ITEAMS - an Intelligent Teaching Environment

Dan Dentinger  Matthias Scheutz
University of Notre Dame  Notre Dame, IN 46556
e-mail: {ddenting,mscheutz}@nd.edu

Abstract
In this paper, we introduce an intelligent teaching environment, which uses dynamically created belief networks to assess students' performance and provides a mechanism for automatic grading of agent-based assignments. Specifically, course instructors can specify the teaching materials including assignments at different levels of difficulty. The system, in turn, selects appropriate parts of the materials as well as agent-based assignments based on students' past performance on exercises and tests.

Introduction
Agent-based programming assignments that require students to specify and implement agent architectures for simulated and robotic agents have become an important aspect of courses in artificial intelligence and even more so in behavior-based robotics (e.g., (Arkin 1998)). Such programming assignments can be very instructive as they help students to appreciate the difficulties connected with making agents perform even simple tasks properly. In introductory artificial intelligence courses, they can also contribute to a better understanding of the various subfields in AI are related (e.g., perception, reasoning, planning, etc.).

However, there are at least two major, related difficulties connected to agent-based programming on the instructor’s side: (1) to be able to maximize the benefit for each student, the instructor will have to provide individual feedback and follow-up assignments that are targeted at weaknesses found in previous ones, and (2) grading agent-based assignments requires a substantial support system on the part of the instructor if the goal is to correct students’ approaches and give them feedback that will improve their understanding of agent architectures. We believe that an intelligent teaching environment that can monitor students’ progress in agent-based assignments, adapt to the students’ current level of knowledge and expertise, and assess the quality of their solutions as well as their knowledge level would be of great benefit to such courses and likely have utility beyond.

In this paper, we introduce a first attempt at developing an Intelligent Teaching Environment with Assessment Modules for Self-Study, called ITEAMS, which addresses the above problems. After a brief summary of the functional requirements of ITEAMS, we present an overview of the ITEAMS architecture together with a functional specification of its components, in particular, the assessment and grading components, which are based on a knowledge base and a belief network. We conclude with a short discussion of the current state of implementation and the future plans of development of the ITEAMS system.

Background and Motivation
The main goal of ITEAMS was to integrate important aspects of teaching environments and tutoring systems, such as the infrastructure to organize lecture materials, keep grades, and store any other course-related information from the former, and to be able to track and assess students’ performance, to dynamically select lecture materials based on the students’ inferred knowledge level, and to grade assignments automatically.

There are several commercial teaching environments available for instructors to use in conjunction with their courses (e.g., Angel, Blackboard, WebCT, and others). These systems allow instructors to create presentations, lecture notes, and quizzes, but lack certain abilities including instructional styles, lectures tailored to the needs of the students, or automatic assessment of the students. ITEAMS incorporates these features which teaching environments lack and adds the ability to give and evaluate assignments.

Intelligent tutoring systems (ITSs) can be placed in two groups: systems that are application or field-specific (Zhou & Evens 1999) and systems that can be used for multiple fields (Nkambou 1999). Since many tutoring systems are designed for a specific application, special care has been taken to make ITEAMS as general as possible to be applicable in many different subject areas.

Most recent research has focused on how to model the student and (not as much) on how to present the material to the student. Patel (Patel 1996b) pointed out that current ITSs do not model the teacher and emphasized that there needs to be an increased focus on the teacher model in ITSs.
Other researchers besides Patel, have brought to light the need to pay more attention to the presentation of material and teaching style (Conati 2002; Self 1990). Existing systems have attempted to handle multiple styles of teaching, but have typically been restricted to a limited set of styles. ITEAMS, on the other hand, allows instructors to incorporate their own style of teaching into the modules. This draws upon the instructors' experience as an educator and exposes students to a possibly more familiar learning environment. Consequently, there is no explicit instructional model in our system, although there is support for structuring the lecture materials.

Every learning environment needs some student evaluation schema. Evaluation schemas range from neural networks (Magoulas, Papanikolaou, & Grigoriadou 1999) to Bayesian networks (Bunt & Conati 2002). Others have implemented a more straight-forward grading approach such as the Marker system (Patel 1996a). ITEAMS uses a combination of Bayesian networks to assess the students' performance and knowledge level.

