Science Research
2026, Vol. 14, No. 1, pp. 1–13
https://doi.org/10.11648/j.sr.20261401.11

Research Article

Learning Path Generation of ITS Using Markov Decision Process

Song-Hwan Kwon[1, *], Jong-Nam Rim[1], Chung-Song Ko[1], Un-Song Ryu[1], Yong-Jin Pak[2], Hyon-Il Son[3]

[1]Department of Information Science, University of Science, Pyongyang, Democratic People's Republic of Korea
[2]Institute of Information Technology, University of Science, Pyongyang, Democratic People's Republic of Korea
[3]Information Department, Sinuiju College of Industrial Technology, Sinuiju, Democratic People's Republic of Korea

Abstract

Probabilistic and stochastic models such as Bayesian and Hidden Markov models can cope well with system uncertainties, but there is a problem of how learning state prediction and learning path generation are performed independently and how to connect them, and the overall effect of the system may be lost even after the connection. Using a Markov Decision Process, a kind of reinforcement learning model, not only can the prediction of the learning state of a student and the generation of a path be implemented simultaneously in a single model, but also the overall error can be reduced. In this paper, we propose to build an intelligent tutoring system into a Markov Decision Process model, an reinforcement learning model, with the aim of reducing learning path generation error and improving system performance by using Markov decision Process model in intelligent tutoring system. In addition, we propose a learning state evaluation method using a Markov Decision Process model to simultaneously proceed the student's learning state estimation and the system's action selection. We also propose a method to apply the value-iteration algorithm to action selection computation in a Markov Decision Process model. Comparison with previous models was carried out and its effectiveness was verified.

Keywords

Reinforcement Learning, MDP, Learning Path Generation

1. Introduction

Markov Decision Process and Partially Observable Markov Decision Process are known as good methods to improve the model online by using reinforcement learning. In the case of using Markov Decision Process for learning path generation in intelligent tutoring system, we assume that the learning state of the student is fully observable. On the other hand, in partially observable Markov Decision Process, the student's learning state is assumed to be observable only partially, and in particular indirectly only through sensory information. We analyze previous studies on the development of intelligent tutoring systems using Markov Decision Process, partially observable Markov Decision Process and reinforcement learning.

*Correspondence: Song-Hwan Kwon (ksh1974@star-co.net.kp)

Received: 27 October 2025; **Accepted:** 17 November 2025; **Published:** 30 January 2026

2. A literature Review on the Development of ITSs Using Reinforcement Learning, MDP and POMDP

2.1. A Previous Study on the Application of Reinforcement Learning Methods to Intelligent Tutoring Systems

Research on the application of reinforcement learning to intelligent tutoring systems began in the 1990s. In Iglesias et al. [3], the authors reported that the intelligent tutoring system used reinforcement learning algorithms to provide adaptive teaching to individual students based on their teaching experience. The author of Litman [4] describe the application of reinforcement learning to an dialogue speech tutoring system. In Sarma et al. [5], the authors propose to use reinforcement learning to build an intelligent tutoring system for students with AS.

In Litman et al. [4], the authors reported the development of a speech dialogue system called ITSPOKE using reinforcement learning. This system uses an intelligent tutoring system for tutoring, called Why2-Atlas. The system presents a qualitative question about physics and the student responds to it in natural language text. The system also answers the student's questions, corrects the student's wrong ideas, and, if necessary, makes the explanation at a deeper level.

In Iglesias Aleven et al. [6], reinforcement learning algorithm was used to build an educational model so that the system can find out what the best educational strategy is suitable for students. One important feature of this approach is that the system can improve its strategy based on feedback data already obtained from other students in similar situations and backgrounds.

Studies have also been conducted to use reinforcement learning in students' emotional surveys in e-learning systems. In Ai et al. [7], the authors reported that the system and student learning behavior features were used in the learning process. This paper emphasizes that the detection of students' emotions plays an important role in the construction of intelligent tutoring systems. Information about student emotions can be used to determine the student's passage time of the subject.

In Janarthanam et al. [8], instead of manual decision rule development, students' feedback information was collected based on various time-interval representations such as task success, ambiguity and user preference, and then based on this, a data-driven decision rule generation method was proposed. This data collected was used to train reinforcement learning strategies to learn adaptive time-domain teaching directions in a variety of environments.

In Pan et al. [9], a intelligent tutoring system that interacts with students was proposed and designed. The student may ask more questions if he is not satisfied with the explanation of the keywords given by the system. The system was built with MDP and all keywords were aligned according to the success rate obtained by performing reinforcement learning.

In addition to the system development work, the system evaluation and analysis were also studied simultaneously. In Chi et al. [10], an empirical evaluation of the results of applying reinforcement learning to adaptive instructional orientation is presented. In Iglesias et al. [11], the learning performance in tutoring systems was evaluated through three issues: convergence of machine learning results, heuristics and reduction of machine learning steps.

In recent studies on the application of reinforcement learning to intelligent tutoring systems, machine learning has been conducted in the MDP framework. Such intelligent tutoring systems could improve the system's teaching ability through interaction between the system and the student, but uncertainty could not be handled. On the other hand, recent studies on developing intelligent tutoring systems based on POMDP have made progress in the uncertainty handling problem, but not immediately following the improvement of teaching ability when the intelligent tutoring system is operating online. In those systems, the tutoring directions were unchanged compared to using MDP, and there was no updating during the teaching by the system.

2.2. A Previous Study to Build Intelligent Tutoring System Using MDP and POMDP

Building an intelligent tutoring system with MDP and POMDP can predict user action and even make a plan even if the environment is uncertain. The system performs the possible actions based on the assumption of the student's current state of learning and then moves to the following physical state. This technique is characterized by maintaining a distribution of as many hypotheses as possible about the current state of conversation.

The research on the application of MDP and POMDP to adaptive or intelligent tutoring began in the 1990s [12-18]. These studies used POMDP to address the uncertainty in the tutoring. In Rafferty et al. [14], the authors reported that have built a system for tutoring concepts and developed tutoring techniques by POMDP planning. The core element in this technique was the set of approximate POMDP policies for uncertainty handling. The work introduced by Folsom-Kovarik [16] aims to solve real-world problems by POMDP solvers. In their work, POMDP was used to address the uncertainty in system planning and the internal processes that takes place in the learner's mind. A data structure for the student's state description is proposed, which consists of a knowledge state and a cognitive state. This state of knowledge, also defined as gaps, was able to detect and eliminate this gap even in the presence of uncertainty in the intelligent tutoring system using POMDP.

