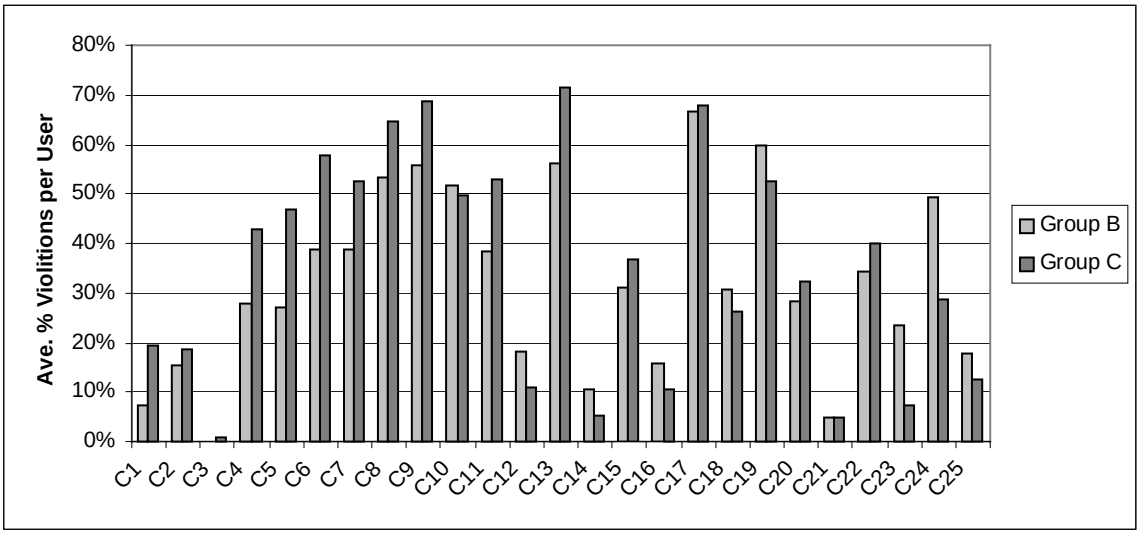


Figure 11. The frequency of constraint violation when relevant.

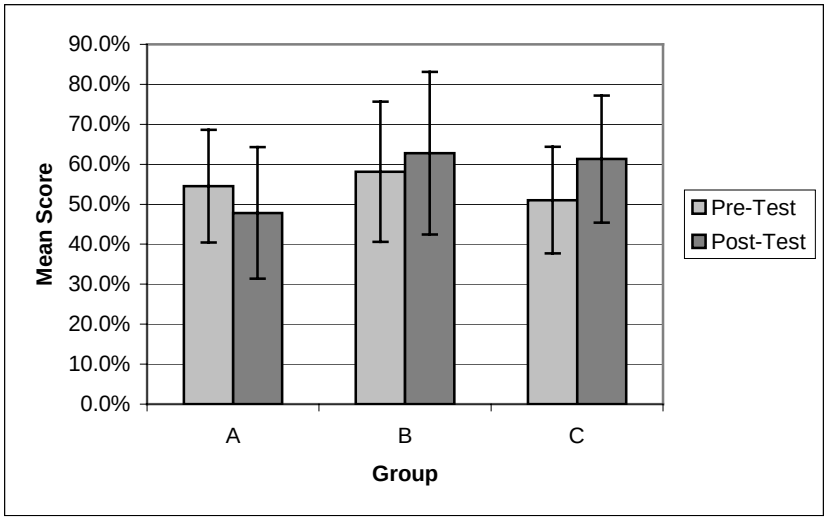


Figure 12. Mean pre- and post-test scores.

Statistical significance tests were also performed to compare the individually matched improvements of Groups B and C from pre-test to post-test. Because the same pair of students in each group completed both a pre- and a post-test, a one-tailed paired difference experiment (McClave & Benson, 1991, pp. 421-7) was performed to gauge the significance of the improvement. With H_0 being the proposition that a group did not improve, it was found that Group B improved with 95% confidence ($\alpha = 0.05$, $t = 1.86$, rejection region ± 1.75) while Group C improved with 99% confidence ($\alpha = 0.01$, $t = 3.4$, rejection region ± 2.6). The improvement is thus much more significant for Group C, which used the decision-theoretic strategies.

