Using POMDP in Building an Adaptive Intelligent Tutoring System

Fangju Wang
School of Computer Science, University of Guelph
Guelph, Ontario, Canada N1G 2W1
fjwang@uoguelph.ca

Abstract

An intelligent tutoring system (ITS) can teach students in a one-to-one, interactive way. It may help students achieve their learning goals better than classroom lecturing. An ITS should be able to teach adaptively based on knowledge states of students. Uncertainty is a challenge in developing an ITS. In practical tutoring, student information available to a teacher may be incomplete and uncertain. The partially observable Markov decision process (POMDP) model provides useful tools for handling uncertainty. It enables an ITS to take optimal teaching actions even when uncertainty exists in tutoring processes. In this paper, we reported an experimental ITS developed on the POMDP model. We describe the definitions of states, actions, observations in the POMDP framework, and the techniques for dealing with exponential state space and POMDP solving, which are major barriers in building POMDP based ITSs for practical applications.

keywords: Intelligent system, computer supported education, partially observable Markov decision process.

1 Introduction

One-to-one, interactive tutoring can help students achieve their learning goals better than conventional classroom lecturing [1]. Intelligent tutoring systems (ITSs) aim to offering the benefits of one-to-one interactive tutoring without the costs of dedicating one human teacher to each student. ITSs have been developed as teaching aids in many fields, including mathematics, physics, medical science, astronaut training, and web-based adult education [13, 8].

Personalization and intelligent tutoring are two crucial elements in computer supported education. For good teaching performance, a system should provide customized instruction according to each individual student’s knowledge states at the time of learning. To achieve this, an ITS maintains a student model and a tutoring model. When teaching a student, the system uses the student model to reason about the student’s current states, and applies the tutoring model to choose the optimal teaching actions.

Uncertainty exists in tutoring processes. Even a human teacher may not be certain about the student’s states and the most beneficial tutoring actions [13]. The Partially observable Markov decision process (POMDP) model offers techniques to deal with uncertainty caused by incomplete observation of states. In our research, we developed an experimental ITS. We casted its components onto the POMDP framework.

In building a POMDP-based system for a practical application, there are two challenging tasks. One is to deal with the state space, which is typically exponential, and the other is POMDP-solving, which is typically exponential as well. In this paper, we first introduce the technical background, and review the related work. Then we describe the ITS, emphasizing the techniques we develop for dealing with the state space, and solving the POMDP. Finally we present our experimental results.

2 Technical Background

2.1 Intelligent tutoring systems

An ITS is a computer system that teaches a subject, usually to a student at a time, in an interactive manner. It provides immediate and customized instruction or feedback to the student. ITSs aim to offer the benefits of one-to-one interactive tutoring with lower costs than human teachers.

The major components of an ITS include a domain model, a student model, a tutoring model, and a user interface. The domain model contains the domain knowledge. It is the source of knowledge, a standard for evaluating student performance or for detecting errors.

The student model contains information about the knowledge states, and learning needs of each student [13]. It may also contain a student’s affective states. A student model is dynamic in the sense that it is updated with the student’s progress. The tutoring model represents the pedagogical strategies of the ITS. It takes the student’s current states and learning requests as input,
and selects the optimal tutoring actions as output.

ITS technology addresses the two major principles of cognitive research in human tutoring, namely tracing knowledge and adaptive instruction. Integrated use of the student model and tutoring model helps equip an ITS with the capabilities of timely tracing knowledge and adaptively responding to students, which distinguish ITSs from traditional computer based training.

2.2 Partially observable Markov decision process (POMDP)

POMDP is an extension of MDP (Markov decision process) for handling uncertainty. POMDP does not require that the agent must know exactly what the current state is. It allows the agent to make decisions based on its beliefs about the states.

POMDP can be represented as tuple \((S, A, T, \rho, O, Z, b_0)\). \(S\) is a set of states, \(A\) is a set of actions, \(T\) is a set of state transition probabilities, \(\rho\) is a reward function, \(O\) is a set of observations, \(Z\) is a set of observation probabilities, and \(b_0\) is the initial belief.

A belief is denoted by \(b\):

\[
b = [b(s_1), b(s_2), ..., b(s_N)]
\]

where \(s_i \in S\) \((1 \leq i \leq N)\) is the \(i\)th state in \(S\), \(N\) is the number of states in \(S\), \(b(s_i)\) is the probability that the agent is currently in \(s_i\), and \(\sum_{i=1}^{N} b(s_i) = 1\).