In Jeremiah et al. [19], we used a representation called state queue and observation chain. We have been able to represent

more than 100 independent learner features using the attributes of the task when building an intelligent tutoring system. This study was conducted on a real-world military training project. The intelligent tutoring system, which consists of POMDP with strong policy improvement ability, has made it easy to adapt to the learning status and input data of many students.

The algorithm developed in Williams et al. [12] was a typical algorithm that allowed POMDP to be used in an speech dialogue intelligent tutoring system. At each time step, the student state is uncertain in the physical state, but the state information about that state is a belief state *b*, which is defined as the ratio of the probability that a particular physical state exists. In specific time steps, both the student's learning process and the system's actions were in an uncertain state. The belief state is denoted by $(b(s_1), b(s_2), ...)^T$ vector equal to a, and the vector space is divided into different regions.

In Wang [20], the authors stated that Markov decision Process is not applicable if the student's learning status is uncertain. In addition, this paper describes that no reinforcement learning method has been proposed to improve on-line strategies in POMDP.

In Hamid et al. [21], a conversational management system called SmartWheeler (tutoring support system) was proposed based on COMDP (Complete Observable Markov Decision Process) and POMDP, which was designed to work with previous conversational history data. This system has built a POMDP with two different models using its historical data. One model was based on keywords, and the other was based on student questions.

In Burhan [2], the authors introduced the use of MDP to build an intelligent tutoring system for math learning.

3. A Learning Path Generation Method Using Markov Decision Process Model

The first step to build a learning path generator into a Markov Decision Process model is to formalize the task environment to be solved into a Markov Decision Process model.

3.1. Formalization of Markov Decision Process Model

3.1.1. General Concept

1) Maximum expected utility value principle.

The maximum expected utility principle is described in Russell et al. [1] as follows. An agent that must make a decision has actions a and states s. Since there may be uncertainty in the current state, we assign probability $P(s)$ to each possible state s. There may also exist uncertainties in the results of the action, given that the transition model is $P(s'|s, a)$, the meaning is the probability of reaching state s' when taking action a in state s. P (RESULT $(a) = s'$) has the meaning that when performing action a, it is the probability value that it reaches s'.

$$P(RESULT(a) = s') = \sum_{s} P(s)P(s' \mid s, a)$$

An agent uses utility function $U(s)$ to express the degree of preference for a state. The utility function is the agent assigned a value to the expected value in a state. When evidence is presented, the expected utility of an action is defined as the average value of the utility of the possible outcome state obtained by taking that action, which is the average value by weighting the probability that outcome state is obtained.

$$EU(a) == \sum_{s'} P(RESULT(a) = s')U(s') \quad (1)$$

The principle that a rational agent will choose an action that maximizes the expected utility of an agent is called the MEU (maximum expected utility) principle.

2) Markov Decision Process

MDP (Markov Decision Process) refers to a fully observable, sequential decision problem in a statistical environment with Markov transition model and reward values, which consists of a set of states containing initial state s0, a set of actions in each state ACTIONS (s), a transition model $P(s' \mid s, a)$, and a reward function $R(s, a, s')$ [1].

In an intelligent tutoring system, Markov Decision Process is defined as a pair of <S, A, T, R>, where S represents the observable state space defined by a set of features representing an interactive learning environment, A represents the space of possible actions that an intelligent tutoring system can execute, T represents the transition probability, and R represents the reward predictor when it transitions from one state to another by taking some action. In this paper, the optimal strategy π^* of MDP is generated by a value iteration algorithm.

3) Partially observable Markov Decision Process

POMDP (Partially observable Markov Decision Process) is an extension of MDP, where A and R are the same as in MDP and are defined as pairs <S, A, R, Ph, Po, B, prior>. S denotes the hidden state space, Ph denotes the transition probability inside the hidden states when taking action, and Po denotes the conditional observation probability. Prior represents a prior probability distribution for hidden states and B represents a belief state space, which is constructed by the IOHMM (Input-Output Hidden Markov Model) proposed by Shen [22].

The policy-driven process of POMDP consists of three steps as follows. First, we transform the training corpus into a hidden state space via Viterbi algorithm. Second, we implement Q-learning to estimate the Q-values for each hidden state and action pair (s, a). Third, we evaluate the value of a confidence state b and a action at time t as follows:

$$Q_t(b, a) = \sum_{s} B_t(s) \bullet Q(s, a) \quad (2)$$

Thus $Q_t(b,a)$ is a linear combination of $Q(s,a)$ for each hidden state with the corresponding confidence level $B_t(s)$. When this process converges, taking the optimal action a at time t associated with the largest $Q_t(b,a)$, we obtain π^*.

3.1.2. Building a Markov Decision Process Model

1) Definition of the pair <S, A, T, R>.

Markov Decision Process is a decision process when there is uncertainty of action transitions in an fully observable environment. Generally, Bayesian networks for state estimation are extended to action and utility value nodes to transform into decision networks. MDP can be represented by an extended dynamic Bayesian network. Markov Decision Processes can be represented by an extended DBN (Dynamic Bayesian Network), consisting of decision, reward, and utility nodes, to create a DDN (Dynamic Decision Network).

We draw the first-order Markov assumption and assume that the student's learning state is sufficiently observable. That is, the learning behaviors extracted by the processes and the learning activities that depend on the learning process are then modeled using Markov assumption assuming that the learning states depend only on the current learning state.

a) Learning state variables

- Representation of the learning state.

Previously (formalized by the hidden Markov model), we have defined the learning state as follows. The LSt consists of a set of slides (S, Slides), comments (Co, Comments), videos (V, Videos), questions and answers (QA, Questions and Answers).

LSt = <St, Cot, Vt, QAt>, where t is the specific time.

LSt=<Slidet(Slidet1, Slidet2,...,Slidetns), Cot(Cot1, Cot2,..., Cotne), Vt(Vt1, Vt2,..., Vtnv), QAt(QAt1: QAt1, QAt2: QAt2,..., QAt1: QAtnq)>

Here ns, ne, nv, nq means that multiple slide displays, video screens, comment additions, and QAs are possible at a specific time stage. Each time it corresponds to the topic to be explained, and one topic is available for several slides, video projections, comment displays, and query presentation.

- Learning state variables

The learning state variable should reflect the learning state, the cognitive state, at a point in the student's knowledge. Previously, when learning state modeling is done in a probabilistic way, the learning state is represented as a random variable, including uncertainty, which is defined as a learning state variable. We defined this variable as $V_{subi}^{hyperl,\ basicm}$(or V_{basick}). In the generation of learning paths by Markov Decision Processes, this learning state variable is used to represent different representations from those in hidden Markov models.

