Science Frontiers
2025, Vol. 6, No. 4, pp. 170–200
https://doi.org/10.11648/j.sf.20250604.16

Research Article

Intelligent Tutoring System Framework and Learning Platform Development

Song-Hwan Kwon[1, *], Yun-Ho Ko[1], Chung-Song Ko[1], Jong-Nam Rim[1], Sin-Hye Jang[1], Yun-Sok Ham[2], Ha-Gyong Kim[1], 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

As a general-purpose intelligent tutoring system framework for the development of intelligent tutoring system for their subjects, any subject teachers at universities and schools at all levels developed a platform for the operation of editors and intelligent tutoring systems. The knowledge graph is used to extend the subject graph, hyper-concept graph, basic concept graph, subconcept and help-concept graph. The intelligent tutoring system is designed to emulate the teaching style of human teachers, but to provide individual teaching to each individual student. Therefore, the skills included lecture slide display, video playback, comment display, comment reading by speech synthesizer, student recognition by face recognizer, question presentation and response processing, response results processing by natural language processing, and student's learning performance prediction et al. Hidden Markov models are known as models that can probabilistically and statistically model the learning path and history of students through time intervals, estimate learning states, and predict learning outcomes. The developed intelligent tutoring system platform estimated and predicted the learning status of the student by the Hidden Markov model and then generated the learning path using a specific algorithm. We also directly generated the learning path using the Markov decision process model. The results of the analysis of the results, using the general-purpose intelligent tutoring system development framework, were very positive, while developing the intelligent tutoring system for AI subjects and allowing students to use it.

Keywords

Tutor Softbots, Intelligent Tutoring Systems, Individual Instruction, Knowledge Graph, Intelligent Tutoring System Platform

1. Introduction

The paper is organized as follows. In Section 1, we describe the areas of interest for system development and previous works and in Section 2 we define some of the concepts we use. Section 3 describes the implementation of group teaching functions, learning by static learning path as a low step in the development of intelligent pedagogy, Section 4 the construction of knowledge graph and editor, Section 5 the construction of learning state prediction model using

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

Received: 16 November 2025; **Accepted:** 3 December 2025; **Published:** 30 December 2025

Hidden Markov model and learning path generation mechanism construction method, and Section 6 the learning path generation method using Markov decision process model. Section 7 presents the performance comparison experiments with our proposed learning path generation methods and other models and the evaluation.

In previous studies [22, 21], Intelligent Tutoring Systems is defined as a teaching system for self-learning to provide appropriate learning content and learning environment, based on evaluating individual learners' knowledge level, learning ability and learning style. In other words, intelligent tutoring system is an educational software that has artificial intelligence elements. These software tracks the student's learning process, implements feedback control, and gives hints to the learning process. In the literature, it is stated that the intelligent tutoring system is generally composed of three models: Learner Model, Tutor Model and Domain Model, and the intelligent tutoring system (ITS) model is formalized as follows.

ITS=(LM, TM, DM)

Here, LM is the learner model, TM is the tutor model, and DM is the domain model. The LM is a model with the ability to evaluate and analyze the state of the learner learning through an intelligent tutoring system. The TM is a model with a learner-specific learning-guidance function based on the tutor's experience and the learners' state analysis reflected in the LM. The domain model DM is a model with the function of controlling the interaction between the learner model, the teacher model and the teaching knowledge base.

In our study, we define an intelligent tutoring system as an intelligent agent that embodies intelligence rather than a simple program, a guiding system or a lecture system that has the recommendation function for the learning process. In addition, our study uses the learner model, the student model, the learning state prediction model, the learning performance prediction model as synonyms, the teacher model, the learning path generation model, and the learning path generator as synonyms. In particular, the remainder of our work, which is a previous work, mainly uses the term Learning State Prediction Model and Learning Path Generator.

Previous studies have proposed different intelligent instructor architectures, which include a framework that separates the learning state prediction model from the learning path generator, and a framework that integrates the learning state prediction model and the learning path generator (Figure 1). In Figure 1, LPG is an abbreviation of Learning Path Generator, and LSEPM is an abbreviation of Learning State Estimation & Prediction Model. Learning state estimation and prediction models are models that summarize the past history of student learning activities, such as understanding, cognition, and response outcomes, and that, when input is given as evidence, estimate the current learning state (cognitive level) of the student and the current learning activity, and output the learning state prediction value of the next concept. Thus, a learning state prediction model is a model that can quantify the degree of student perception.

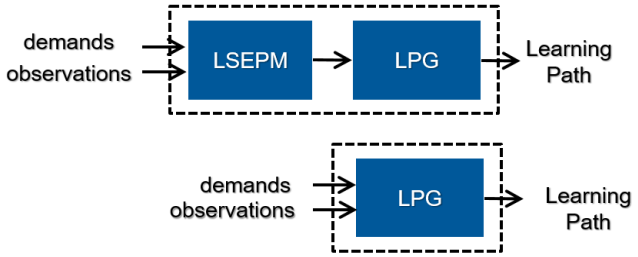

Figure 1. *Intelligent Tutoring System Architecture.*

In the study [23] of the adaptive learning system, an adaptive engine consisting of two blocks was developed, consisting of a Bayesian knowledge tracking to estimate the learning state and a recommendation engine that uses the output of knowledge tracking as input.

1.1. Objectives of Our Research Team in the Research and Development of Intelligent Tutoring Systems

Our research team's immediate goal is to develop a logical, systematic and software infrastructure for intelligent tutoring systems research and development ahead of others. It is not to develop an intelligent tutoring system in an embedded and closed form in cooperation with an artificial intelligence expert, a program development expert, and a subject expert, but in an open and service form to enable any subject teacher to develop an intelligent tutoring system for his or her subject. Thus, it is not limited to the development of intelligent tutoring systems as a very limited number of specialists, but to encourage the development of intelligent tutoring systems by allowing many teachers to do so.

1.2. Research Topics Needed to Develop the Intelligent Tutoring System Development Framework and Learning Platform

The research topics needed to develop the intelligent tutoring system development framework and learning platform are briefly reviewed, taking the previous literature as an example.

1.2.1. Research and Development Status of Existing Intelligent Tutoring System Development Framework and Learning Platforms

Intelligent tutoring systems fall within the category of adaptive education systems [2-6]. AISs(Adaptive Instructional Systems) include Cognitive Tutor, AutoTutor, GIFT(the Generalized Intelligent Framework for Tutoring), Dragoon, ASPIRE-Tutor, TLA(the Total Learning Architec-

ture), and ALOSI(Adaptive Learning Open Source Initiative) [2, 6-12, 23]. The research on these systems attempts to standardize learners modeling, teaching strategies, and domain modeling, which increases the potential effectiveness of reinforcement learning strategies used to improve the accuracy and effectiveness of predicting learners' status and making instructional strategies decisions [6].

The framework of the intelligent tutoring system, GIFT, has made use of the learning resources(lecture presentation, video, comments, question answering, etc.) used in the learning path in the framework by objectification, visualization. It also provides the modeling of the course of the intelligent tutor and the compilation of the content of the lessons by objectification, visualization, and streamlining [1]. Any teacher, researchers, can use this GIFT to develop an intelligent tutoring system for their subject. Figure 1 shows the course objects of GIFT and a configured set of objects for a lesson, as presented in previous work [1].