ITEAMS uses several types of belief networks to evaluate, assess, and progress the student through the material. These belief networks consist of (1) singly connected networks, (2) multiply connected networks, and (3) structurally dynamic networks. Each type of network is required for different situations and at any given moment information will be obtained from at least one of each type of network. One goal of ITEAMS is to assist in the learning process of a student; hence, the time required to perform queries and update the network based on students' actions must be minimal and efficient. In order for each type of network to meet these requirements most efficiently, it is important to consider how and why each type of network is used.

The ITEAMS Architecture

ITEAMS consists of two major components: (1) a student model and (2) a model of the domain. The student model contains information about the students that is used in their assessments. The model of the domain contains the material for the module and possible instructional paths (i.e., sequences of presentation) through the material. ITEAMS also contains several sub-components: (1) a teaching module interface, (2) an external interface, and (3) an intelligent assessment component. The teaching module interface allows instructors to prepare and structure lecture material for each teaching module and allows the student to work with each teaching module. The external interface allows ITEAMS to connect to other systems through the use of “plugins” (i.e., purpose-written interfaces that handle the data exchange between ITEAMS and other systems). The ability to allow students to use external systems (e.g., a robot simulator) for assignments and to gather data from that system about the students' performance is unique to ITEAMS. Finally, assessing the students' overall performance and level of knowledge (to the extent that this can be measured with the methods employed in ITEAMS) is accomplished by causal queries of the various belief networks.

Teaching Module Interface

The teaching module interface gives instructors the freedom to create modules containing lecture materials, quizzes, and assignments arranged in the instructor's teaching style. We have defined a simple graphical user interface for the teaching module, which allows the instructor to design the material in terms of instructional categories, teaching modules, sections, and goals and associate them with each other.

Instructional categories are instructor-specific objectives or learning categories that are used to help assess students' performance. These categories are used by quizzes and assignments to direct how a question or exercise relates to the material and overall performance of a student. An introductory course in programming could have categories of implementation, programming language knowledge, analytical ability, etc.

The instructor can simply define and structure the sequence of lecture material and instructional goals in order for ITEAMS to be able to present the material to the student in a coherent manner. The arrangement is accomplished graphically by “dropping” a section, quiz, or assignment into a teaching module and then adding the necessary instructional material.

Sections are created by adding learning objectives, goals (which, if reached, will accomplish the objectives), quizzes, and assignments. Since ITEAMS attempts to tailor lecture materials to meet students' learning needs, materials can be added at different levels of difficulty. This allows instructors to provide a greater degree of flexibility in the students learning progress. Each goal represents a lecture or lecture topic, allowing one or more goals to be accomplished in a given session. Each section can have a quiz and an assignment associated with it which are presented to students after they have completed all of the section's goals. Each goal can also have a quiz and/or assignment associated with any or all of the levels of material.

Currently, quizzes allow one type of question. Questions are in multiple choice format where each answer...
has some “percentage correct” associated with it and can be specified by: (1) giving the question text, (2) which instructional categories it references, (3) the difficulty of the question, and (4) any answers with their associated correctness.

Assignments allow multiple parts which are separated by exercises. An exercise is very open. It can be a question that students need to answer or it can be a programming problem that requires an external system (e.g., a simulation environment), which is accessed through a plugin. As more types of questions are added to ITEAMS, more will become available for exercises. Exercises using external systems via plugins allow the instructor to specify parameters of the external system that will be used for grading (see the next subsection).

Once one or more sections have been created, an instructor links sections together to create the flow of information and depict instructional dependencies in the material (e.g., one section should come before another).

### External Interface

The external interface allows instructors to use applications, environments, etc. for their course and subsequently allows students to interact with them from within ITEAMS. For each application, a separate “plugin” has to be defined and implemented, which translates the ITEAMS external interface format into a format recognized by the external application (e.g., via the Component Object Model or external function calls in MATLAB), if it supports information exchange.

The plugins provide several integral functions. They determine the kinds of assignments that are possible based on the parameters they provide, and they also provide the means to grade/evaluate the students’ performance on exercises based on these parameters. In general, they are used to determine the students’ performance on an assignment.

A typical use of the plugin is as follows: first, the plugin will start an external program or environment and load any instructor-provided files the exercise requires. Students must provide the name of an output file that is created once they have finished the exercise (i.e., the file that contains the “results” of the exercise—what these results are will depend on the type of exercise; for a programming exercise, it will simply be the code written for it). The contents of the file are what the plugin subsequently uses for assessing the student.