- Graph-based learning path representation and its corresponding Markov chain

The learning path is a sequence of knowledge nodes, whose representation is done by the knowledge graph [23, 24]. Representing the learning path LP as a Markov chain, it consists of a sequence of Learning State LS in discrete time steps. The learning state corresponds to the nodes of the basic concept graph.

LP = {LS_1, LS_2,...,LS_m},

where m is the number of basic concepts as the number of times.

- Learning state estimation is based on the assumption of the system constructor

When an intelligent tutoring system is constructed using a hidden Markov model or a Markov Decision Process model, the first problem is whether to define this state as fully observable or partially observable, with the definition of a state. The definition of a learning state as a fully observable state or partially observable state is determined by the system developer.

It is not possible to look directly into the student's head, and it is not yet known how the student will benefit from further research by utilizing the knowledge, so the value of a student's learning status value at a knowledge node is currently not accurately measured. It is also a matter of future discussion, even if the test score is high, what value will be taken for the final test in the future, and how the student's results will be assessed when the other teacher has presented the application in a different way. Therefore, whether it is fully observable or only partially observable, determines the system developer.

- Assume a fully observable state

We assume the learning state as a fully observable state. That is, the student's learning state is assumed to be observed completely by the evaluation of the ability at that node, i.e., the cognitive comprehension test.

b) Defining Learning State Variables

The Markov model is constructed between the Learning State Variables LSV_t (corresponding to Cbasict in the study of the previous hidden Markov model) and LSV_{t+1}(corresponding to $C_{basic\ t+1}$ in the study of the previous hidden Markov model). There are many knowledge nodes (state variables) between two state variables to maximize the recognition rate for each student. The value of a learning state variable is defined as the H (High), M (Medium) and L (Low) level of student's cognition. "High"(H) is when the value of this state variable lies in the interval between [0.8, 1], "medium"(M) is when the value lies in the interval of [0.4, 0.8], and "Low"(L) is in the interval of [0.0, 0.4]. That is value of learning state variable LSVt = {H, M,L}.

The LSV_t is defined by dividing the state variables into the following. That is, we define state variables as the notion of subdividing the learning state variables.

LSV_t = {Basic_Concept State Variable, Sub_Concept State Variable, Help_Concept State Variable, Supplement_Explanation State Variable}

- Basic_Concept State Variable Ba

The main purpose of the lecture is to enable the student to study and recognize the basic concepts themselves, or to solve the application or research tasks corresponding to the basic concepts. The basic concepts include concepts, applications, and practical problems.

- Sub_Concept State Variable Sub

In cases where there is a lack of understanding of the basic concept or a weak problem-solving ability, it is used to further investigate the subconcepts that support the basic concept. The subconcept state variable includes the contents of the basic_concept state variable and the contents of the sub_concept.

- Auxiliary_Concept State Variable Aux

A Auxiliary_Concept State Variable is a variable that supports the subconcept. Then, the Auxiliary_Concept State Variable is added to the contents of Auxiliary_Concept State Variable, together with the contents of Sub_Concept State Variable.

- Supplement_Explanation State Variable Supp

The concept below is the same as the axiom, which is the explanation that the intelligent tutoring system explains to the bottom level. The state variable is then added to the contents of supplementary state variable, with the contents of the auxiliary concept state variable.

c) Defining action

The action in an intelligent tutoring system is contingent, enabling the transition from a knowledge node (the learning state variable) to the next knowledge node (the learning state variable) connected in the next time order. Thus, the transition action from the current state to the next one has a probabilistic property. We will see this by a detailed example in the following.

The actions include Up_1, Up_2, Up_3, $Down_1$, $Down_2$, $Down_3$, Right, and Jump. The meaning of these actions will be discussed in detail when illustrated in Figure 1.

d) Definition of the transition model

The transition model describes the results of each action in each state. The result of the performance of the action has a statistical property, which is denoted by P (s' | s, a), indicating the probability value of reaching s' when executing action a in state s. Then, we assume that these transitions have Markovian properties, which means that the probability of reaching s from s' depends only on state s and not on the previous state histories.

Table 1 shows an example of the transition model, where a:y = 1.0. a:y means that enumerate from each of the values from the leftmost cell value a to the rightmost cell value y of any one row of the transition model table lists.

Table 1. *Transition Model.*

S_{t-1}	$P(S_t \mid S_{t-1})$		
	H	**M**	**L**
H	0.85	0.1	0.05
M	0.3	0.4	0.3
L	0.1	0.4	0.5

e) Definition of the reward function

To define a task environment completely, we must also consider utility functions. In a sequential decision problem, the utility function depends on a sequence of states and actions, i.e., the environment history, rather than on any single state. For all transitions from state s to s' using action a, the intelligent tutoring system will receive reward R (s, a, s'). This reward value may be positive or negative, but has a limit of ±Rmax.

We define the value of the reward function as follows. In the case of the basic concept state variable, we assign +3 when we have reached high state (state H), +1.5 when we have reached intermediate state (state M), and -3 when we have reached low state (state L). In the case of other state variables, assign a value of -0.1 to the H state, -0.2 to the M state, and -0.3 to the low state.

2) Value identification Algorithm of State Variables

In intelligent tutoring systems, the state variables reflect the learning state of the student.

a) Value identification algorithm of the basic concept state variables

[Algorithm 1. Value identification algorithm of basic concept state variables]

Step 1: Presently, query presentation that elicit the basic concept of time t to the student himself, if the answer score for the question is 10, then move to step 3, step 2, if no.

Step 2: Carry out a supplementary lecture that helps the student to derive the basic concepts by himself, and move to step 1.

Step 3: A thorough lecture on the basic concept of time t (meaning that the intelligent tutoring system is doing a full-scale re-lecture on the concept, considering that the student himself has derived the basic concept but is not yet fully understood).

Step 4: A question asked to understand the basic concept of time t. The result is one of {H, M, L}.

b) Value identification algorithm of the sub_concept state variables sub

[Algorithm 2. Value identification algorithm of sub_concept state variables]

Step 1: Presently, query presentation that elicit the sub concept of time t to the student himself, if the answer score for the question is 10, then move to step 3, step 2, if no.

Step 2: Carry out a supplementary lecture that helps the student to derive the sub concepts by himself, and move to step 1.

Step 3: A thorough lecture on the sub concept of time t (meaning that the intelligent tutoring system is doing a full-scale re-lecture on the concept, considering that the student himself has derived the sub concept but is not yet fully understood).

Step 4: A question asked to understand the sub concept of time t. The result is one of {H, M, L}.

c) Value identification algorithm of the Auxiliary_Concept State Variable Aux.

[Algorithm 3. Value identification algorithm of Auxiliary_Concept State Variable]

Step 1: Presently, query presentation that elicit the auxiliary concept of time t to the student himself, if the answer score for the question is 10, then move to step 3, step 2, if no.