In a POMDP, a policy is used to guide the agent to take actions. It is of the form \(\pi(b)\). The task of finding the optimal policy is referred to as POMDP-solving.

At \(t\), the decision agent is in \(s \in S\). Since states are not completely observable, the agent has only the belief \(b\) about the states. With the guidance of \(\pi(b)\), the agent chooses \(a \in A\) to take. The action causes state transition into new state \(s' \in S\). At \(t + 1\), the agent observes \(o \in O\) that is emitted by \(s'\), and has new belief \(b'\). \(b'\) is a function of \(b, a, o, \) and \(s'\):

\[
b'(s') = \sum_{s \in S} b(s)P(s'|s, a)P(o|a, s')/P(o|a)
\]

where \(P(s'|s, a)\) and \(P(o|a, s')\) are transition and observation probabilities, \(P(o|a)\) is the total probability to observe \(o\) after \(a\) is taken, used as a normalization.

Typically, POMDP-solving is a task of great complexity. A practical method is to choose a policy tree, in which nodes are actions, and edges are observations. When a policy tree is executed, the agent first takes the root action \(a_r\), observes \(o\), then takes the action at level 2 that is connected by the edge of \(o\), and so on.

In the policy tree method, in a decision step, the agent chooses the optimal policy tree based on the current belief. A policy tree is associated with a value function. We denote the value function of \(b\) with policy tree \(\tau\) by \(V^\tau(b)\). The optimal policy tree for \(b\) is the \(\tau\) that maximizes \(V^\tau(b)\). \(V^\tau(b)\) is calculated from \(V^\tau(s)\), the value function of state \(s\):

\[
V^\tau(s) = R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) \sum_{o \in O} P(o|a, s') V^{\tau(o)}(s')
\]

where \(a\) is the root action of policy tree \(\tau\), \(o\) is the observation after \(a\) is taken, \(\tau(o)\) is the subtree in \(\tau\) which is connected to the root by the edge of \(o\), \(\gamma\) is a reward discounting factor, and \(R(s, a)\) is the expected immediate reward after \(a\) is taken in \(s\).

Based on the value function of state \(s\) given \(\tau\), we define the value function of belief \(b\) given \(\tau\) as an expectation over all the states:

\[
V^\tau(b) = \sum_{s \in S} b(s)V^\tau(s).
\]

When the agent uses \(\pi(b)\) to choose an action, \(\pi(b)\) returns a policy tree that maximizes the value of \(V^\tau(b)\):

\[
\pi(b) = \hat{\tau} = \arg \max_{\tau \in T} V^\tau(b),
\]

where \(T\) is the set of policy trees.

3 Related Work

The POMDP model has been applied to computer supported education since 1990s [2]. In the early work, POMDPs were applied to model internal mental states of individuals, and to find the best ways to teach concepts. In recent years, extensive research activities were conducted in using POMDP for intelligent tutoring. In the following, we review some representative work.

SPAIS (Socially and Physically Aware Interaction Systems) was a teaching system based on POMDP [6]. Optimal teaching with SPAIS corresponded to solving an optimal policy in a very large factorial POMDP. To address the difficult problem of POMDP-solving, the researchers developed a policy-switching method among simpler solutions, each representing the best way to teach a type of students.

In the work reported in [5], the researchers developed a technique of faster teaching by POMDP planning. They framed the problem of optimally selecting teaching actions by using a decision-theoretic approach, and formulated teaching as a POMDP planning problem. In the POMDP, the states represented the learners’ knowledge, the transitions modeled how teaching actions changed the learners’ knowledge, and the observations indicated the probability that a learner would give a particular response to a tutorial action.
The work described in [4] was aimed at making POMDP solvers feasible for real-world problems. In the work, a data structure was created to describe the current mental status of a particular student. The status was made up of knowledge states and cognitive states. The knowledge states were defined in terms of gaps, which are misconceptions regarding the concepts in the instructional subject. The system actions were performed to tutor gaps. Observations were indicators that certain gaps and states were present or absent. To address the problem of exponential state space, the researchers developed two scalable POMDP state and observation representations.

Research work related with applying POMDPs for intelligent tutoring also included [12], [7], and [3]. The work was characterized by using POMDPs to optimize teaching, but varied in the definitions of states, actions, and observations, and in POMDP-solving.