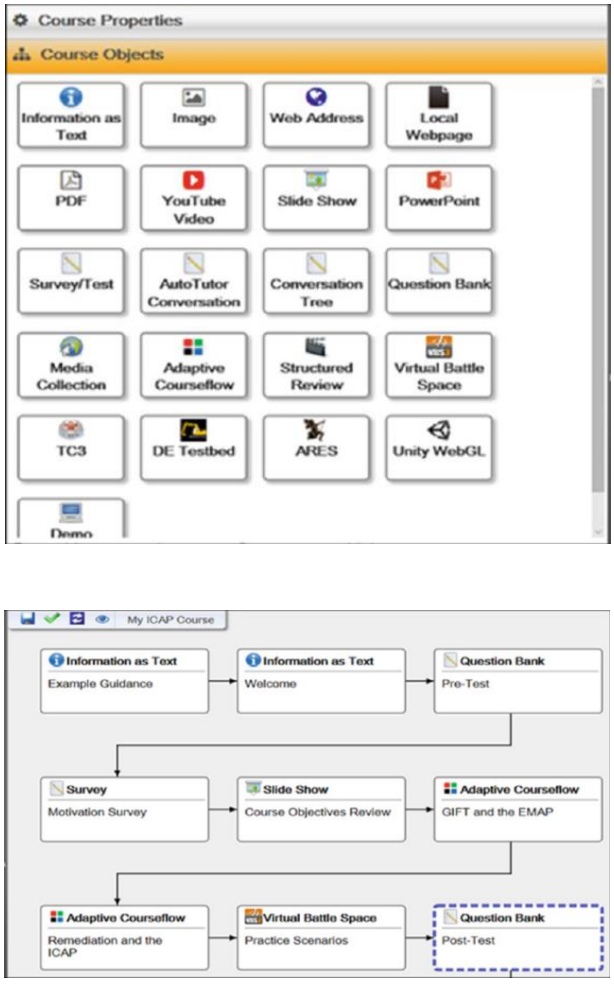

Figure 2. *Available GIFT Course Objects (Left) and a Configured Set of Objects for a Lesson (Right) [1].*

1.2.2. Identification of Knowledge Representation Methods

One of the foregoing challenges for the development of intelligent tutoring systems is to identify ways in which knowledge can be expressed and inferred.

In Ramirez et al. [13], the authors categorized knowledge, defined knowledge representation models, and described the type of knowledge representation models. The types of knowledge include declarative, procedural, structured and unstructured knowledge, and the types of representation models used in knowledge systems include distributed, symbolic, non-symbolic, declarative, probabilistic, and regular types, which are selected to fit specific inference types such as deductive, inductive, analogical, and hypothesis-making [14]. Symbolic systems have a semantic network, a rule-based system, a frame-like structure, and distributed systems have neural networks and probabilistic networks. Figure 3 shows a summary of the semantic network as a symbolic system and the neural network as a distributed system.

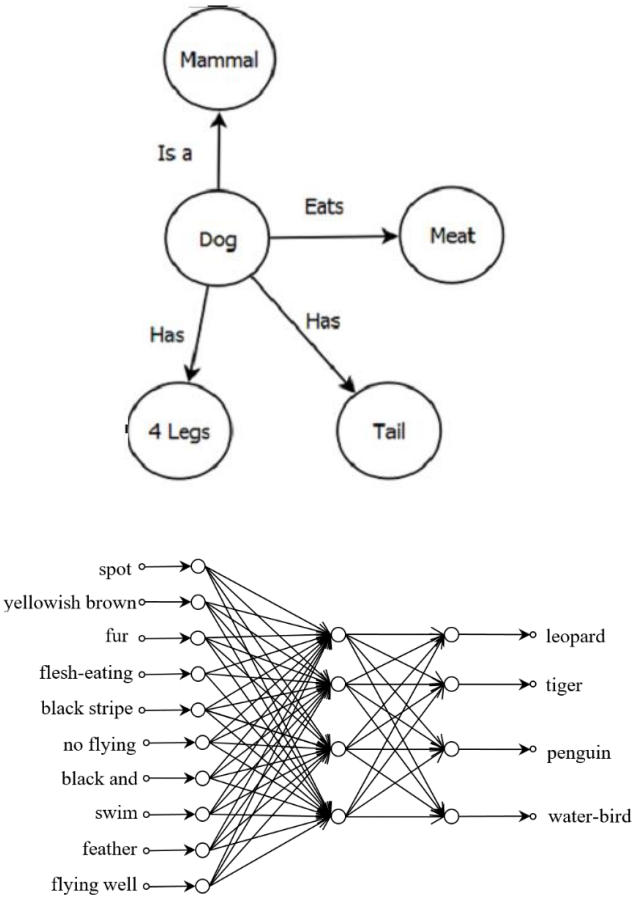

Figure 3. *Example of a Semantic Network for Symbolic Representation and a Neural Network for Distributed Representation.*

A semantic network is a concept network, in which concepts are represented by nodes, relations are represented by

arcs, and are defined by graph notation for propositional logic [15]. A rule-based system is a symbolic representation model with procedural knowledge as the main object, which consists of a rule library in the form of condition-action [13]. Frames can be viewed as a form of semantic networks that incorporate declarative and structured procedural knowledge [13]. Neural networks are distributed knowledge representation models that do not use symbols for concept representation [13]. How ontologies are used in computer science and artificial intelligence has been discussed in many literatures [13]. ontology is a clear representation of a concept, concept is an abstract and simple representation of the environment we want to express, ontology does not discuss facts or benefits, and it is a harmonious document that is created for social relations to achieve some goal [13, 16].

Through Ramirez et al. [13], the authors state that Memory Map is a knowledge representation model of concepts and skills, and its purpose is to represent the interactions between them in different contexts, including the representation of concepts such as context-specific semantic changes. Memory maps can be represented by semantic networks, and the difference between memory maps and other models is that they are of great interest to context flexibility.

In Chen et al. [18], the authors review the knowledge graph, a kind of semantic network widely used in natural language processing, intelligent query answering system, intelligent recommendation system, intelligent tutoring system, etc. With big data, deep learning, knowledge graph has become one of the core drivers for AI development today [18]. the concept of knowledge graph was first proposed by Google in 2012 and knowledge graph is defined as a large-scale knowledge base consisting of many entity nodes and their relationships [17]. In previous studies, the partitioning methods of knowledge graph, knowledge graph, and query response have been mostly proposed for partition storage and query response on large data in the network.

Our study seeks to select a knowledge representation model that is strong and extensible in knowledge representation, extensible, easily understandable by general teachers and easy to develop programs and systems.

1.2.3. Development of an ITS Editor to Provide an Intelligent Tutoring System Development Framework

The teaching resources of an intelligent tutoring system must be represented by certain data structures or built into a database. Using this data structure or database, we estimate the learning status of a student or generate a learning path. However, it is very difficult for general users(common ITS developers of their subjects) to build such data structures or databases directly. This is because the IT level of the common developers developing intelligent tutoring systems is different. To solve these challenges, we provide them with an ITS Editor as an ITS development framework.

In Goldberg et al. [1], the Course Objects of Course Properties were provided to select learning contents (learning objects) and the My ICAP Course was provided to set the learning path directly statically or to allow the system to dynamically generate automatically.