Step 2: Carry out a supplementary lecture that helps the student to derive the auxiliary concepts by himself, and move to step 1.

Step 3: A thorough lecture on the auxiliary concept of time t (meaning that the intelligent tutoring system is doing a full-scale re-lecture on the concept, considering that the student himself has derived the auxiliary concept but is not yet fully understood).

Step 4: A question asked to understand the auxiliary concept of time t. The result is one of {H, M, L}.

d) Value identification algorithm of the Auxiliary_Concept State Variable Aux.

[Algorithm 4. Value identification algorithm of Supplement_Explanation State Variable Supp]

Step 1: supplementary lecture

Step 2: A question asked to understand the supplement_explanation of time t. The result is one of {H, M, L}.

3) Dynamic decision networking for Markov decision process

Figure 1 shows the dynamic decision network of the intelligent tutoring system in three layers, which may be composed of four or five layers as needed. In the figure, the green rectangle indicates state variables, the green rectangle indicates high state H, intermediate state M, and low state L in the state variable, the gray rectangle represents non-transitive state variables, violet circles represents actions, and the blue trapezoidal represents the reward value and utility function value.

a) Expression of state variables

The intelligent tutoring system consists of the state variables of the basic concept state variables, the sub_concept state variables, the Auxiliary_Concept State Variable, and the Supplement_Explanation State Variable, and has the state values {high (H), medium (M), and low (L)}. In the figure, Bt indicates the basic concept state variable (abbreviated Ba), Sbt indicates the subconcept state variable (abbreviated Sub), At indicates the subconcept state variable (abbreviated Aux), and St indicates the supplementary description state variable (abbreviated Supp). Due to the presence of state variables with different names, the sequence of state variables that constitutes the learning path consists of state variables with different names. We use the symbol S for convenience, ignoring the other variable names when representing a learning path consisting of state variables with different names. If we consider the logical upper and lower levels between state variables, then the basic concept state variable >> subconcept state variable >> auxiliary concept state variable >> supplement explanation state variable. For example, a basic concept state variable is a top-level concept compared to a subconcept state variable. The dotted lines indicate that the two states are not actually transitive. In the figure, the concept state variables are uppercase, and in our development we denote each state variable by a lowercase letter unless we specifically refer to the set of states.

b) Expression and meaning of actions

It consists of actions named Up1, Up2, Up3, Down1, Down2, Down3, and Right.

- Up1, Up2, Up3

These actions mean a transition to a higher-level concept state than itself, which is classified as a transition to a one-level higher state and a transition to a two-step higher state by subscript.

- Down1, Down2, Down3

These actions mean a transition to a lower-level concept state than itself, which is classified as a transition to a one-step lower state, and a transition to a two-step lower state, depending on the subscript.

- Right

This action has the meaning of moving to the same level of concept state in the next time step.

- meaning of subscript and superscript of action notation.

Up3,t1 is an Up3 action, which is used in the state variables at time t, 1, meaning that it moves to a three-level lower concept state than the current concept state.

c) The transition probability of intelligent tutoring system

The intelligent tutoring system performs its actions (i.e., by "doing lectures", "doing learning") with Bt as the initial state in the network of Figure 1, and completes its actions in the Bt+1 state. Each performance is considered probabilistic, determined through experiments, empirically or through reinforcement learning. Up1, Up2, Up3, Down1, Down2, Down3, and Right actions are able to move with a probability of 0.91 to the state variable to which they are moving, and then move to other state variables at the next time is possible with a probability of 0.03, respectively. The transition to the next subconcept state variable on the right in each state variable, e.g. Pt-1, is performed with a probability of 0.91, and the other transition to the different level state variables at the next time is done with probabilities of 0.03, 0.03, and 0.03, respectively. For example, by executing Down1, t-13 on the St state variable, the probability of reaching a state Pt,1 is 0.91 (probability of reaching a high state H is 0.31, probability of reaching an intermediate state is 0.3, probability of reaching a low state is 0.3), and the probability of reaching other states is 0.03, respectively. The transition probabilities of Down1, t-13 are {0.91, 0.03, 0.03}.

Table 2. *Transition Model.*

S_{t-1}	$P(S^1_t \mid S_{t-1}) = 0.91$			$P(S^2_t \mid S_{t-1})=0.03$			$P(S^3_t \mid S_{t-1})=0.03$			$P(S^4_t \mid S_{t-1})=0.03$		
	H	**M**	**L**	**H**	**M**	**L**	**H**	**M**	**L**	**H**	**M**	**L**
H	0.91×0.85	0.91×0.1	0.91×0.05	0.03×0.85	0.03×0.1	0.03×0.05	0.03×0.85	0.03×0.1	0.03×0.05	0.03×0.85	0.03×0.1	0.03×0.05
M	0.91×0.3	0.91×0.4	0.91×0.3	0.03×0.3	0.03×0.4	0.03×0.3	0.03×0.3	0.03×0.4	0.03×0.3	0.03×0.3	0.03×0.4	0.03×0.3
L	0.91×0.1	0.91×0.4	0.91×0.5	0.03×0.1	0.03×0.4	0.03×0.5	0.03×0.1	0.03×0.4	0.03×0.5	0.03×0.1	0.03×0.4	0.03×0.5

(Example 1)

In the future, to illustrate the learning path generation method by Markov decision process, we assume the sequence of actions that generate the learning path as follows. The intelligent tutoring system has to go through the basic concepts B0, B1, …, Bt, …, Bn, where the value of initial state Bt is H and the value of target state Bt+1 is H. The sequence of actions that the intelligent tutoring system performs is as follows: In the initial state Bt with value H, the intelligent tutoring reaches state Sbt,1 with value M by executing Down1, t-13, and then performing Down2,t1 reaches state St,2 with value L, and reaches the target state Bt+1 with value H by executing Up3,t2.

$$B_t(H) \xrightarrow{Down^3_{1,t-1}} Sb_{t,1}(M) \xrightarrow{Down^1_{2,t}} S_{t,2}(L) \xrightarrow{Up^2_{3,t}} B_{t+1}(H) \quad (3)$$