4 An Overview of the ITS

Our experimental system is for teaching concepts. The subject is basic knowledge of software. In a tutoring session, the student asks questions about the concepts, and the system answers the questions.

A goal of developing this system is to achieve adaptive responses based on student knowledge states. In many subjects of mathematics, science and technologies, for example software basics, a concept may have prerequisites. To understand a concept well, a student has to understand its prerequisites first. When a student asks a question about a concept, he/she may or may not have studied all the prerequisites. If the teacher directly teaches the concept without teaching the prerequisites that the student has not studied, the student will have difficulties to understand the concept asked. If the teacher always works in this way, the student may become frustrated, and the teaching would be ineffective. On the contrary, if the teacher teaches lots of prerequisites that the student has already studied, the student may become impatient and the teaching would be inefficient. To be effective and efficient, a teacher must be adaptive to the student. When responding to a question about a concept, the teacher should choose a “right” concept to start with.

To achieve the goal of adaptive teaching, in the ITS we created a student model to trace individual students’ knowledge and progress, and also to model the general students’ behavior, a tutoring model to provide customized instruction according to each individual student’s knowledge states. Since a student’s states may not be completely observable to the teacher, we use the POMDP model to handle the uncertainty.

We cast our ITS onto the framework of POMDP: The states, state transitions and observations represent the student model; The policy works as the tutoring model. The POMDP actions model ITS tutoring.

At a point of time, the agent is in state $s$, which represents the student’s current knowledge state. To respond to a question from the student, it calculates its belief $b$, and applies policy $\pi(b)$ to choose the optimal teaching action $a$ and takes it. After $a$, the student may ask another question, which is observed by the agent as $o$. Then the agent calculates new belief $b'$, and chooses the next tutoring action $a' = \pi(b')$, and so on.

5 Student Model on States

5.1 State definition

The student model has two components: the knowledge states of individual students and the behavior of general students. We create a POMDP state space for each student, a state representing a knowledge state.

We define POMDP states in terms of the concepts in the instructional subject. A subject may include a number of concepts, for example, program, database in software basics. Learning a subject requires understanding its concepts. To provide adaptive tutoring, a teacher must have the information about what concepts the student understands.

For each concept $C$, we define two conditions:

- The understand condition, denoted by $\sqrt{C}$, indicating that the student understands $C$, and
- The not understand condition, denoted by $\neg C$, indicating the student does not understand $C$.

We use conjunctive formulas made of $\sqrt{C}$ and $\neg C$ to represent knowledge states. Each POMDP state is associated with a formula, which is referred to as a state formula. For example, $(\sqrt{C_1} \land \sqrt{C_2} \land \neg C_3 ...)$ is a state formula. When the student has $N$ concepts, a state formula is

$$(C_1C_2C_3...C_N)$$

of length $N$, where $C_i$ is the condition variable for the $i$th concept, taking a value $\sqrt{C_i}$ or $\neg C_i$ ($1 \leq i \leq N$). States thus defined have the Markov property.

5.2 Dealing with the exponential state space

When there are $N$ concepts in a subject, the number of state formulas is $2^N$. To develop a feasible POMDP
for practical tutoring, we must deal with the problems caused by the size of state space. It has been recognized that the exponential state space is a challenging problem in using POMDPs for building ITSs [6] [5] [4].

We developed a technique to address the space problem. In the technique, the information of pedagogical orders (e.g. prerequisite relationships) plays an important role in determining the actual size of a state space, and reducing computational costs.

Let \( C_1 \) and \( C_2 \) be two concepts. To understand \( C_2 \), if a student must first understand \( C_1 \), we say that \( C_1 \) is a prerequisite of \( C_2 \). A concept may have zero or more prerequisites, and a concept may serve as a prerequisite for zero or more other concepts. Concepts and their prerequisite relationships can be represented in a directed acyclic graph (DAG). Figure 1 shows the DAG for a sub-set of 14 concepts in software basics.