1.2.4. Challenges in Developing a Learning Platform

1) Development of the inference engine

The problem of developing an engine that can evaluate and infer the content of knowledge representation provided through the editor must be solved. This is referred to as the study of student learning state estimation and prediction model construction and learning path generation.

2) Building a data structure and database that can arrange learning objects such as lecture slides, video, comment, query and response into a time series.

This is not a study, but a matter of system development. The learning objects must be stored in a computer with a certain structure, either by defining a data structure or by building a database. The construction of data structures is very complex and therefore it seems much easier to build them into a database.

3) To solve the problems of applying artificial intelligence techniques such as speech recognition, speech synthesis, face recognition, and emotion recognition for smooth interaction between the intelligent teacher and the student.

The intelligent tutoring system to be developed can give students artificial emotions only by visual inspection as a general e-learning system. To get the student to feel that he or she is actually working with a human teacher who teaches himself or herself, we must implement the human skills, such as speech recognition, speech synthesis, face recognition, and emotional recognition, in the system. So the system can perceive the student's voice, read the relevant visual text data provided by the system, identify the student's face and call his name, and even understand his emotional state, so that it can be used for feedback control in teaching.

4) Student Response Results Analysis and Evaluation Techniques by Natural Language

Currently, many intelligent tutoring systems are being developed in a discourse, in which natural language questions and responses are essential. Eventually, the system must be able to interpret what the student has responded to in natural language.

2. Definition of Some Concepts

We define the following concepts.

2.1. Group and Individual Teaching

We define collective teaching as a teaching activity that is conducted in a classroom with dozens of students. The framework and platform we are developing is intended to add a collective teaching function that works with multiple

static models. The activity of a human teacher or intelligent tutoring system that deals with individual students as one-to-one is defined as a tutorial.

2.2. Teaching Flow

The human teacher performs the teaching activities with a kind of ordered teaching direction to how to teach according to his or her educational experience and academic content. This teaching direction is defined as the teaching flow. Some studies also call the teaching flow a TS(Tutoring Strategy). The TF(Tutoring Flow) consists of a sequence of LS(Lecture State) in discrete visual steps.

$TF = \{LS_1, LS_2,...,LS_m\}$,

where m is the number of visual stage. In group teaching, the number of lecture states present in a section, and in individual teaching, the number of basic concepts present in the subject. The LS_t at t visual stage consists of a set of S(Slides), Co(Comments), V(Videos), QA(Question and Answer).

$LS_t=<S_t(S_t^1, S_t^2,..., S_t^{ns}), Co_t(Co_t^1, Co_t^2,..., Co_t^{nc}), V_t(V_t^1, V_t^2,..., V_t^{nv}), QA_t(Q_t^1: A_t^1, Q_t^2: A_t^2,..., Q_t^1: A_t^{nq})>$,

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

2.3. Teacher Model

Teaching resources such as slides, corresponding comments, videos, and questionnaire data used simultaneously with the human teacher's ordered teaching flow are collectively called teacher models in AI. The teacher model will also be called the instructional model. In an intelligent tutoring system, a teacher model may exist statically many and may be dynamically generated.

3. Implementation of Group Teaching Skills

3.1. Stepwise Teaching Principle

In the past, the research and development of intelligent tutoring systems has proposed step by step (step) in the process of developing a research to implement the teaching methods of human teachers in machines(Vande Pol et al. [20]). This approach is a way of implementing a framework called KCDs(Knowledge Construction Dialogues), which guides all students according to a pre-planned "guide direction of inference", taking into account that students' abilities are different, KCD applies how to exit from the main course of the course and return to the main path before it, when the student has a wrong answer, for the sub-session for "therapeutic" teaching. This approach requires repeated learning for a long time and does not take into account the individual student's level of knowledge, which can be disappointing and inefficient for some students.

The teacher construct few static student' model manually using his or her teaching experience and academic knowledge, apply a step-by-step approach to teaching, and return to the lesson when judged to have failed to understand the content of the current teaching. That is, The lecture on the subject that is not understood is repeated, we revisit the supplementary lecture on the wrong problem and then the previous question was re-presented and the 10 points were passed to continue the next topic. In this way, the teacher's unilateral teaching flow prevails because he is not in the mind of the student, the teacher is motivated to take education and the student is in a passive state. It is also a unilateral teaching mode that does not take into account each individual student, i.e., each student will have different learning efficiencies and different learning outcomes.

3.2. Collective Instructional Data Structures with Text Representations

The data needed for group instruction is first edited in text by the subject teacher and the database edited using the provided editor is imported into the program. In our proposed intelligent tutoring system, data structures for collective teaching data that are used by the algorithm of this collective teaching model are constructed using a database. The following is a text-based data structure for a subject.

[Chapter 1] Basics// [Section 1] Concept//(Slide 1) Slide1.png//(Comment 1) From this time I will study this subject with you. //(Slide 2) Slide5.png//(Comment 2) The curriculum of the subject is as follows.//(Video 2) //(Questions for topic 1) (Question 1, 2, Audio) What is the meaning of artificial? (a)imitative (b) spurious (c) human/answer: (a)5(b)5(c)0//(Question 2, 2, Audio) What is the meaning of intelligence? (a) Ability to acquire and apply knowledge (b) Ability to think and reason (c) Ability to understand and benefit from experience /answer: (a)4(b)3(c)3//(Question 3, 2, Audio) What are there in human intelligence?//(a) Human intelligence includes thinking, reasoning, oral speaking, ear listening, language, visual understanding, nose smell, and skin feeling. (b) Intelligence of man involves intelligent behavior and action./Answer: (a)5 (b)5 //(Question 4, 2, Audio) Why do AI technologies and products need? (a) To replace what man does.(b) To enhance human intelligence.(c) To make an autonomous being like a human being. /answer: (a)3(b)3(c)4//(Question 5, 3, Audio) What is AI defined in a narrow range? Theory and development of (b)? systems that perform tasks that (a) such as perception, classification, learning, abstraction, reasoning, and behavior./answer: (a) human intelligence, machine intelligence (b) computer// [Question 6, 4, Audio] What is AI defined in a broad and general context? (Correct response) To