We also calculated the effect size, which is defined as the difference in the mean gains of the control (Group B) and experimental groups (Group C), divided by the standard deviation of the mean gain of the control group. This measure gives the magnitude of the change attributable to the intelligent PAS strategy as opposed to the random one. The effect size is 0.557, a value that is comparable to the effect size of 0.63 found by Albacete & VanLehn (2000) after a two-hour session with their tutor. (The average total interaction time in our case was less than two hours for both groups.)

The pre- and post-tests analysis, and the frequencies in Figure 11, confirm that Group C was initially less able than Group B, but learned the constraints at a faster rate. We decided to investigate the constraint violation frequencies further. Each attempt at a problem was analysed, and the total proportion of violated constraints was calculated for each attempt. This was averaged over all students in each group, and the result is depicted in Figure 13. The scatter plot shows that Group C initially made more errors than Group B, but that the rate of constraint violation decreased much faster for that group, supporting the hypothesis that Group C learned the rules of the domain more quickly. Figure 14 shows the results of the same analysis, as an example, for constraints 4 and 7 only, which both depend on the child's cognitive ability to separate the problem text into sentences. The difference is much more marked for these constraints than for the average of all the constraints, but the trend is the same. For both scatter diagrams, a cut-off point of 125 attempts was selected because approximately half of the pairs of students reached this number of attempts, and beyond this number statistical effects arising from the smaller number of pairs tend to corrupt the trend.

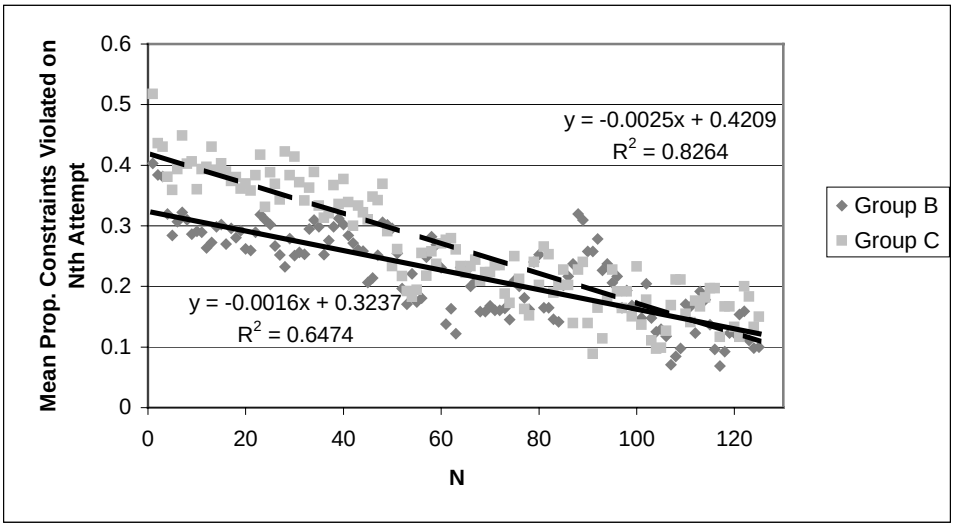


Figure 13. Rate of constraint violation by attempt, for all constraints.

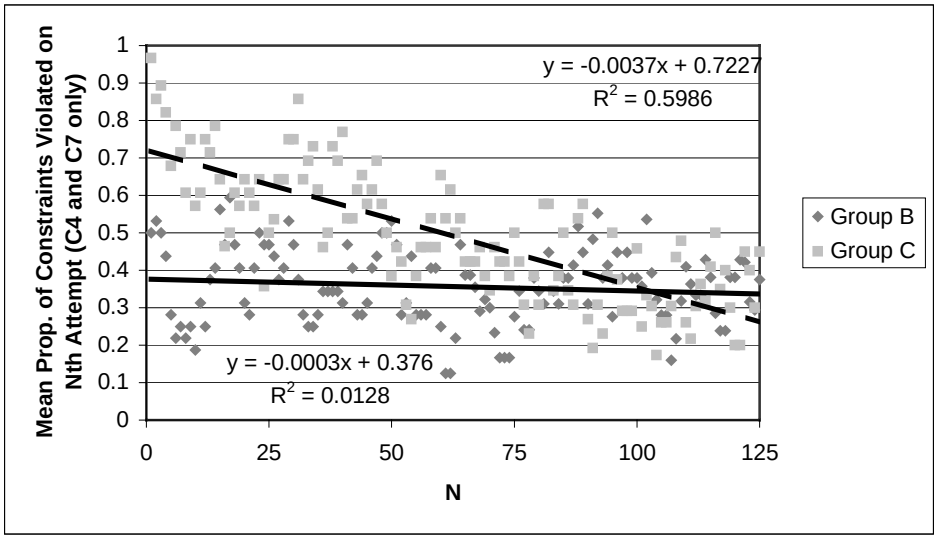


Figure 14. Rate of constraint violation by attempt, for constraint 4 and 7.