![Figure 1: A directed acyclic graph (DAG) of 14 concepts and their prerequisite relationships. An arrow represents “is a prerequisite of”.](image)

Although there are \( 2^N \) possible formulas, many of them represent non-existing states. For example, when \( C_1 \) is “binary digit”, \( C_2 \) is “bit” and \( C_3 \) is “byte”, formula \( (\sqrt{C_1 C_2 C_3}...) \) represents a non-existing state, in which a student understands “byte” without knowledge of its prerequisite “bit”. In a state, if a concept is in “\( \sqrt{ } \)” condition but one or more of its prerequisites are in “\( \neg \)” condition, we call the state an invalid state and the formula associated with it an invalid formula.

Take the concepts in Figure 1 as an example. The concept conditions can form \( 2^{14} = 16,384 \) formulas. However, according to the prerequisite relationships and the definition of invalid states, there are only 68 valid states. The number is less than 0.5% of 2\(^{14}\).

When encoding state formulas in the form of (6), we perform topological sorting on the DAG to generate a 1-D sequence of concepts. In a formula of sorted concepts, prerequisites of any concept are to the left of the concepts. The sorting facilitates manipulating the formulas and identifying invalid formulas/states. In addition to eliminating invalid states, we developed a new technique to partition the state space into sub-spaces. The partitioning allows the agent to localize computing when it chooses an action to answer a question. For details of space partitioning and invalid state elimination, please see [9].

6 Actions, Observations, and State Transitions

6.1 Actions and observations

In a tutoring session, asking and answering questions are the primary actions of the student and system. Other actions are those for greeting, confirmation, etc.

Student actions are mainly asking questions about concepts. We assume that a student question concerns only one concept. We denote a student action of asking about concept \( C \) by \( (?C) \), and use \( (\Theta) \) to denote an acceptance action that accepts a system answer, like “I see”, “please continue” etc.

System actions mainly include answering questions about concepts. We use \( (!C) \) to denote a system action of teaching \( C \), and use \( (\Phi) \) to denote a system action that does not teach a concept, for example a greeting.

In the experimental ITS, a tutoring session is a sequence of \((a, o)\) pairs, where \( a \) is a system action, and \( o \) is a student action the agent observes after taking \( a \).

6.2 State transition

In \( s \in S \), the agent is allowed to take any one of the system actions available in the state. An action may transit \( s \) into \( s' \). Take \( s = (\neg C_1, \neg C_2, ..., \neg C_N) \) as an example. It represents the knowledge state that the student understands nothing. We also use it as the initial state \( s_0 \) in which the agent has no information about a new student. Assume the system action in this state is a greeting denoted by \( (\Phi) \). This action causes a state transition into \( s' \). \( s' \) can be any valid state. The probability of transition into \( s' \) is \( P(s'|s_0, \Phi) \).

Assume the student asks question \((?C_l)\) after the state transition into \( s' \). The probability for the student to ask question \((?C_l)\) in \( s' \) after system action \( (\Phi) \) is \( P((?C_l)|\Phi, s') \). Also assume \( C_1, C_{l-1} \) and some concepts between them are prerequisites of \( C_l \). The following are the \( s' \) states such that \( P((?C_l)|\Phi, s') \neq 0 \):

- \((\neg C_1, \neg C_{l-1}, \neg C_l, ..., \neg C_N)\)
- \((\sqrt{C_1, \neg C_{l-1}, \neg C_l, ..., \neg C_N})\)
- ...
- \((\sqrt{C_1, \sqrt{C_{l-1}}, \neg C_l, ..., \neg C_N})\)
The above states represent different student knowledge states. For example, the last one represents that the student does not understand $C_i$ but understands all its prerequisites. The states have different observation probabilities of $P(\mathcal{C}_1 | \Phi, s')$. The differences in observation probabilities help us calculate the agent’s belief about the current states.

6.3 Calculation of a new belief

In a state, for a given student question, there is one or more optimal teaching actions. Continue the example in the previous subsection. To answer question $(?C_i)$, if the agent knows that the new state $s' = (\sqrt{C_1} - C_2 - \ldots - C_i - \ldots - C_N)$, $(!C_2)$ is one of the optimal actions when $C_2$ is a prerequisite of $C_i$. However, when the agent does not know exactly, we need to calculate a belief as the basis for the agent to choose an action.

As showed in Eqn (1), a belief is a vector. The task of calculating new belief $b'$ is to calculate all the $N$ elements of $b'$. Eqn (2) is the formula to calculate individual elements in $b'$ at $t + 1$. Once $b'$ has been calculated, it becomes the basis for the agent to apply a tutoring model to choose the next optimal action.