develop a fully autonomous agent if AI is defined in a broad sense./(Answer)(a) Fully autonomy (b) Agent// (Supplementary lecture for topic 1)(slide 2-1) Slide6.png.//(Comment 2-1) Artificial means imitative and spurious, and artificial intelligence means fake intelligence and intelligent mimic.// (Video 2-1) 1-1.mp4.// (slide 2-2) Slide7.png//(Comment 2-2) Artificial intelligence technology and products began to develop from the human needs to make machines that substitute human intelligence and function or at a higher level, and this is moving towards the desire to develop fully autonomous machine agent with the rapid development of technology and computing resources.//(slide 2-2) Slide7.png//(Comment 2-2) Human intelligence includes perception, classification, learning, abstraction, inference, and behavior.//(Video 2-2) 1-1.mp4//(slide 2-3) Slide7.png//(Comment 2-3) Artificial intelligence technology and products began to develop from the human needs to make machines that substitute human intelligence and function or at a higher level, and this is moving towards the desire to develop fully autonomous machine agent with the rapid development of technology and computing resources.//(Video 2-3)//(slide 2-4) Slide7.png//(Comment 2-4) Artificial intelligence is concerned with the theory and development of computer systems that, in a narrow sense, perform tasks that augment human intelligence such as perception, classification, learning, abstraction, reasoning, and behavior, and, in a wide range, develop fully autonomous agents.//(Video 2-4)// (slide 3) Slide8.png//(Comment 3).....//(Video 3)//(Section 1.1. Final Testing)//(Question 1, 1, audio) In the above lecture, we have defined intelligence qualitatively and reasonably. Define intelligence more quantitatively. Choose one of the quantitative definitions of intelligence below which you think is right. (a) Intelligence refers to the ability to apply knowledge to achieve better performance in a certain environment. (b) 2. It is an intelligent action to put your hand on the hot stove and take it off without knowing it./(Answer) (a) 10 (b) 0// (Question, 2, 2, audio) Choose a single reflection of the following items. (a) The ability to distinguish the car that appeared ahead of the way. (b) The ability to distinguish dogs that appear in the front of the road. (c) the movement of putting your hand on the hot stove and removing it without knowing it./(Answer) (a) 5 (b) 5 (c) 0//(Question 3, 3, audio) Complete the following sentence. AI? This means the research and development of (c) that allow actors with a given (a) ? resource to perform better in a given (b).//(Answer): (a) Computation, (b) Environment, (c) Agent Program, Physical Machine//(Question 4, 4, Audio) Answer what AI is./(Answer) thought, behaviors, and machines.

3.3. Building a Group Teaching Database

The collective instructional data structure used in the program was implemented using the database, which is as follows.

Table 1. *Group Teaching Database.*

record name	Data type	main key	meanings
Chapter table			
ChapterID	INTEGER	√	Chapter number
ChapterTitle	TEXT		Chapter title
Section table			
ID	INTEGER	√	unique identifier of section
ChapterID	INTEGER		the number of the chapter in which the section belongs
SectionID	INTEGER		the number of the section in the chapter
SectionTitle	TEXT		section title
Lecture table			
comment: Each slide is represented as a line of this table and is separated by the value of the SectionID field. In order to obtain the slide information for Section 1.2, we obtain the unique identifier(ID) of the row corresponding to Section 1.2 in the table and return only the rows with the same value and SectionID.			
ID	INTEGER	√	unique identifier of slide
SectionID	INTEGER		Unique number of section containing a slide, In what section is the slide?
SlideName	TEXT		"Subject name/group teaching/slide" Name of the picture file in the path (including the exten-

record name	Data type	main key	meanings
			sion)
Comment	TEXT		comment
VideoName	TEXT		"Subject name/group teaching/slide" Name of a movie file in the path (including an extension)
ShowOrder	INTEGER		Numerical values for slide display order
hasTest	INTEGER		If there is questions for slide, then 1, 0, if no.
FinalTest table			
comment:			
In group teaching, there is a final question at the end of each section.			
The information is a table. As in this table, in all test-related tables, the test data is entered in XML format. The test editor that prints the test problem in XML has been developed and is therefore used. Each row represents one test problem. Thus, the way to obtain the necessary test data is as in the table above, so that ParentSection only obtains the same rows as the unique number of the section.			
ID	INTEGER	√	unique identifier
XMLData	TEXT		XML data for testing
ParentSection	INTEGER		corresponding section number
Table for SlideTest			
comment: In the slide of the Lecture table, if "hasTest" is 1, there is an slide. Then, we get the same lines of ID value of the slide line and the same value of ParentSlide in the SlideTest table and then ask.			
ID	INTEGER	√	unique identifier
XMLData	TEXT		XML data for testing
ParentSlide	INTEGER		unique number of the slide, The ID of slide in the Lecture table.
SlideName	TEXT		The name of the picture file (including extension) in the path of "the subject name/group teaching/slide"
Comment	TEXT		comment
VideoName	TEXT		The name of the movie file (including extension) in the path of "the subject name/group teaching/slide"

3.4. Group Teaching Algorithm

The above definition of the teaching flow is given, and the algorithm works in units of LS_t, the lecture topic at each time.

[algorithm 1. Group teaching algorithm]

1: function General_Teaching(GT_MDB)
2: {
3: for(i=0; i<n_{max}; i++) // n_{max}=max{n_s, n_e, n_v, n_q}
4: {
5: ith slide display;
6: ith movie playback;
7: for(j=0; j<n_q; j++)
8: {
9: jth question;
10: jth answer;
11: while (result mark < threshold mark)
12: {
13: Additional lecture on the jth question;
14: jth query suggestion;;
15: }
16: }
17: }
18: }