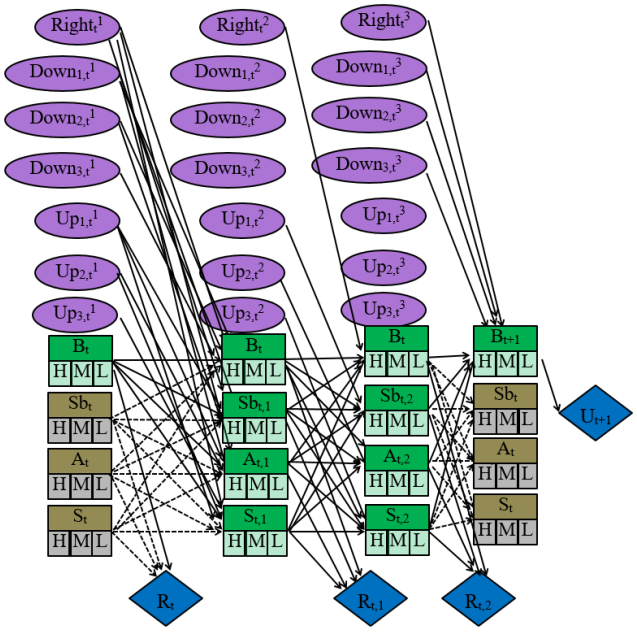

Figure 1. *Dynamic Decision Networks for Intelligent Tutoring Systems.*

d) applicability of the action

In Markov Decision Processes, state identification, i.e., uncertainty in sensing, is not present, and only uncertainty in action performance is assumed. That is, the intelligent tutoring system is able to specify exactly the set of actions that are feasible in each state variable, since the current state knows exactly which state variable is.

Representation of the reward value and utility function.

In the figure, the reward is represented by Rt and the utility function value by Ut.

3.2. Computational Methods

3.2.1. Calculation of Utility Functions and Policy

a) Calculation of utility function values

In Example 1, the probability of reaching the H state of the target state variable starting from the initial state variable is 0.913 = 0.7536, and the total utility value is (-0.2) + (-0.3) + (+1.5) = 1.0. The negative reward value is the value that motivates the student to quickly reach the key concept, the goal when performing individual learning with the intelligent tutoring system.

b) policy

A common way to solve MDP is dynamic programming, which solves the problem by recursively dividing the problem into small problems and storing the optimal solution in these small problems. A fixed sequence of actions cannot solve the problem, since it may end in some intermediate state rather than reaching the H state of the target basic concept state variable. Thus, the problem of how an agent should do in the state to be reached must be solved. This problem is solved by what is called policy. Traditionally, we denote the strategy π, π(s) is the action recommended by the direction π in state s. The policy is implemented starting from an initial state, and the statistical properties of the environment will produce different environmental histories. The evaluation of how good a policy is done by expected utility values of possible environmental history generated by the policy. The optimal policy is denoted

by π* as the policy with the highest expected utility. The balance between risk and reward, which gives a greater weight, varies depending on the reward value r = R(s, a, s') for transitions between non-target states.

3.2.2. Learning Performance Evaluation Method for Individual Learning Paths

In an intelligent tutoring system, the performance of learning through a learning path assigned to a student is calculated as the sum of the reward values in the experienced transitions. This performance evaluation method cannot use any method, and only utility functions for the environmental history are used, which we write Uh([s0,a0,s1,a1,…,sn]).

Then, we have to establish how to calculate the utility value for the state sequences. We use additive discounted rewards method (Russell et al. [1]). The utility value for a certain history is calculated as follows.

$$\text{Uh}([s0,a0,s1,a1,s2,\ldots]) = \text{R}(s0,a1,s1)+\gamma\text{R}(s1,a2,s2)+\gamma 2\text{R}(s2,a3,s3)+\ldots$$

The discount factor is the value between 0 and 1. The discount factor shows the degree of preference for future rewards by an agent with the current reward value. The closer γ is to 0, the less reward values for the distant future are treated as insignificant, and the closer γ is to 1, the longer rewards will be expected. If γ is positively 1, the discount factor becomes a special case of pure additive rewards. We use this for computation because additive discount factor greatly reduces the complexity in the histories estimation.

3.2.3. Optimal Policy, Utility Value of States

When we want to sum the utility value of a given history as the sum of the discount reward values, a comparison of policiesis made by comparing the expected utility values obtained at the execution of the policy. As mentioned above, we usually denote these state variables as a symbol s, since there are basic concept state variables(Bt), subconcept state variables(Sbt), auxiliary concept state variable(At), supplementary explanation state variable(St) as the learning state variables. Let us define the learning state variable as s now. Define the state that will arrive at time t when executing a specific policy st. When the initial state is S0 = st, the probability distribution of the state sequences (S1, S2,..) is determined by the initial state s, the policy π, and the transition model of the environment.

a) Action selecting using expected utility value

In Russell et al. [1], the following action selection method using expected utility is described. The expected utility value obtained when executing the policy π starting from the basic concept state variable Bt is. The state notation uses s, the general notation of the state variable.

$$\text{U}^{\pi}(S_t) = E\left[\sum_{t=0}^{\infty}\gamma^t \text{R}(S_t,\pi(S_t),S_{t+1})\right] \quad (4)$$

The expected value E depends on the probability distribution for the state sequences determined by St and π. Starting at the initial state St, one or more of all policies an agent can choose to perform an action, have a larger expected utility than the others. One of such policies is denoted by $\pi^*_{S_t}$.

$$\pi^*_{S_t} = \arg\max_{\pi} U^{\pi}(S_t) \quad (5)$$

$\pi^*_{S_t}$ also recommends some action for every state as a policy, especially if St is an initial state, then it is an optimal policy for the initial state. The use of continuous results of utility values at infinite limits makes the optimal policy independent of the initial state. Of course, the action sequence is not independent of the initial state, but the policy is a function that indicates the action in each state. In the future, we simply denote the optimal strategy by π^*.

From this definition, the actual utility value for the state St is $U^{\pi^*}(S_t)$. That is, $U^{\pi^*}(S_t)$ is the sum of the expected discount rewards when the intelligent tutoring system executes some optimal policy. This is simply denoted U (St).

This utility function U (St) allows us to choose actions using the maximum expected utility principle. That is, we choose the action that maximizes the value of the next step of the reward value plus the discount utility value that is then connected.

$$\pi^*_{S_t} = \arg\max_{a\in A(S_t)} \sum_{S'_t} P(S'_t \mid S_t, a)[R(S_t, a, S'_t)] \quad (6)$$

b) Method of calculation of utility function using Bellman equation

As above the utility function U (St) of a state is defined by calculating the sum of the forward expected discount rewards from that point. Hence, the utility value of a state is directly related to the utility value of a neighboring state. Assuming that the intelligent tutoring system selects the optimal policy, the utility value in a state is the sum of the expected reward associated with the transition to the next state and the discount utility value of the next state. Thus, the utility function of a state is given by

$$U(s) = \max_{a\in A(s)} \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma U(s')] \quad (7)$$

This expression is called the Bellman equation. Defining the utility of the state sequences as the expected utility of the sequence of continuous states as in Eq. (6) is a solution to the set of Bellman equations. In fact, the solutions are unique. In Figure 1, an example of the calculation of the Bellman equation for the state variable Bt with H values is shown below when the actions are applied in the order $Down^3_{1,t-1}$,

$Down_{2,t-1}^{3}$, $Down_{3,t-1}^{3}$, and $Right_{t-1}^{3}$.