Further analysis investigated the mean number of attempts, and the mean time required, to solve the *n*th problem. Figures 15 and 16 show the results of this analysis. Both line graphs

show the same basic trend; Group C was less to able initially, but improved at a faster rate than Group B.

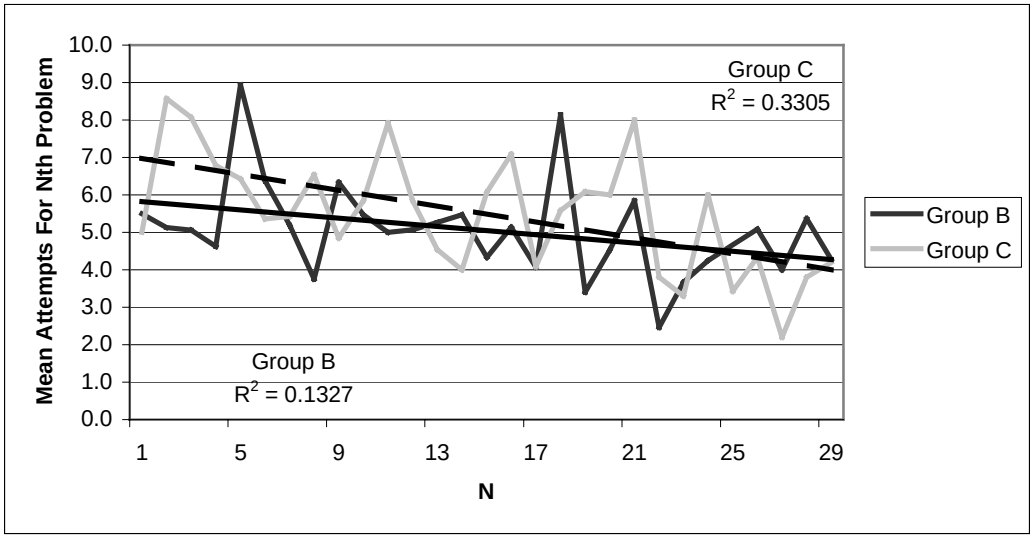


Figure 15. Number of attempts solving the *n*th problem.

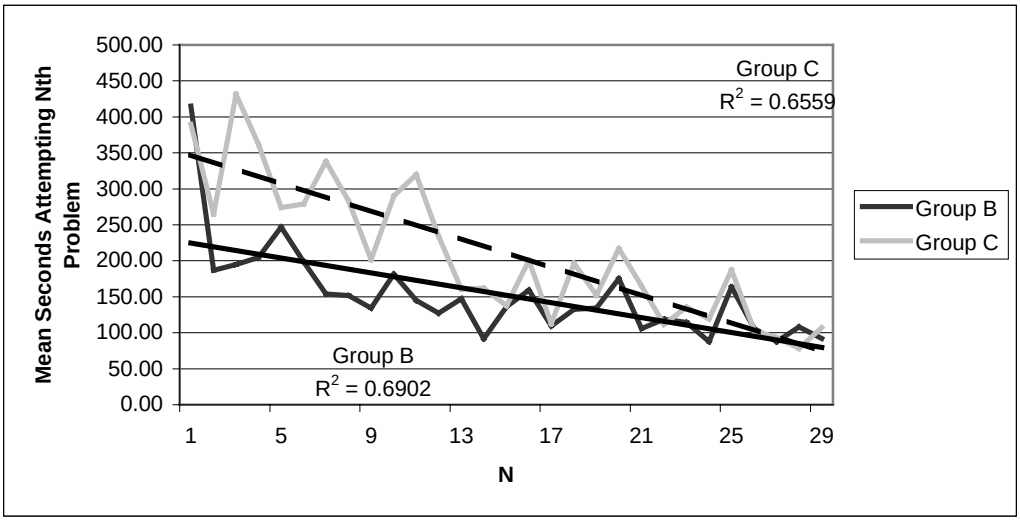


Figure 16. Time required solving the *n*th problem.