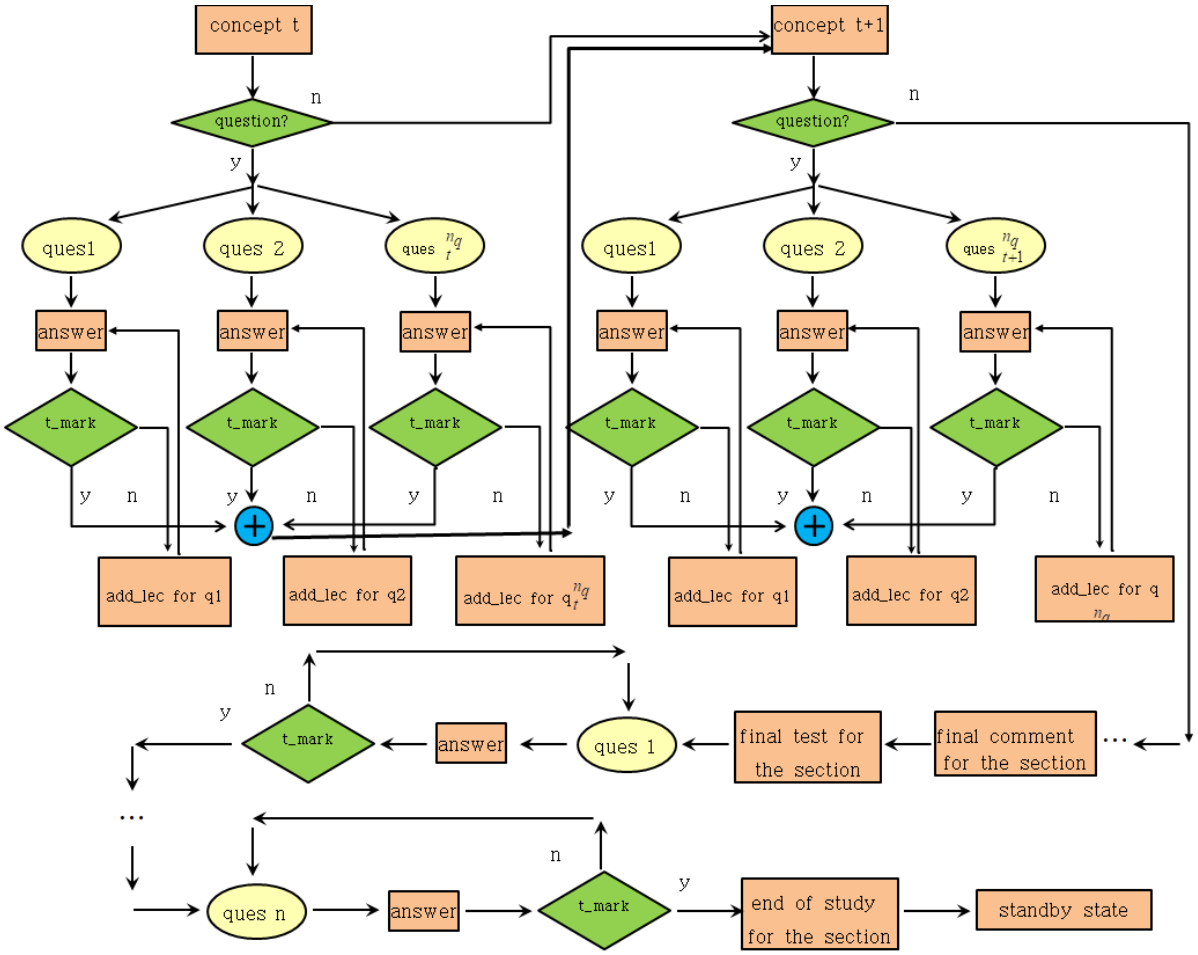

Figure 4. *Group Teaching Flow Diagram (Step by Step Flow Diagram).*

In the above figure, "question?" is to ask if there is a question, and "ques i" means "question 1", "t_mark" means a baseline score or a threshold score, and "add_lec for qi" means a supplementary lecture on the question qi.

4. Extended Knowledge Graph and Editor

4.1. Constructing an Extended Knowledge Graph

In our study, we define an extended knowledge graph and use it for knowledge representation. A knowledge graph is a kind of ontology, and it is a graph whose relation between nodes only reflects a forward-backward relation or a weight value or a semantic relation with a forward-backward relation. Many literatures represent nodes of knowledge graphs as vertices and arc of knowledge graphs as edges. We understand that the term node and edge represent the relationship between nodes as a two-dimensional view of the knowledge graph, and the term vertex and edge represents the relationship as a multidimensional view. In our future work, we use the terms node and vertex simultaneously to make a further description. A knowledge graph is defined as a weighted (or weightless) directed graph between knowledge concepts in terms of memory principles, in terms of the ordering of scientific content, or in terms of academic and pedagogical aspects. The knowledge concept is the central kernel of knowledge (or memory unit), and in intelligent tutoring systems, it includes slides, comment texts, corresponding video data, questions and answers to the concept. The vertex of the knowledge graph represents the knowledge concept, the edges represents the weighted order relation, i.e., the order relation with the relation strength value.

In our paper, we consider a knowledge graph whose relations between nodes have only a forward and backward order relation. In addition, many approaches have been proposed for the extension and decomposition of knowledge graphs, but our paper proposes an extended knowledge graph that is easily understandable by common teachers and applicable to their subjects.

Previous studies have defined many different types of knowledge graphs [17-19]. We define the knowledge graph as follows.

$G = (V, E)$, V is the set of vertices, E is the set of edges(relations), i.e., $V=\{v_i|\ i=1,2,...,n\}$ is the set of possible concepts. When $E=\{e_j|e_j=<v_{j1},v_{j2}>,\ v_{j1},v_{j2} \in V,\ j=1,2,\ldots,m\}$ is

a set of possible directed edges constructed by concept-association relations, the graph $G=(V, E)$ is called a learning path graph. A vertex in a knowledge graph reflects the knowledge concepts present in the subject and an edge the order relationships between those knowledge concepts. The knowledge graph must be acyclic. If we have to learn concept v_1 for two concepts v_1 and v_2, then we have to learn concept v_2, we denote it by the directed edge $e=<v_1, v_2>$.

We intend to extend and refine the knowledge graph for the construction of intelligent tutoring systems. We use the knowledge graph starting from the "subject graph" and extending it to the "hyper-concept graph", "basic concept graph", and "subconcept and help-concept graph". These graphs have an edge with a vertex, which indicates an educational or cognitive sequence. The definition of the updated knowledge graph is as follows.

4.1.1. Subject Graph

We define the subject graph as $G_{sub}=(V_{subi}, E_{subi})$. The index i is an integer index symbol, $i \in [1, N_{sub}]$, N_{sub} is the total number of vertices of the subject graph, V_{subi} represents one vertex of the subject graph, and E_{subi} represents one relation. A vertex of a subject graph defines the set of the largest concept blocks in the subject and the relationship defines the sequential relationship between the vertices. Figure 5 shows an example of a subject graph.

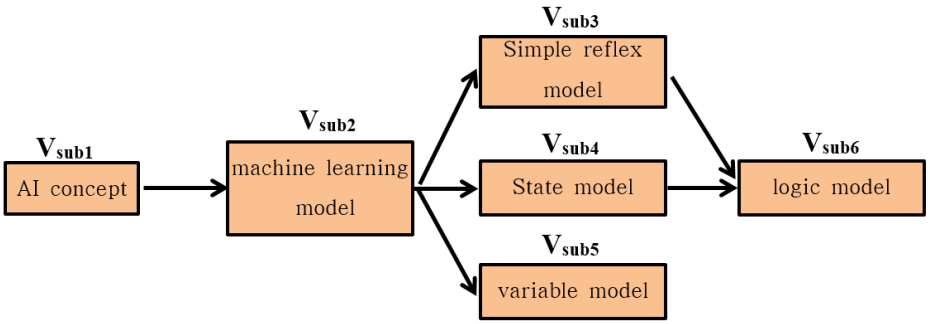

Figure 5. *Subject Graph Example.*

4.1.2. Hyper-concept Graphs

A hyper-concept graph is further constructed by decomposing the vertices of a subject graph, which is defined as follows:

$G_{sub} = (V_{subi}, E_{subi}) \approx G_{hyper} = (V_{hyperj}, E_{hyperj}) = (V_{subi}^{hyperl}, E_{subi}^{hyperl})$, $i \in [1, N_{sub}]$, $j \in [1, N_{hyper}]$, $l \in [1, N_{subi}]$ i, j, l are the total number of vertices of the hyperconcept graph, and N_{subi} is the number of child vertices belonging to the ith node of the s subject graph. The symbol ≈ indicates that the meaning of the two graphs is equivalent to each other. The following figures show the hyper-concept graph.