Max{[0.91×0.85×(-0.1+γU(Sbt,1(H)))+0.91×0.1×(-0.2+γU(Sbt,1(M)))+0.91×0.05×(-0.2+γU(Sbt,1(L)))+
(0.03×0.85×(-0.1+γU(Bt(H)))+0.03×0.1×(-0.2+γU(Bt(M)))+0.03×0.05×(-0.3+γU(Bt(L))))+
(0.03×0.85×(-0.1+γU(St,1(H)))+0.03×0.1×(-0.2+γU(St,1(M)))+0.03×0.05×(-0.3+γU(St,1(L))))+
(0.03×0.85×(-0.1+γU(At,1(H)))+0.03×0.1×(-0.2+γU(At,1(M)))+0.03×0.05×(-0.3+γU(At,1(L))))],
[0.91×0.85×(-0.1+γU(St,1(H)))+0.91×0.1×(-0.2+γU(St,1(M)))+0.91×0.05×(-0.2+γU(St,1(L)))+
(0.03×0.85×(-0.1+γU(Bt(H)))+0.03×0.1×(-0.2+γU(Bt(M)))+0.03×0.05×(-0.3+γU(Bt(L))))+
(0.03×0.85×(-0.1+γU(St,1(H)))+0.03×0.1×(-0.2+γU(St,1(M)))+0.03×0.05×(-0.3+γU(St,1(L))))+
(0.03×0.85×(-0.1+γU(At,1(H)))+0.03×0.1×(-0.2+γU(At,1(M)))+0.03×0.05×(-0.3+γU(At,1(L))))],
[0.91×0.85×(-0.1+γU(At,1(H)))+0.91×0.1×(-0.2+γU(At,1(M)))+0.91×0.05×(-0.2+γU(At,1(L)))+
(0.03×0.85×(-0.1+γU(Bt(H)))+0.03×0.1×(-0.2+γU(Bt(M)))+0.03×0.05×(-0.3+γU(Bt(L))))+
(0.03×0.85×(-0.1+γU(St,1(H)))+0.03×0.1×(-0.2+γU(St,1(M)))+0.03×0.05×(-0.3+γU(St,1(L))))+
(0.03×0.85×(-0.1+γU(At,1(H)))+0.03×0.1×(-0.2+γU(At,1(M)))+0.03×0.05×(-0.3+γU(At,1(L))))],
[0.91×0.85×(-0.1+γU(Bt,1(H)))+0.91×0.1×(-0.2+γU(Bt,1(M))+0.91×0.05×(-0.2+γU(Bt,1(L)))+
(0.03×0.85×(-0.1+γU(St(H)))+0.03×0.1×(-0.2+γU(St(M)))+0.03×0.05×(-0.3+γU(St(L))))+
(0.03×0.85×(-0.1+γU(St,1(H)))+0.03×0.1×(-0.2+γU(St,1(M)))+0.03×0.05×(-0.3+γU(St,1(L))))+
(0.03×0.85×(-0.1+γU(At,1(H)))+0.03×0.1×(-0.2+γU(At,1(M)))+0.03×0.05×(-0.3+γU(At,1(L))))]

Now, to choose the optimal action in the initial state variable Bt, we have to iteratively calculate the function value for the remaining U.

c) Q function, which is an action utility function

Q(s, a) is the expected utility value when a given action is chosen for a given state.

$$U(s) = \max_{a} Q(s,a) \tag{8}$$

The optimal policy can be extracted using the Q function as follows:

$$\pi^{*}(s) = \arg\max_{a} Q(s,a) \tag{9}$$

The Bellman equation can be developed using the Q function, where the expected total reward value at which an action is selected is obtained by adding the reward value at the state immediately reached by the action to the value of the discount utility value at the resulting state.

$$\begin{aligned} Q(s,a) &= \sum_{s'} P(s' \mid s,a)[R(s,a,s') + \gamma U(s')] \\ &= \sum_{s'} P(s' \mid s,a)[R(s,a,s') + \gamma \max_{a'} Q(s',a')] \end{aligned} \tag{10}$$

Solving the Bellman equation using U(or Q) gives a solution to how to find the optimal policy. The Q function is obtained by the following method.

function Q-VALUE(mdp, s, a,U) returns utility

$$\text{return} \sum_{s'} P(s' \mid s,a)[R(s,a,s') + \gamma U(s')]$$

3.3. MDP Algorithm

There are a number of methods for generating offline solutions, such as value iteration, policy iteration, linear programming, and Monte Carlo planning as an online approximation algorithm. We use a value iteration algorithm.

The Bellman equation in Eq. (7) is the basis of a value iteration algorithm for solving MDP. If n possible states exist, there are n Bellman equations for each node. These n equations include n unknown utility values (utility values for states). Eventually, we have to solve the system of equations to find these utility values. The problem is that the equations are nonlinear since this "max" operator is not a linear operator. The linear equation system can be solved immediately using linear algebra techniques, but the nonlinear equation system still remains open. One possibility is the iteration method. The algorithm starts with any initial utility value, computes the right-hand side of the equation, and then holds it to the left-hand side of the equation. That is, we update the utility value of each state by its neighbor utility value.

Let Ui(s) be the utility value for state s at the ith iteration. Such an iteration is called the Bellman update.

[Algorithm 5. Value iteration algorithm computing the utility value of states]

function VALUE-ITERATION(mdp, ε) returns utility function value
inputs: mdp
ε
local variables: U, U'
δ
repeat
U← U'; δ ← 0
for each state s in S do
U'[s] ← maxa∈ A(s) Q-VALUE(mdp, s, a, U)
if |U'[s] − U[s]| > δ then δ ←|U0[s] − U[s]|
until δ ≤ε(1 − γ)/γ
return U

$$U_i(s) \leftarrow \max_{a \in A(s)} \sum_{s'} P(s' \mid s,a)[R(s,a,s') + \gamma U_{i-1}(s')] \tag{11}$$

We assume that for every iteration, this update is applied

continuously for all states. We ensure that if we apply this Bellman update infinitely many times, we will reach some equilibrium state (convergence of the value iteration). Then the final utility value must be the solution of the Bellman equation. This is a unique solution and the corresponding policy (obtained using Eq. (6)) is optimal. A detailed algorithm including the termination condition when the utility value is fully converged is shown in algorithm 5.