To summarise, we have demonstrated that the version of CAPIT with decision-theoretic problem and error message selection, and an adaptive Bayesian network student model, has led to a faster rate of learning than the same system with randomised PAS. This completes the fifth and final step of the application of the general methodology we have proposed to CAPIT.

DISCUSSION AND CONCLUSION

The significant contribution of this paper is the proposal and demonstration of a general methodology for the design and implementation of normative intelligent tutors. CAPIT, a tutor for capitalisation and punctuation, is both a working illustration that decision-theoretic computations in intelligent tutors can be tractable, and evidence that the methodology works. It is therefore possible to build normative intelligent tutors (with Bayesian student models and decision-theoretic PAS) that are guaranteed to be optimal with respect to the normative principles of rational behaviour. We have also made explicit the data-centric approach to

building Bayesian network student models, whereby the structure and conditional probabilities are learned from data, and then continuously adapted on-line to the student. We believe that this approach produces student models better able to predict student performance than both the expert- and efficiency-centric approaches. The data-centric approach also results in more compact student models, because it only explicitly models observable variables.

An interesting conclusion of the statistical comparison between different Bayesian networks architectures in Step 2 is that the *Small* specification produced networks that predicted student performance almost as well as the various large network specifications (the r^2 values varied by at most 0.06). The question arises as to why this is so. An informal analysis of the data collected during Step 1 revealed that, on average, a constraint previously satisfied will be satisfied on the next attempt 91% of the time. This regularity may well explain *Small*'s relatively good performance. However, other domains may not exhibit this degree of regularity. For example, in a domain where the constraints are highly interdependent, the probability of a constraint being satisfied on the next attempt may depend much more on the previous (and current) outcomes of other constraints. On the other hand, *Small* can be expected to outperform the large networks on domains where constraint mastery is wholly or mostly probabilistically independent.

This suggests that there must be some careful justification (e.g. the statistical significance tests performed in Step 2) when a larger, complex model is chosen over a much simpler one. This issue is important, and it should not be skirted over when describing the rationale for a particular intelligent tutor architecture. Furthermore, the relatively good performance of a Bayesian network with no explicit model of the student's internal representation at predicting student performance begs the question of which domains in general are suitable for such an approach. We suggest that suitable domains are those where the concepts are ill-defined, or where different students are expected to conceptualise the domain in different ways, (e.g. constructivist environments). Also, as discussed earlier, domains where the conceptualisation is too complex to produce a simple, tractable model might benefit.

Another advantage of this approach is that it bypasses the problem of prior probabilities in Bayesian networks. VanLehn et. al. (1998) report that different choices of prior probabilities for root nodes in a network can significantly influence the posterior probabilities of other nodes. The workaround suggested by VanLehn et. al. is to treat only the difference between a variable's prior and posterior probability as significant. Our Bayesian model circumvents this problem entirely. Whenever the network is evaluated, the root nodes $L_1..L_{25}$ are always known with certainty because they represent the observed student's history. That is, the causality is always directed from the known ($L_1..L_{25}$) to the unknown ($N_1..N_{25}$), and not the other way around. Therefore we do not even need to maintain priors for these variables.

One issue arising from the design of the evaluation study was that it was inevitable that Group C, which had an intelligent version of CAPIT, would outperform Group B, who were using the randomised version. While it would have made for a better comparison if Group B had been using an alternate intelligent version of CAPIT (e.g. perhaps a version using heuristics to select the next problem and error message), the amount of data required to establish any significant differences between the two strategies would have had to have been much greater, requiring a much more large-scale evaluation study.

One area of concern with this approach is scalability, both to larger domains and different domains. In a larger domain, the space of <*State*, *Action*, *Outcome*> triples may be so large as to effectively render network induction impossible. This may be because the size of the *State* variable is very large (much larger than the 25 constraints modelled CAPIT), or there may be a high number of values that *Action* can possibly take (the limit in CAPIT was 45). A possible solution is to manipulate the learning algorithm to compensate for this additional complexity. For example, if the number of variables in the network is higher, then less numbers of edges should be added to the network during the learning process in order to keep the complexity down. Cheng's (1998) algorithm is flexible enough to perform this. With respect to actions, a possible solution is to divide the set of actions into groups, and then apply decision-theoretic action selection twice: firstly to select the group; secondly to select the action within that group.