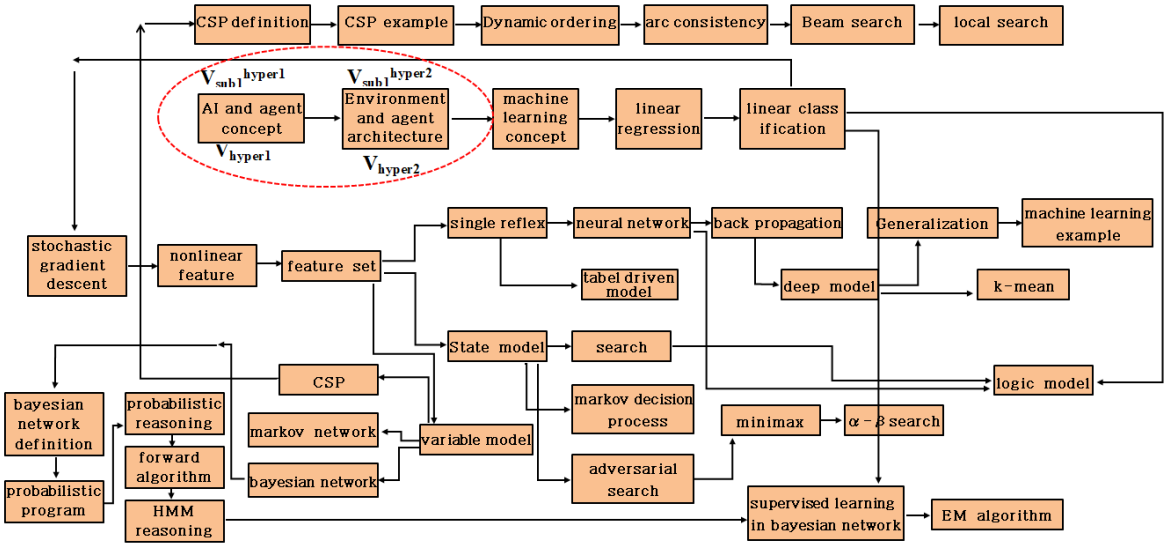

Figure 6. *Hyper-concept Graphs Example(1).*

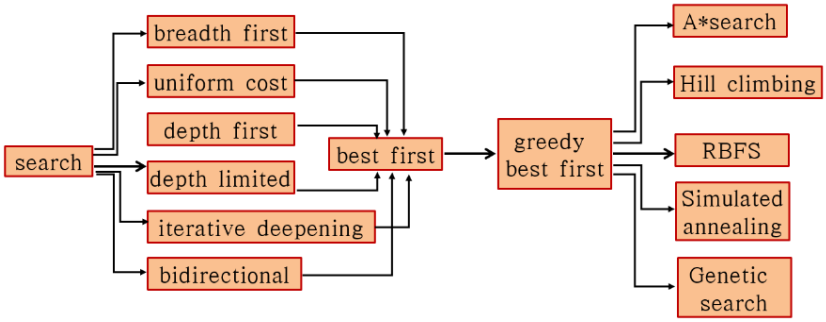

Figure 7. *Hyper-concept Graphs Example(2).*

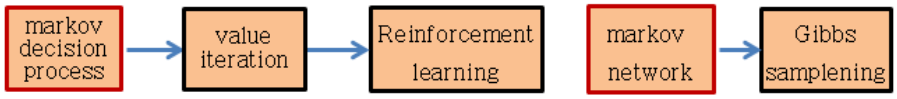

Figure 8. *Hyper-concept Graphs Example(3).*

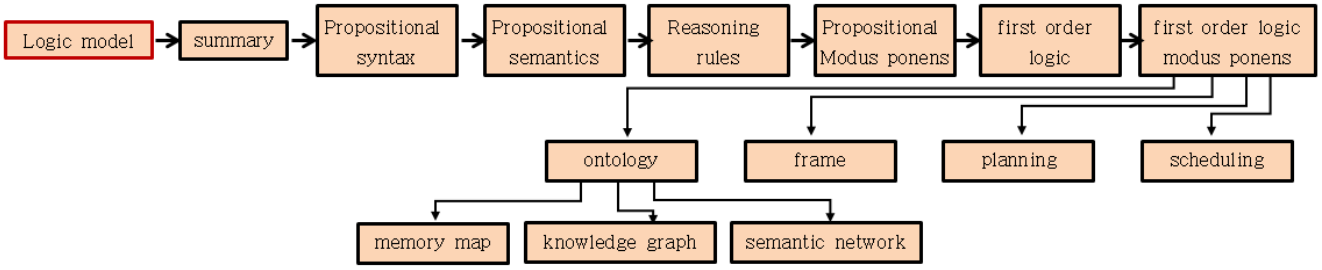

Figure 9. *Hyper-concept Graphs Example(4).*

4.1.3. Basic Concept Graphs

The basic concept graph is further decomposed into vertices of the hyper-concept graph, which is defined as follows.

$G_{sub} = (V_{subi}, E_{subi}) \approx G_{hyper} = (V_{hyperj}, E_{hyperj}) = (V_{subi}^{hyperl}, E_{subi}^{superl}) \approx G_{basic} = (V_{basick}, E_{basick}) = (V_{subi}^{superl,basicm}, E_{subi}^{superl,basicm})$, $i \in [1, N_{sub}]$, $j \in [1, N_{hyper}]$, $l \in [1, N_{subi}]$, $k \in [1, N_{basic}]$, $m \in [1, N_{subi,\ hyperl}]$

Let i, j, k, m be integers, N_{basic} be the total number of vertices of the basic concept graph, and $N_{subi,hyperl}$ are the number of children of vertices V_{subi}^{hyperl} of the hyper-concept graph. The basic teaching unit of tutoring is the basic concept. Eventually, in the intelligent tutoring system, all the instructional nodes of tutoring are composed of basic concepts. The list of contents displayed on the left-most part of the user interface of the developing intelligent tutoring system is the basic concepts in the subject, not the chapters and sections in the subject, as in the group teaching (Figure 11). Figure 10 shows an example of a basic concept graph corresponding to the nodes (V_{sub1}^{super1}) and (V_{sub1}^{super2}) (represented by a red circle) of the hyper-concept graph in Figure 6.