6.4 Modeling behavior of general students

The student model has two components: One models knowledge states of individual students, and the other models the behavior of general students. The former is represented by the POMDP states, and the latter by the transition probabilities and observation probabilities.

Transition probability $P(s'|s, a)$ provides information about how $a$ may probably change a student’s knowledge state from $s$ to $s'$. Observation probability $P(o|a, s')$ provides information about how a student probably asks questions in $s'$ after the agent takes $a$.

The two sets of probabilities form the foundation for a POMDP to operate. They are initialized before the system teaches for the first time, and then they are updated when the system interacts with its students, in order that they can model the students better. For details about initialization and update of the two sets of probabilities, please see [11].

7 Tutoring Model as Trees

7.1 Policy tree, value function, and reward function

The agent uses the tutoring model to select the optimal actions. We express the tutoring model as policy trees. To answer a question, the agent evaluates a set of pre-constructed policy trees and selects the tree that maximizes the value function, given in Eqn (5).

To apply Eqn (5) to select the optimal policy tree, we need function $V^\tau(b)$ for each tree $\tau$. It is defined as:

$$V^\tau(b) = b \cdot V^\tau.$$  \hspace{1cm} (7)

where $V^\tau$ is a vector:

$$V^\tau = [V^\tau(s_1), V^\tau(s_2), ..., V^\tau(s_N)]. \hspace{1cm} (8)$$

For a given $s$, $V^\tau(s)$ is calculated by using Eqn (3).

We define the reward function as $\rho : S \times A \rightarrow \mathbb{R}$. Thus the function is of the form $\rho(s, a)$. Let $C_i$ be a concept $(1 \leq i \leq N)$, $\varphi_{C_i}$ be the set of all the direct and indirect prerequisites of $C_i$, and $s$ be the current state represented by $(C_1C_2...C_N)$. We define the reward function as

$$\rho(s, (\mathcal{C}_i)) = \begin{cases} r & \text{if } \forall \mathcal{C}_i \in \varphi_{C_i} = \sqrt{C_i} \land (C_i = \neg C_i) \\ r' & \text{otherwise} \end{cases} \hspace{1cm} (9)$$

where $r$ and $r'$ are scalar values, and $r > r'$. In our experiment, we choose $r = 2$ and $r' = 1$. Eqn (9) can be verbally stated as: When the agent takes action $(!C_i)$ in state $s$, if in the state formula of $s$ all the direct and indirect prerequisites of $C_i$ are in the “$\sqrt{ }$” condition, and $C_i$ is in “$\neg$” condition, the agent receives high reward $r$, otherwise receives low reward $r'$.

7.2 Policy tree construction

For discussing the algorithm for constructing policy trees, we classify student questions into the “original questions” and “current questions”. An original question starts a tutoring session, and a current question is to be answered in the current step.

In the following, we assume $(?C_i)$ and $(?C_j)$ are the original and current questions respectively, and $C_j \in (\varphi_{C_i} \cup C_i)$. For every pair of $(?C_i)$ and $(?C_j)$, we construct a set of policy trees. To answer $(?C_i)$ the agent evaluates this set of policy trees, selects the optimal, and takes the root action of it. The agent uses a policy tree to estimate the expected return that results from taking the root action in the current step.

We construct a policy tree by integrating atomic trees. For each concept in the subject, we create an atomic tree. In the atomic tree of $C$, the root is $(!C)$, and there is one or more edges for connecting the root with its children. The edges are labelled with the predictable student actions after the root action is taken: After $(!C)$, the predictable student action can be an acceptance action, or a question about one of the prerequisites of $C$. 


In a policy tree for $C_i$, the root action is $(IC_i)$, and every leaf is an action for terminating the session. The terminating action is connected by an edge of $(\Theta)$ to an action of $(IC_i)$. This represents that the student accepts the answer to the original question. Every path from the root to a leaf is a process of tutoring. It starts with answering the current question, and ends when the student accepts the answer to the original question. Figure 2 illustrates a policy tree, in which the original and current questions are both (?ML). In this tree, a thick horizontal line denotes a terminating action.

Figure 2: A policy tree for ML. (ML – machine language, IN – instruction, BD – binary digit, PL – programming language.)