This way, the total number of actions being considered will be equal to the number of groups plus the number of items in the selected group.

Scaling the system to different domains is another issue. A limitation of CBM is that constraints must either match or fail to match; the constraint author decides beforehand the conditions of satisfaction and violation. While ambiguity can be encoded into individual constraints (e.g. in CAPIT, commas separating short clauses can be made optional), higher-level ambiguity is not handled by CBM. To illustrate, the possessive pronoun *teachers* could be punctuated to either *teacher's* or *teachers'*. The latter is less likely (unless there is specific contextual evidence that there is more than one teacher), but both are technically correct. CAPIT resolves the problem by accepting only the single most likely solution as the correct solution (*teacher's* in this example), and treating other solutions as incorrect. This is acceptable for a system designed for children, because the system needs to control what its students are actually learning. For example, it is not ideal for a child to continually punctuate possessive nouns such as *teachers* to *teachers'* when the goal of the problem is to teach the correct punctuation of singular possessive nouns. However, in other domains, it may be that the system needs to know when a solution is technically correct. This would require a comprehensive problem-solving module. In a literacy domain, advanced natural language processing would be needed in order to enumerate possible correct solutions. This is beyond the scope of CBM.

To reiterate, the results of the evaluation study are positive and show that the application of normative theories to intelligent tutoring is effective. The log analysis shows that the class using the decision-theoretic version of CAPIT learned the constraints of the domain at a faster rate than another class using the randomised version. The pre- and post-test results support this. Furthermore, on the post-test, both classes outperformed another class that did not have access to the tutor at all.

To conclude, this paper has introduced a methodology for the design of Bayesian long-term student models and decision-theoretic PAS strategies for intelligent tutors. The methodology encompasses a number of new features not present in other systems such as the integration of prior knowledge and data into a single Bayesian network; the learning of both the structure and parameters of the network from data; on-line adaptation; and, decision-theoretic PAS. Furthermore, this methodology is compatible with existing methods for domain knowledge representation and short-term student modelling, such as CBM. The methodology has been found to be effective in the domain of English capitalisation and punctuation, as demonstrated by our new intelligent tutoring system CAPIT.

Acknowledgements

This work has been partly supported by the Department of Computer Science, University of Canterbury, Christchurch, New Zealand. The authors would like to thank Westburn and Ilam Schools, Christchurch, for their participation in the evaluations of CAPIT, Jane McKenzie for assisting us to tailor CAPIT's interface and feedback messages for the schoolchildren, and Microsoft for use of their Microsoft Belief Networks (MSBN) library.

References

Albacete P. and VanLehn K. (2000). The Conceptual Helper: An Intelligent Tutoring System for Teaching Fundamental Physics Concepts. In Gauthier G., Frasson C., and VanLehn K. (Eds.), *Proc. of 5th International Conference on Intelligent Tutoring Systems*, Springer-Verlag, pp. 564-573.

Bauer E., Koller D. and Singer Y. (1997). Update rules for parameter estimation in Bayesian networks. In Geiger D. and Shenoy P. (Eds.) *Proc. of the 13th Conference on Uncertainty in Artificial Intelligenc*e, Morgan Kaufmann Publishers Inc., San Francisco, pp. 3-13.

Bayes, Rev. T. (1763). An Essay Toward Solving a Problem in the Doctrine of Chances, Philos. Trans. R. Soc. London 53, pp. 370-418; reprinted in *Biometrika 45*, pp. 293-315 (1958), and Two Papers by Bayes, with commentary by W. Edwards, Deming, New York, Hafner, 1963.

Beck J. and Woolf B.P. (2000). High-level student modeling with machine learning. In Gauthier G., Frasson C., and VanLehn K. (Eds.), *Proc. of 5th International Conference on Intelligent Tutoring Systems*, Springer-Verlag, pp. 584-593.

Bouwer A. (1998). An ITS for Dutch Punctuation. In Goettle B., Halff H., Redfield C., and Shute V. (Eds.) *Proc. of the 4th International Conference on Intelligent Tutoring Systems*, Springer-Verlag, pp. 224-233.