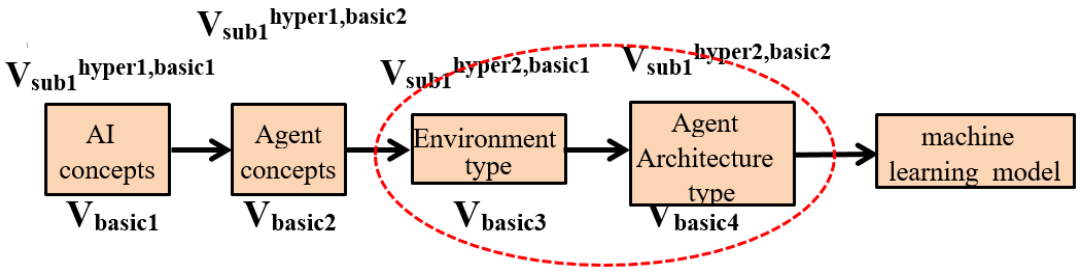

Figure 10. *Basic Concept Graph Example.*

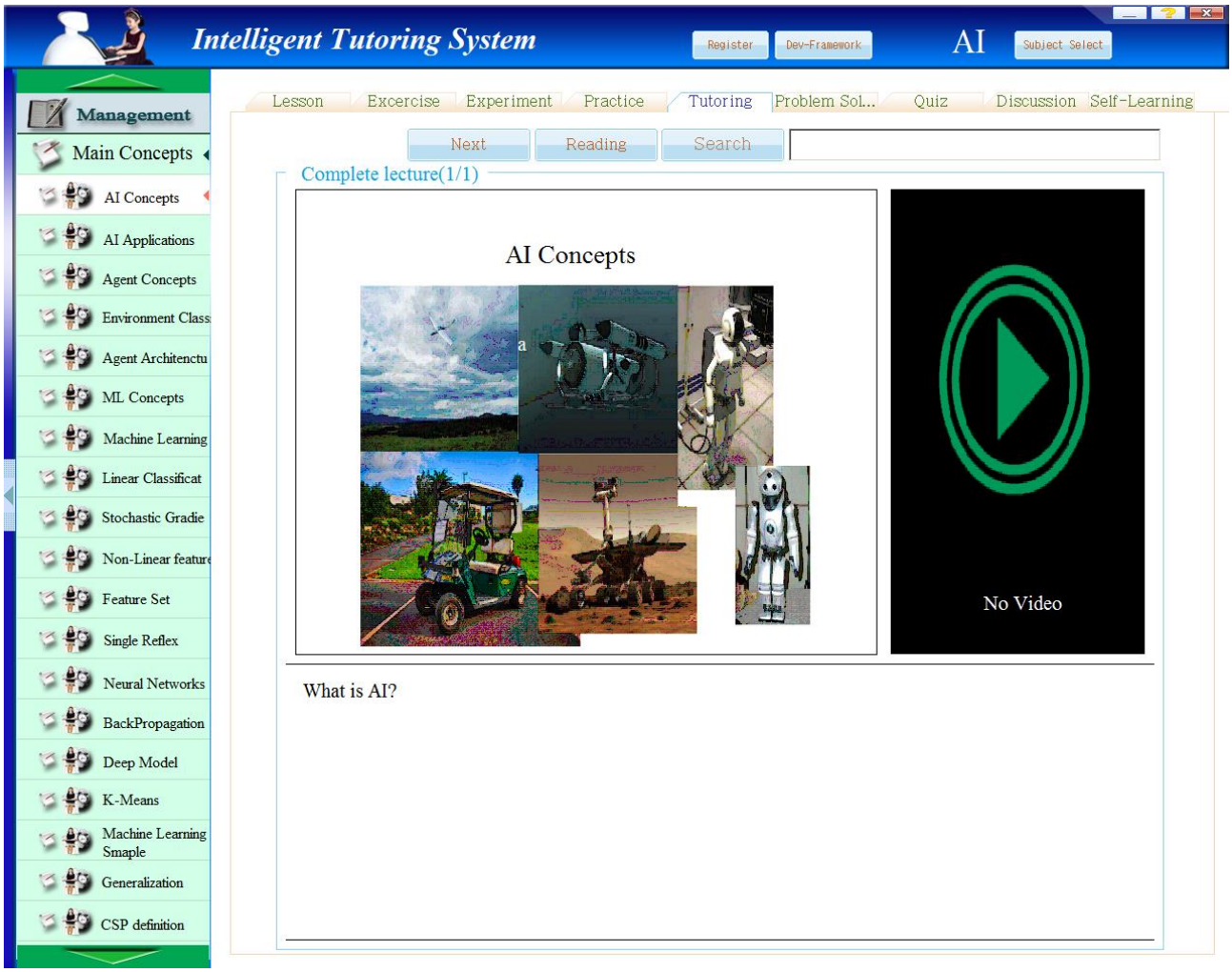

Figure 11. *In Tutoring System, the Choice is a Basic Concept.*

4.1.4. Subconcept and Help-concept Graphs

Figure 12 shows the sub-concepts and help-concepts between the basic concepts "Environmental Classification" and the basic concept "Active Structure Classification" in AI. This sub-concept and help-concept graph reflects the action between state and state, the result of inference of hidden Markov model, and the actual tutoring flow of intelligent teacher as tutor is implemented by this graph. The program uses this graph to generate a specific teaching flow for each student, i.e., an action that generates an appropriate teaching path for each student.

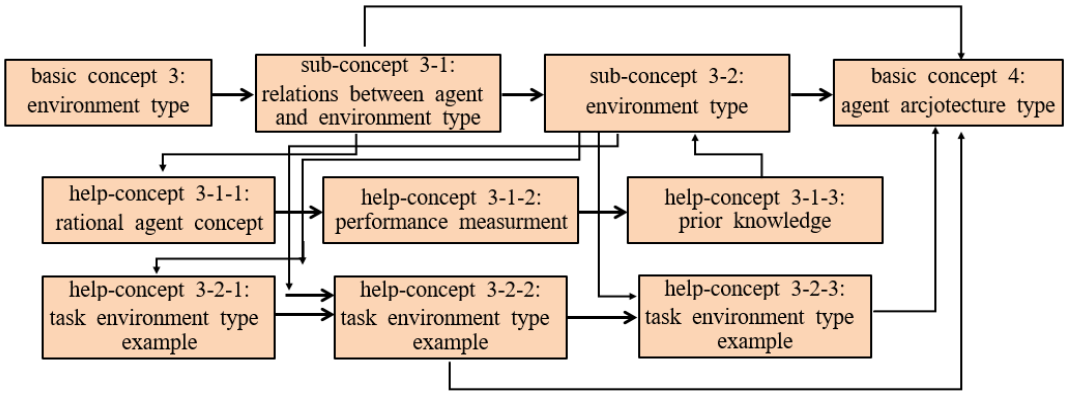

Figure 12. *Subconcept and Help-concept Graph Example.*

4.2. Editor Development

Our team developed an editor as an intelligent tutoring system development framework. The editor's mission is to use teacher's instructional resources to develop a static instructional model or to assist in generating a dynamic and adaptive instructional model. This approach is supported by the construction of a collective pedagogical model, the construction of knowledge graphs, the construction of Hidden Markov Models, and the construction of Markov Decision Processes for tutoring.