4. Experiment and Evaluation

A comparison experiment with Markov Decision Process model and other models was carried out. Two performance comparisons were conducted for the framework that there exists learning state prediction model and learning path generator separately, and the framework integrated with a learning state prediction model and a learning path generator. The first experiment was conducted between the integrated model with the model that separately exists.

Both Bayesian and Hidden Markov models are stochastic and statistical modeling methods. These two models represent one characteristic at a time point and the other one characteristic at a time interval. Also, when these two models are used in the development of intelligent tutoring systems, the model of student learning state prediction and the learning path generator are separately present. We have conducted a performance comparison experiment between the Bayesian model, the Hidden Markov Model and our Markov Decision Process model.

The second experiment was conducted between the models that integrated the learning state prediction model and the learning path generator. A performance comparison experiment between Markov Decision Process model, partially observable Markov Decision Process model and random model was conducted.

4.1. Performance Comparison Experiments Between the Integrated Model with the Model that Separately Exists

Theoretically, if a state is generally composed of n states with at least d values, then the corresponding HMM transition matrix is of size O(d2n) and its size is O(d2n) per update count [1]. On the other hand, in the Markov Decision Process model, when the number of parents of each variable is limited to k, its size is O(ndk). This means that the number of variables has a linear rather than exponential nature.

A pre-testing was conducted on 200 students who did not use the intelligent tutoring system, and then students were self-studying using the intelligent tutoring system. The results of the final examination of students who completed their own studies with the intelligent tutoring system and the analysis of the M (Means) and SD (Standard Deviations) of the scores are shown in the table below (10-point criterion).

Table 3. *Evaluation Through Means (M) and Standard Deviations (SD).*

	pretest mean (M)	pretest standard deviation (SD)	final test mean (M)	final test standard deviation (SD)
Bayesian Model	6.54	1.85	6.57	1.41
MDP	7.55	1.75	8.27	1.49
HMM	7	1.84	7.32	1.66

The analysis shows that the use of the Markov Decision Process model results in higher mean values of students' final tests and lower standard deviations than other models.

4.2. Comparison Between MDP, POMDP, and Random Model

1) Experimental Preparation

The effectiveness of intelligent tutoring systems generally depends on learning paths, where policies are used to determine what actions to take next for learning path selection. We computed the learning path-determining policies based on reinforcement learning structures such as POMDP and MDP, given the limited feature space, and compared the policies induced by reinforcement learning with the random and rational strategies. In the experiments, we also used a simple baseline educational policy in which the system randomly selects a worked example or a problem solving in the next time step. The worked example is a form of presenting a problem together with its solution, and a problem solving is a form of presenting a question or a problem to be solved.

a) experimental condition

124 students randomly participated in one of the three methods evaluation experiments.

MDP (N = 45), POMDP (N = 40), Random (N = 39)

In addition, 124 students were classified as fast and slow in response time.

fast group(n = 61), slow group (n = 63)

Combining these two groups and three methods, six experimental groups were obtained.

MDP-Fast(N = 22), MDP-Slow(N = 23), POMDP-Fast(N = 18), POMDP-Slow(N = 22), Random-Fast(N = 21), Random-Slow(N = 18)

$\chi 2$-test($\chi 2$ = 0.03, p = 0.86) showed no significant difference between the fast and slow classes among the three methods. Here, the $\chi 2$ -test is a method that uses the $\sum^2$distribution when testing whether the theoretically assumed distribution of the population and the distribution obtained in the real sample match each other.

2. Training procedure

Students have to solve three or four problems per level and 18-24 problems in total. Level 1 sets up a preliminary test phase to assess student's initial ability, and students accept the same problems that exist at Level 1. All of these problems take the form of PS(problem solving). To fairly assess the students' learning status from level 2 to level 6, students must solve the problem PS before completing each level. The policies implemented during training allow different decisions to be made. The intelligent tutoring system makes 10-15 decisions to complete each student's training.

3. Assessment of Learning Status

To fully assess students' performance (learning status), a final test (or post-test) was organized. Students' test results were rated by a 1-100 point by a teacher other than a member of the study group. Posttest results were taken as the final assessment of students' learning.

4. Collection of training data

Training data were collected between 2023 and 2024. All students were subjected to the same training procedure, the same training data as the same intelligent tutoring system, and only the randomly determined training data were different from the WE and PS. The training dataset contained a history of 306 students interacting with the intelligent tutoring system, the average number of students' solutions was 23.7, and the average time that students had with the intelligent tutoring system was 5.29 h. There are 133 students' learning behavioral features, and the experiments generated the same feature space for MDP and POMDP using the MDP-based feature selection method proposed by Shen [22]. It has six features as follows.

a) totalPSTime: Total time spent to solve PS
b) easyProbCount: The number of easy problems that a student has solved so far.
c) newLevel: Are students jumping to a new level?
d) avgStepTime: Average step time to date
e) hintRatio: The ratio between the number of times a reminder is given and the number of times a rule is applied.
f) NumProbRule: The number of rules in the current problem solution

2) Experimental results

The students' pre-test scores showed that the initial performance of the students in the experiment was equal without significant difference. Table 4 shows the M and SD values of the pre-test and post-test scores for each group as a measure of 100 points. The experimental results confirm that the students' ability to pass through the learning process by generating learning paths using Markov Decision Process methods is higher than those of students passing the learning path generated using other methods.

The experimental results showed that the MDP-based approach can significantly improve student learning than the random criterion when the learning content is coordinately controlled, and the POMDP-based approach does not perform better than MDP. The reason is that the features selected for MDP structure cannot be the optimal feature space for POMDP.

Table 4. *Pretest and Posttest Scores for Each Group.*

policy	Pre-test			Post-test		
	overall	fast	slow	overall	fast	slow
MDP	74.90(26.3)	75.34(27.6)	74.48(25.5)	88.26(15.2)	84.23(17.7)	92.12(11.3)
POMDP	75.18(25.9)	74.01(29.1)	76.15(23.2)	79.53(24.4)	86.47(23.6)	73.86(24.1)
random	65.99(28.1)	67.69(28.8)	64.02(27.8)	82.85(22.3)	88.98(17.9)	75.69(25.3)

5. Conclusion

MDP is a good way to apply reinforcement learning. However, this MDP is a method that can only be used when the students' learning state is assumed to be fully observable.

In our study, we proposed a method to generate learning paths for intelligent tutoring systems using MDP. To do this, we defined state and transition probabilities and proposed an algorithm to select the optimal learning action according to the learning policies.

The experimental results showed that the MDP showed superior performance compared to other methods without reinforcement learning algorithms. However, there was no significant difference between MDP and POMDP.