Brusilovsky, P. (2000). Course Sequencing for Static Courses? Applying ITS Techniques in Large-Scale Web-Based Education. In Gauthier G., Frasson C., and VanLehn K. (Eds.), *Proc. of 5th International Conference on Intelligent Tutoring Systems*, Springer-Verlag, pp. 625-634.

Cheng J., Bell D. and Liu W. (1998). Learning Bayesian Networks from Data: An Efficient Approach Based on Information Theory. On World Wide Web at http://www.cs.ualberta.ca/~jcheng/bnpc.htm

Collins J., Greer J., and Huang S. 1996. Adaptive Assessment Using Granularity Hierarchies and Bayesian Nets. In Frasson C., Gauthier G., and Lesgold A. (Eds.) *Proc. of the 3rd International Conference on Intelligent Tutoring Systems*, Springer-Verlag, pp. 569-577.

Conati C., Gertner A., Van Lehn K., and Druzdel M. (1997). On-Line Student Modeling for Coached Problem Solving Using Bayesian Networks. In Kay J. (Ed.), *Proc. of the 7th International Conference on User Modeling (UM99)*, Springer-Verlag, pp. 231-242.

Corbett A. and Anderson J. (1992). Student Modeling and Mastery Learning in a Computer-Based Programming Tutor. In Gauthier G. (Ed.), *Proc. of the 2nd International Conference on Intelligent Tutoring Systems*, Springer-Verlag, pp. 413-420.

Corbett A. and Bhatnagar A. (1997). Student Modeling in the ACT Programming Tutor: Adjusting a Procedural Learning Model with Declarative Knowledge. In Jameson A., Paris C., and Tasso C. (Eds.), *User Modeling: Proceedings of the 6th International Conference*, pp. 243-254.

Cowell R. (1999). Introduction to Inference in Bayesian Networks. In Jordan M. (Ed.) *Learning in Graphical Models*, pp. 9-26. MIT Press.

D'Ambrosio, B. (1999). Inference in Bayesian Networks. *AI Magazine*, 20(2), pp. 21-36.

Everson H. (1995). Modeling the student in ITS: the promise of a new psychometrics. *Instructional Science*, 23, 433-52.

Ganeshan R., Lewis Johnson W., Shaw E., and Wood, B.P. (2000). Tutoring Diagnostic Problem Solving. In Gauthier G., Frasson C., and VanLehn K. (Eds.), *Proc. of 5th International Conference on Intelligent Tutoring Systems*, Springer-Verlag, pp. 33-42.

Gärdenfors P. (1989). The dynamics of normative systems. In Martino A., *Proc. of the 3rd International Congress on Logica*, Informatica, Dritto, pp. 293-299. Consiglio Nazionale delle Ricerche, Florence 1989. Also published in A.A. Martino (Ed.) *Expert Systems in Law*, Elsevier, pp. 195-200, 1991.

Gertner A. (1998). Providing feedback to equation entries in an intelligent tutoring system for Physics. In Goettle B., Halff H., Redfield C., and Shute V. (Eds.) *Proc. of the 4th International Conference on Intelligent Tutoring Systems*, Springer-Verlag,, pp 254-263.

Gertner A. and VanLehn K. (2000). Andes: A Coached Problem Solving Environment for Physics. In Gauthier G., Frasson C., and VanLehn K. (Eds.), *Proc. of 5th International Conference on Intelligent Tutoring Systems*, Springer-Verlag, pp. 133-142.

Gertner A., Conati C., and VanLehn K. (1998). Procedural help in Andes: Generating hints using a Bayesian network student model. In *Proc. of the 5th National Conference on Artificial Intelligence (AAAI-98)*, MIT Press, pp. 106-111.

Heckerman D. (1999). A Tutorial on Learning with Bayesian Networks. In Jordan M. (Ed.) *Learning in Graphical Models*, pp. 301-354. MIT Press.

Horvitz E., Breese J. and Henrion M. (1988). Decision Theory in Expert Systems and Artificial Intelligence. *International Journal of Approximate Reasoning*, 2:247-302; also On World Wide Web at http://www.auai.org/auai-tutes.html

Jensen, F., Lauritzen, S. and Oleson, K. (1990). Bayesian Updating in Causal Probabilistic Networks by Local Computations. *Computational Statistics Quarterly*, 4, 269-282.