In order for a plugin to assess the progress of students, the instructor must provide a metric for the solution. This is done by specifying ranges in the data that the plugin reads from the output generated by an external application. For example, an agent-based programming exercise asks the student to design an architecture using a particular design paradigm for a virtual agent that makes it search for a target location, while avoiding obstacles. The evaluation could then be based on the agent’s performance: how fast the agent reaches the target location and how many collisions it had on the way there. The instructor would only have to specify the ranges and percentages for both dimensions and the corresponding percentage of the score that is awarded if the respective criteria are met (see Table 1).

<table>
<thead>
<tr>
<th>Collisions</th>
<th>Time-to-Target</th>
<th>Correctness</th>
</tr>
</thead>
<tbody>
<tr>
<td>x ≤ 1</td>
<td>x ≤ 100</td>
<td>1.0</td>
</tr>
<tr>
<td>1 &lt; x ≤ 5</td>
<td>x ≤ 100</td>
<td>0.75</td>
</tr>
<tr>
<td>≤ 1</td>
<td>100 &lt; x ≤ 500</td>
<td>0.5</td>
</tr>
<tr>
<td>1 &lt; x ≤ 5</td>
<td>100 &lt; x ≤ 500</td>
<td>0.25</td>
</tr>
<tr>
<td>5 &lt; x</td>
<td>500 &lt; x ≤ 500</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 1: Sample ranges for plugin

The only requirement for an external application is that it can produce an output that contains information that can be evaluated by a plugin. Plugins are designed to adhere to certain properties. They need to accept learning parameters from ITEAMS— which are partially generated by ITEAMS and partially specified by the instructor - start any external environment, prepare it for the students’ use, and use the parameters specified to assess students’ performance.

### Performance Assessment

There are several “decisions” ITEAMS must make during the course of an instructional session: (1) what different levels of knowledge the student has obtained, (2) what level of material to present, (3) the order in which quiz questions are presented, (4) how many questions (or exercises) are presented, and (5) which exercises to present in an assignment. These decisions, among others, can take place at different times and rely on several different aspects of students’ performances. All of these decisions are made by belief networks. These networks are dynamically constructed, either in part or in full, for each teaching module from one or more of the following: (1) the number of possible questions, (2) the possible assignment exercises, (3) the number of goals included, and (4) the material being presented.

All inferences performed by the belief nets are causal inferences of the form: \(P(\text{Query}|\text{Evidence}_1 \land \text{Evidence}_2 \land \cdots \land \text{Evidence}_n)\). Inference and update of singly connected belief networks uses an implementation of Pearl’s message passing algorithm (Pearl 1988). Multiply connected networks use an implementation of stochastic simulation (Pearl 1988).

### Structurally Dynamic Belief Networks

One of the types of belief networks that ITEAMS uses is a derivative of dynamic or temporal belief networks (Murphy 2002; Neapolitan 2004). In conventional dynamic belief networks, there is a defined set of nodes which make a temporal slice and through time the network changes the link structure between nodes as new evidence appears. Not only can the link structure between the nodes in one temporal slice change, but links can form between nodes in adjacent temporal slices.

Structurally dynamic belief networks (SDBNs) differ from the previous type in two distinct ways. The first difference is that the structure of the network changes
in each temporal slice. The second difference is there are no links between the temporal slices. Figure 2(A) demonstrates a temporal slice in an SDBN network and Figure 2(B) shows the network in the next time slice. SDBNs can be created either as singly connected networks or multiply connected networks. If the network being created is singly connected then an addition to the network requires two steps. The first step adds a node to the network and links it to it’s parent or parents. The second step is to calculate the proper conditional probabilities for its parent(s). If the network being generated is multiply connected, the same two steps are needed to add a node to the network. Once an addition has been completed, the network can be updated and queried as new evidence is found.

As mentioned above, ITEAMS currently uses an implementation of Pearl’s message passing algorithms (Pearl 1988; Neapolitan 2004) for singly connected SDBN networks. Since these algorithms already handle passing messages among nodes, the process of dynamically adding nodes and notifying the other nodes becomes much less difficult. Multiply connected SDBN networks use an implementation of stochastic simulation (Pearl 1988; Russell & Norvig 1995) that adds the notification scheme for the addition of new nodes.

**Student Performance with Belief Networks** To determine the students’ different knowledge levels, ITEAMS uses two different types of belief networks simultaneously. One belief network can either be a singly connected network or a multiply-connected network (depending on the teaching module structure) while the other type is a SDBN. These SDBNs are composed of all the questions and exercises a student has been presented in the current teaching module.