All the sets of policy trees are pre-constructed and stored in a tree database. When the agent needs to answer a question, it retrieves the database, and gets a set of trees to evaluate to find the optimal. A major strength of our policy tree technique is high efficiency. When looking for an action to answer a question, the agent needs to evaluate a finite set of pre-constructed policy trees, which is determined by the original and current questions. For further efficiency improvement, we store each calculated $V^r(s)$ value in a database for possible re-use until it must be updated. For details about our policy tree techniques, please see [10].

8 Experiment and results

30 students participated in the experiment. The students knew how to use desktop or laptop computers, or smart phones, Windows, Mac, or phone operating systems, and application programs for browsing the Web, word processing, email, and so on. They did not have formal training in computer science and software development.

The students were randomly divided into two groups of the same size. The students in Group 1 studied with the ITS with the POMDP turned off. When a student asked about a concept, the system either taught the concept directly, or randomly chose a prerequisite to start. The students in Group 2 studied with the ITS in which the POMDP was used to choose optimal actions.

The ITS taught a student at a time. Each student studied with the ITS for about 45 minutes. For each student, the question-answer sessions were recorded for performance analysis.

The performance perimeter is rejection rate. Roughly, if right after the system teaches concept $C$, the student asks a question about a prerequisite of $C$, or says something like “I already know $C$”, we consider the student rejects the system action. For a student session, the rejection rate is defined as the ratio of the number of system actions rejected by the student to the total number of system actions. We used the rate to measure the optimality of choosing actions.

8.1 Result Analysis

We applied a two-sample $t$-test method to evaluate the effects of the POMDP-based decision making to the teaching performance of the ITS. The test method is the independent-samples $t$-test.

For each student, we calculated the mean rejection rate. For the two groups, we calculated means $\bar{X}_1$ and $\bar{X}_2$. Sample mean $X_1$ is used to represent population mean $\mu_1$, and $X_2$ is used to represent $\mu_2$.

The alternative and null hypotheses are:

$$H_a: \mu_1 - \mu_2 \neq 0, \quad H_0: \mu_1 - \mu_2 = 0$$

Table 1: Number of students, mean, and estimated variance of each group.

<table>
<thead>
<tr>
<th></th>
<th>Group 1</th>
<th>Group 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of students</td>
<td>$n_1 = 15$</td>
<td>$n_2 = 15$</td>
</tr>
<tr>
<td>Sample mean</td>
<td>$\bar{X}_1 = 0.5966$</td>
<td>$\bar{X}_2 = 0.2284$</td>
</tr>
<tr>
<td>Estimated variance</td>
<td>$s^2_1 = 0.0158$</td>
<td>$s^2_2 = 0.0113$</td>
</tr>
</tbody>
</table>

The means and variances calculated for the two groups are listed in Table 1. In the experiment, $n_1 = 15$ and $n_2 = 15$, thus the degree of freedom is $(15 - 1) + (15 - 1) = 28$. With alpha at 0.05, the two-tailed $t_{crit}$ is 2.0484 and we calculated $t_{obt} = +8.6690$. Since the $t_{obt}$ is far beyond the non-reject region defined by $t_{crit} = 2.0484$, we could reject $H_0$ and accept $H_a$.

As listed in Table 1, the mean rejection rate in Group 1 was 0.5966 and the mean rejection rate in Group 2 was 0.2284, and the accepted alternative hypothesis indicated the difference between the two means was
significant. The analysis suggested that by using the POMDP-based decision making, the rejection rate has been reduced from 0.5966 to 0.2284.

When making a decision, the agent evaluated a small number of trees. The average was less than 10 in the experiment. When the experimental ITS run on a desktop computer with an Intel Core i5 3.2 GHz 64 bit processor and 16GB RAM, the response time for answering a question was less than 300 milliseconds. This included the time for calculating a new belief, choosing a policy tree, and accessing the database of domain model. For a tutoring system, such response time could be considered acceptable.

9 Concluding Remarks

In teaching a student, a competent teacher should be able to choose the most beneficial teaching actions based on his/her information about the student’s knowledge states. An ITS should have the same ability. We aim at developing techniques for building an ITS that can teach adaptively.

We developed novel techniques for POMDP-based decision making, dealing with exponential state space, and efficient POMDP solving. The techniques can be applied to subjects that consist of knowledge components (KCs) with prerequisite relationships. The techniques enable us to develop our experimental system that could teach adaptively. In some sense, its teaching has been well accepted by students, in terms of choosing actions and response time.

Acknowledgements

This research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References