Krause P. (1998). Learning Probabilistic Networks. On World Wide Web at http://www.auai.org/auai-tutes.html

Lauritzen S. and Spiegelhalter D. (1988). Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *Journal of the Royal Statistical Society Series B*, 50, pp. 157-224.

Mayo M. and Mitrovic A. (2000). Using a Probabilistic Student Model to Control Problem Difficulty. In Gauthier G., Frasson C., and VanLehn K. (Eds.), *Proc. of 5th International Conference on Intelligent Tutoring Systems*, Springer-Verlag, pp. 524-533

Mayo M., Mitrovic A. and McKenzie J. (2000). CAPIT: An Intelligent Tutoring System for Capitalisation and Punctuation. In Kinshuk, Jesshope C. and Okamoto T. (Eds.) *Advanced Learning Technology: Design and Development Issues*, Los Alamitos, CA: IEEE Computer Society (ISBN 0-7695-0653-4), pp. 151-154.

McClave J. and Benson P. (1991). *Statistics for Business and Economics* (Fifth Edition). Dellan Publishing Company.

Millán E., Pérez-de-la-Cruz J., and Suárez E. (2000). Adaptive Bayesian Networks for Multilevel Student Modelling. In Gauthier G., Frasson C., and VanLehn K. (Eds.), *Proc. of 5th International Conference on Intelligent Tutoring Systems*, Springer-Verlag, pp. 534-543.

Mislevy R. and Gitomer D. (1996). The Role of Probability-Based Inference in an Intelligent Tutoring System. On World Wide Web at http://cresst96.cse.ucla.edu/CRESST/pages/reports.htm.

Mitrovic, A. and Ohlsson, S. (1999). Evaluation of a Constraint-Based Tutor for a Database. *International Journal of Artificial Intelligence in Education*, 10(3-4), 238-256.

Murray R. and VanLehn K. (2000). DT Tutor: A Decision-Theoretic, Dynamic Approach for Optimal Selection of Tutorial Actions. In Gauthier G., Frasson C., and VanLehn K. (Eds.), *Proc. of 5th International Conference on Intelligent Tutoring Systems*, Springer-Verlag, pp. 153-162.

Murray W. (1998). A Practical Approach to Bayesian Student Modeling. In Goettle B., Halff H., Redfield C., and Shute V. (Eds.) *Proc. of the 5th International Conference on Intelligent Tutoring Systems*, Springer-Verlag, pp. 424-433.

Murray W. (1999). An easily implemented, linear time algorithm for Bayesian student modeling in multi-level trees. In Lajoie S. and Vivet M., *Proc. of the 9th International Conference on Artificial Intelligence and Education (AI-ED 99*), IOS Press, pp.413-420.

Ohlsson S., (1994). Constraint-based Student Modeling. In: Greer, J.E., McCalla, G.I. (eds.): *Student Modeling: the Key to Individualized Knowledge-based Instruction*, pp. 167-189. NATO.

Pearl J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference* (revised 2nd edition). Morgan Kauffman, USA

Reye J. (1998). Two-Phase Updating of Student Models Based on Dynamic Belief Networks. In Goettle B., Halff H., Redfield C., and Shute V. (Eds.) *Proc. of the 4th International Conference on Intelligent Tutoring Systems*, Springer-Verlag, pp. 274-283.

Russell S. and Norvig P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice-Hall.

Savage L. (1954). *The Foundations of Statistics*. Wiley.

Shafer, G. (1986). Probability judgement in artificial intelligence. In L. Kanal and J. Lemmer (Eds.), *Uncertainty in Artificial Intelligence*. New York: North-Holland.

Stern M., Beck J., and Woolf B. (1999). Naïve Bayes Classifiers for User Modeling. Center for Knowledge Communication, Computer Science Department, University of Massachusetts.

VanLehn K., Niu Z., Siler S. and Gertner A. (1998). Student Modeling from Conventional Test Data: A Bayesian Approach without Priors. In Goettle B., Halff H., Redfield C., and Shute V. (Eds.) *Proc. of the 5th International Conference on Intelligent Tutoring Systems*, Springer-Verlag, pp. 434-443.

Vigotsky L.S. (1978). *The development of higher psychological processes*, Cambridge, MA: Harvard University Press.

Zadeh, L. (1983). The role of fuzzy logic in the management of uncertainty in expert systems. *Fuzzy Sets and Systems*, 11, 199-227.