ITEAMS creates separate networks to simplify some of the evaluation of instructional categories. If more than one category exists, a belief network consisting of only questions and exercises that pertain to a given category is created. Thus if there are four categories, five belief networks are created, one for each of the categories and one representing the teaching module. This is done because the belief networks can become very large, and inferences can become slow as the size grows. The SDBNs are constructed from the questions and assignment exercises contained in a teaching module. As a student works on a question or exercise, it is added to each category SDBN to which it is related. The conditional probability tables (CPT) are also dynamically created from the “percent correct” value of each question’s answers. Instructional categories can then be evaluated by calculating \( p(\text{Mastery of Category } X | \text{Set of all questions and exercises related to Category } X) \).

A belief network representing the teaching module is also created consisting of nodes representing student performance in sections, quizzes, and assignments in an attempt to give a more accurate overall evaluation of the student than just the average of the categories’ performances. For efficiency reasons a singly connected network is used for teaching modules unless the teaching module structure is multiply connected. Each nodes’ CPT contains five values which represent normal grade ranges, such as A or 100% - 90%. The value for a node is determined by the student’s performance in the associated section, quiz, or assignment.

ITEAMS also performs inferences to determine an area of deficiency and suggest a review of previous sections or goals. Deficiencies are determined on a category basis, and when a deficiency is found, a student will be alerted and a review can be suggested.suggesting a review requires determining where the deficiency began by querying a category’s belief network on a section by section basis. In other words, ITEAMS attempts to answer the question: “Does the deficiency exist given their performance on a particular section?”

As will be discussed below, many different pieces of information affect the conditional probability tables in the network. Some of these include: previous answers given for the current question, number of times the current question has been seen, and number of times a related goal has been reviewed. All of this information updates the conditional probability tables by either increasing or decreasing the probability that a student has mastered the material.

**Lecture Material and Belief Networks** Since different students learn best in different ways and at different levels of abstraction, instructors can define multiple levels at which the material can be presented to students. ITEAMS, in turn, will choose which level of material to present to the student at any given time as determined by a belief network. The belief network is structured to model the flow of information through the goals in each section. In the simplest case, the roots of the network are the different levels of material for the first goal, which are connected to all levels of the next goal, and so on until all the sections’ goals are incorporated. In more complex cases, the network’s roots are the nodes which influence each goal. Figure 3 shows the basic structure of the belief network used to evaluate students’ progress in sections.

Currently, there are four levels of material supported: overview, intermediate, proficient, and complete. The material for each level does not differ in content or subject matter, but increases in depth as the level increases. The decision as to which level of material to present is determined by the current performance of the student for a given goal along with the performance of the student for any related goals. Students’ performance on
Figure 3: Basic format for section belief networks

A goal is determined by their performance on an associated quiz or assignment. If no quiz or assignment is associated with a goal, the goals performance is determined by the overall performance for the section.

To determine the next goal’s level of difficulty of lecture material, ITEAMS updates the network with the student’s performance on the last quiz or assignment. Once the update has taken place each level of difficulty in the next goal is ranked based on the updated probabilities. If ITEAMS determines the student can progress through or perform as well on the next goal at a level of difficulty greater than the current level, the student’s current level of difficulty is increased. If it is determined that the student cannot progress at the current level or higher the student’s difficulty level is lowered.

Quiz Belief Networks  ITEAMS uses a set of rules in conjunction with the difficulty and instructional categories to create the network structure for quizzes. The rules used to create the network are (1) if an instructional category is associated with two questions of the same difficulty, the nodes are linked, (2) when linking two questions of the same difficulty level, the question encountered first becomes the parent, and (3) a question of difficulty level i+1 if and only if they are both associated with the same instructional category. The second rule ensures that the network is a directed acyclic graph. The nodes for questions and assignment exercises use the correctness values of the possible answers as the nodes values. Thus a question with four possible answers would become a node with four possible values.