Abbreviations

MDP	Markov Decision Process
POMDP	Partially Observable Markov Decision Process
MEU	Maximum Expected Utility
IOHMM	Input-Output Hidden Markov Model
DBN	Dynamic Bayesian Network
DDN	Dynamic Decision Network
S	Slides
Co	Comments
V	Videos
QA	Questions and Answers
LP	Learning Path
H	High
M	Medium
L	Low
HMM	Hidden Markov Model
M	Means
SD	Standard Deviations

Author Contributions

Song-Hwan Kwon: Project Administration, Writing – original draft, Conceptualization
Jong-Nam Rim: Writing-review & editing, Methodology
Chung-Song Ko: Software, Validation
Un-Song Ryu: Formal Analysis, Software
Yong-Jin Pak: Investigation
Hyon-Il Son: Software, Formal Analysis

Conflict of Interest

The authors hereby declare no conflict of interest.

References

[1] Russell, S. J., & Norvig, P. (2022). Artificial Intelligence: A Modern Approach. Prentice Hall. Retrieved from http://www.amazon.com/Artificial-Intelligence-ApproachStu-art-Russell/dp/0131038052

[2] Burhan Aji S. (2022). Intelligent Tutoring System Design Using Markov Decision Process. Emerging Information Science and Technology. Vol. 3, No. 1, pp. 1 9-28.

[3] Iglesias, A., Martinez, P. and Fernandez, F. (2003). An experience applying reinforcement learning in a web-based adaptive and intelligent educational system. Informatics in Education. vol. 2, no. 2, pp. 223-240.

[4] Litman, A. J. and Silliman, S. (2004). Itspoke: An intelligent tutoring spoken dialogue system. in Proc. Human Language Technology Conference 2004.

[5] Sarma, B., and Ravindran, B. (2007). Intelligent tutoring systems using reinforcement learning to teach autistic students. Home Informatics and Telematics: ICT for The Next Billion, Springer, vol. 241, pp. 65-78.

[6] Iglesias, A., Martnez, P., and Fernndez, F. (2009). Learning teaching strategies in an Adaptive and Intelligent Educational System through Reinforcement Learning. Journal Applied Intelligence, Volume 31 Issue 1, August 2009. 58.

[7] Ai, H., Litman, D. J., Forb es-Riley, K., Rotaru, M., Tetreault, J., and Purandare, A. (2006). Using system and user performance features to improve emotion detection in sp oken tutoring dialogs. In Proceedings of the International Conference on Spoken Language Processing (Interspeech 2006 (ICSLP), pages 797–800, Pittsburgh.

[8] Janarthanam, S., Hastie, H., Lemon, O., and Liu, X. (2011). The day after the day after tomorrow: a machine learning approach to adaptive temporal expression generation: training and evaluation with real users. SIGDIAL '11 Proceedings of the SIGDIAL 2011 Conference (Pages 142-151), Stroudsburg, PA, USA 2011.

[9] Pan, Y. Lee, H., and Lee, L. (2012). Interactive Spoken Document Retrieval With Suggested Key Terms Ranked by a Markov Decision Process. IEEE Transactions on Audio, Speech, and Language Processing archive Volume 20 Issue 2, February 2012, Page 632-645.

[10] Chi, M., Lehn, K., Litman, D., and Jordan, P. (2011). Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. User Model User-Adap, Kluwer Academic, pp. 137-180.

[11] Iglesias, A., Martnez, P., Aler, R., and Fernndez, F. (2009). Learning teaching strategies in an adaptive and intelligent educational system through reinforcement learning. Applied Intelligence, vol. 31, no. 1, pp. 89-106.

[12] Williams, J. D., and Young, S. (2007). Partially observable Markov decision processes for spoken dialog systems. Elsevier Computer Speech and Language, vol. 21, pp. 393-422.

[13] Theocharous, G., Beckwith, R., Butko, N., and Philipose, M. (2009). Tractable POMDP planning algorithms for optimal teaching in SPAIS. in Proc. IJCAI PAIR Workshop 2009.

[14] Rafferty, A. N. et al. (2011). Faster teaching by POMDP planning. in Proc. Artificial Intelligence in Education (AIED) 2011, pp. 280-287.

[15] Chinaei, H. R., Chaib-draa, B., and Lamontagne, L. (2012). Learning observation models for dialogue POMDPs. Canadian AI'12 Proceedings of the 25th Canadian Conference on Advances in Artificial Intelligence, Springer-Verlag Berlin, Heidelberg, pp. 280-286.

[16] Folsom-Kovarik, J. T., Sukthankar, G., and Schatz, S. (2013). Tractable POMDP representations for intelligent tutoring systems. ACM Transactions on Intelligent Systems and Technology (TIST) -Special Section on Agent Communication, Trust in Multiagent Systems, Intelligent Tutoring and Coaching Systems Archive, vol. 4, no. 2, p. 29.

[17] Julien Seznec. (2020). Sequential machine learning for intelligent tutoring systems. Machine Learning [cs.LG]. Universit é de Lille, 2020. English. ffNT: LILUI084ff ffel-03490620f

[18] Whiteley, W. (2005). Artificially Intelligent Adaptive Tutoring System, Education, IEE Transactions on, Volume 48, Issue 4.

[19] Jeremiah, T. F., Gita, S., and Sae, S. (2013). Tractable POMDP representations or intelligent tutoring systems. ACM Transactions on Intelligent Systems and Technology (TIST) - Special section on agent communication, trust in multiagent ystems, intelligent tutoring and coaching systems archive, Volume 4 Issue 2, March 2013.

[20] Wang, F. (2018). Reinforcement Learning in a POMDP Based Intelligent Tutoring System for Optimizing Teaching Strategies. International Journal of Information and Education Technology, Vol. 8, No. 8, August 2018, pp 553-558.

[21] Hamid, R. C., Brahim, C., Luc, L. (2012). Learning observation models for dialogue POMDPs. Canadian AI'12 Proceedings of the 25th Canadian conference on Advances in Artificial Intelligence, Pages 280-286, Springer-Verlag Berlin, Heidelberg 2012.

[22] Shen, S. (2020). Empirically Evaluating the Effectiveness of POMDP vs. MDP Towards the Pedagogical Strategies Induction, AIED 2020, LNAI 10948, pp. 109–113. https://doi.org/10.1007/978-3-319-93846-2_21

[23] Li, J., and Hou, L. (2017). Review of knowledge graph research. J. Shanxi Univ., Natural Sci. Ed., vol. 40, no. 3, pp. 454–459, Mar.

[24] Chen, Z. et al. (2020). Knowledge Graph Completion: A Review. IEEE Access, 2020.3030076, VOLUME 8.