Figure 4: A possible multiply connected belief network for a quiz or assignment

To illustrate the construction of a network, suppose there are two difficulty levels: one with five questions, and one with four questions. If the teaching module has five instructional categories, and a maximum of four questions sharing each category, ITEAMS can construct the network shown in figure 4. This network contains loops on both the difficulty level as well as between the difficulty levels. Since the networks are constructed to preserve dependencies between questions the formation of loops in the network becomes extremely difficult to avoid. Since ITEAMS constructs these networks while a student is working on a teaching module, ITEAMS is not able to spend much processing time on optimizing the structure of the networks. Once the network is constructed, it is updated after each question is answered. Once the update is complete, ITEAMS uses a ranking heuristic to determine which questions the student is most likely to get correct and then randomly chooses one of these questions as the next question to present to the student. The questions are ranked by computing the probability that the student understands the material based on the question answer, p(Mastery of Material|Current question Q).

Assignment Belief Networks  Like quizzes, the assignment belief networks must determine which and how many exercises to present to a student based on overall performance. Even though exercises can be in different formats (question or plugin), these belief networks are formed the same way as the quizzes.

New evidence is given to ITEAMS from one of two places: either student responses or plugin responses to an exercise. Plugins must provide a percentage correct back to ITEAMS in order to correctly be evaluated in the belief network. Note that for questions to be answered directly in ITEAMS the answers already have the percentage correct associated with them.

Student Model Description

ITEAMS requires different levels of student information to make decisions within the system. This information ranges from qualitative information on the progress of students to quantitative data used to calculate their level of mastery of the material. All of the information recorded is used to help construct the belief networks or is used to update the conditional probability tables of the several belief networks in ITEAMS.

Student Solution Records  It has been suggested that knowing the history of a student’s answers to questions can serve as a metric in evaluating the student’s knowledge level (Zhou & Evens 1999). ITEAMS makes use of this idea based on the two different answers it tracks: question answers and exercise solutions. The record of a student’s answer to a question is comprised of the actual answer and the quantitative value of the answer. In the case of an assignment, the record consists of any information a plugin gives to ITEAMS and a qualitative assessment of the performance. Since students can review the material as often as they want, they could receive the same question or exercise more than once when retaking a quiz or assignment. In this case, ITEAMS modifies the values in the CPTs of the questions and exercises the student has already answered to reflect the student’s previous performance.
We are currently evaluating different ways to update the values.

Goals Assessment The student model keeps track of the student’s quantitative assessment for each section and goal respectively. It also keeps track of the student’s overall quantitative assessment. This allows the instructor to quickly see a student’s performance without having to re-evaluate the student each time.

Current State ITEAMS also keeps track of its current state per student. This state consists of the current active section, active question if the student stops halfway through a question, and what goal of the section is active. Keeping this information up-to-date allows students to exit ITEAMS at any time and continue later at the point where they left it. The state is specific to each teaching module, which allows the student to work on and switch between multiple teaching modules.

Discussion

Instructors can use ITEAMS in at least three different ways or any combination thereof: (1) as an organizing tool for instructors that allows them to map out their courses and keep track of their students data (such as grades, completed assignments, etc.); (2) as an assessment tool that allows them to define assessment criteria for the automatic determination of the levels of mastery of their students for a given subject; and (3) as a supervision tool that will save them time supervising students while the students are working on assignments.

Students can use ITEAMS in also in at least three different ways or any combination thereof: (1) as an organizing tool for course materials and assignments; (2) as an exercise tool that grades their assignments and provides immediate feedback as well as additional exercises targeted at their strengths and weaknesses; (3) as an assessment tool that gives them feedback as to their level of mastery of a subject (e.g., in the course of studying for an exam).

ITEAMS is currently in its final testing phase and will find application in a real-world setting for the first time in Spring 2004, where it will be used for a teaching module on robotics architectures in a behavior-based robotics course (at our institution).

The current implementation of ITEAMS still has several limitations, such as limited types of questions and students’ affective information. We plan on integrating more question types into ITEAMS, such as fill in the blank and short answer, as well as a set of plugins for standard software systems (e.g., Matlab). We also see much room for improvement at the level of the student model. Affective information, however, may prove useful in selecting particular problem sets or in providing help to students (e.g., it may be instructionally more effective to switch to a different problem set or to provide more clues for students who are already irritated or agitated because they cannot solve their assignment). Furthermore, information about the students’ personality profile (i.e., the addition of a personality model based on a questionnaire) could help in the self-adaptation of the system to particular learning and working styles (e.g., a patient, pragmatically oriented student may prefer to spend more time working out details of the programming code than would a short-on-patience, conceptually oriented student).

References

Patel, K. 1996b. Intelligent tutoring tools: Redesigning its for adequate knowledge transfer emphasis.