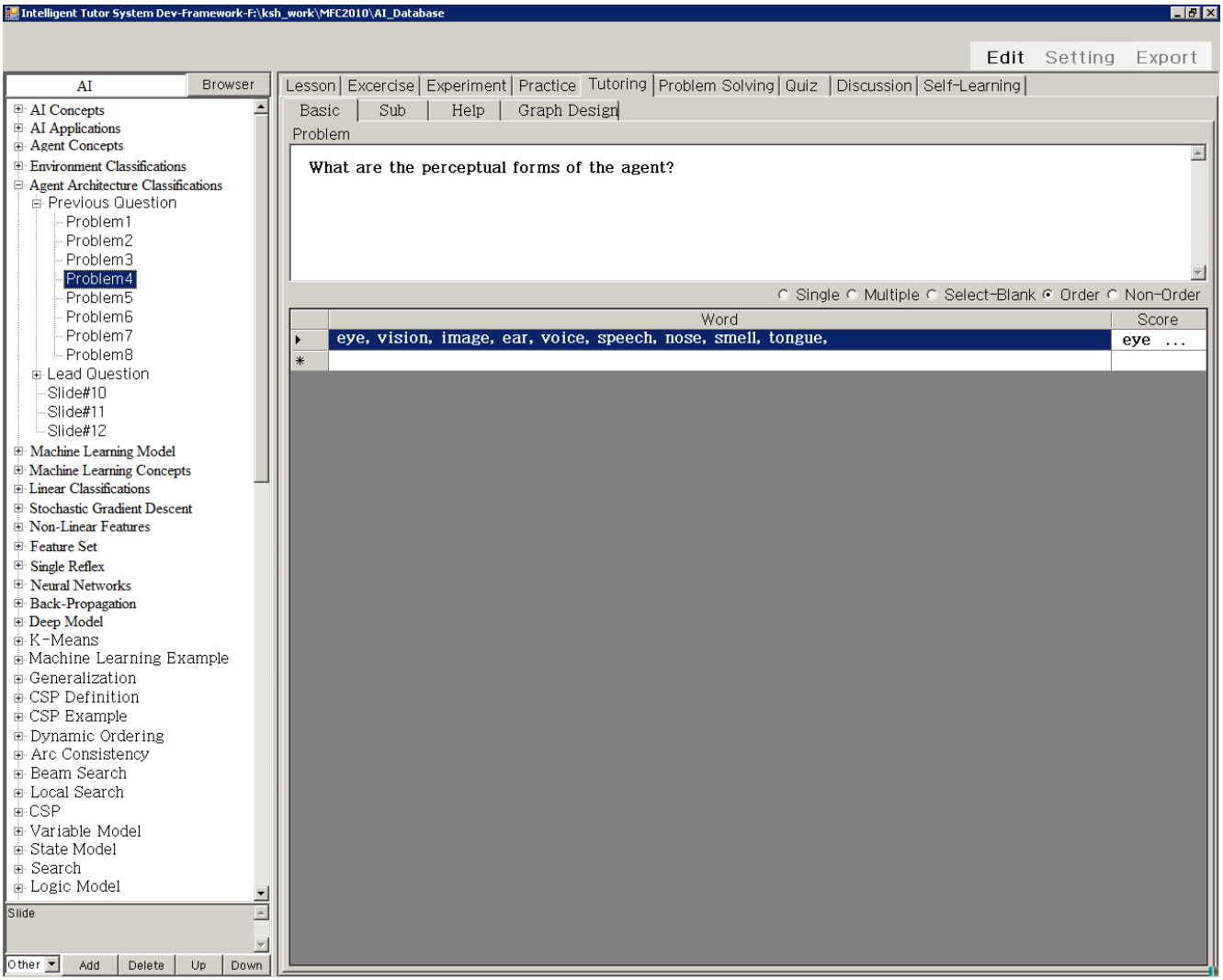

Figure 13. *Intelligent Tutoring System Editor.*

5. LSEPM and LPG Construction Method Using HMM

5.1. Formulation of Knowledge Graphs into Hidden Markov Models

5.1.1. Formulation of Hidden Markov Models

In the various types of knowledge graphs written by the subject teacher, the type of intelligent tutoring system using Hidden Markov Model is the basic concept graph. The Hidden Markov formalism for this basic concept graph is developed in Figure 14.

When learning state modeling is done in a probabilistic way, the learning state is represented as a random variable, which contains uncertainty, and is defined as a learning state variable. The Hidden Markov model is an extended dynamic Bayesian network, where nodes in this network consist of a sequence of learning state variables (learning state random variables). Eventually, each node of the basic concept graph must be converted into the learning state variables of the Hidden Markov model.

Learning state variables: $V_{subi}^{hyperl,\ basicm}$(or V_{basick})

(This variable is a random variable that reflects the student's current learning status, i.e., objective information about cognitive performance).

The state value of the learning state variable = {excellent(ex), very good(v_g), good(g), average(aver), failure(fail)}.

The state value of this learning state variable is the value that reflects the degree of cognition as the value of the student's learning state. Since the state values of the learning state variables cannot be determined directly and accurately, we instead define the observation variables and determine them indirectly and probabilistically through the observation values of the observation variables.

Observations of Observational Variables(Answer_Marks) = {1.0(interval 1), [0.8, 1) (interval 2), [0.6, 0.8) (interval 3), [0.4, 0.6) (interval 4), [0.0, 0.4) (interval 5) }

In the future, Answer_Marks is abbreviated as A_M, and V_{basick} is also denoted as V_k. The intervals in parentheses in the observation variable indicate the values of the observation variable. Figure 14 shows the Hidden Markov model corresponding to the basic concept graph, which is indicated in bold because the learning state variables and observation variables are vectors.

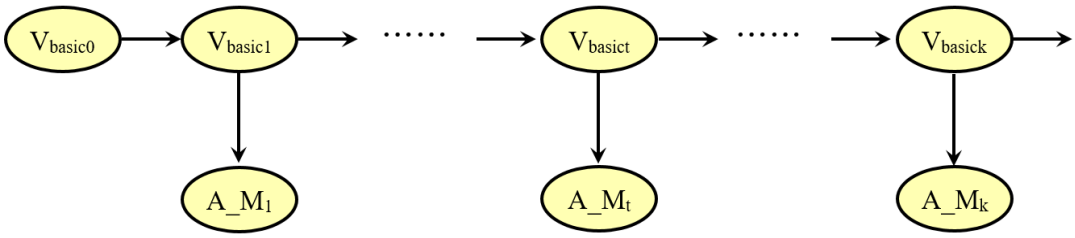

Figure 14. *Hidden Markov Models Corresponding to Basic Concept Graphs (Example 1).*

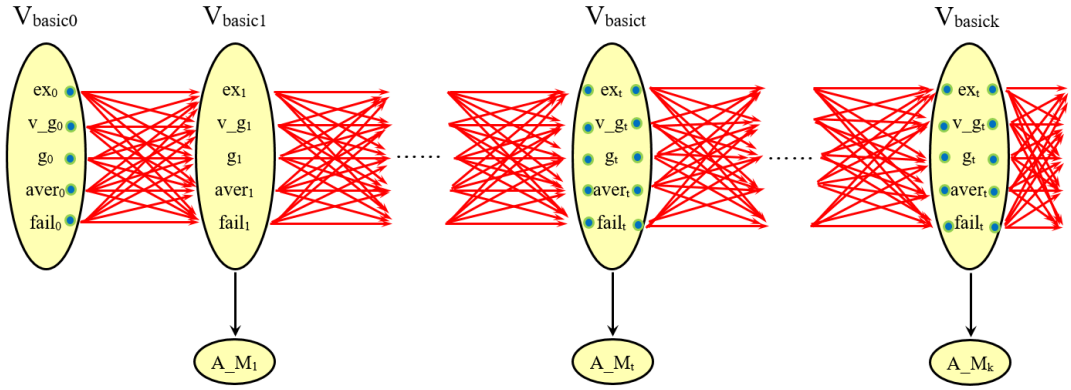

Figure 15. *Hidden Markov Models Corresponding to Basic Concept Graphs(Example 2).*

The transition model is shown in Table 2. Then ∑a:y=1.0. a:y means that in any row of the transition model table, each of the values from the first value a to the last value y is listed.

Table 2. *Transition Model.*

Vt-1	**P(Vt|Vt-1)**				
	ex	**v_g**	**g**	**aver**	**fail**
ex	0.85	0.1	0.02	0.02	0.01
v_g	0.2	0.4	0.2	0.1	0.1
g	0.3	0.2	0.3	0.15	0.05
aver	0.01	0.1	0.3	0.3	0.29

Vt-1	P(Vt|Vt-1)				
	ex	**v_g**	**g**	**aver**	**fail**
fail	0.0	0.0	0.2	0.3	0.5

The sensor model is a model that represents the probability that a certain value of the observed variable at time *t* will occur when the correct value of the student's learning state variable at time *t*-1 is already known. The sensor model is shown in Table 3. Also ∑a:y=1.0.

Table 3. *Sensor Model.*

V_{t-1}	$P(A\_M_t\|V_{t-1})$				
	Interval 1	**interval 2**	**interval 3**	**interval 4**	**interval 5**
ex	0.7	0.1	0.1	0.1	0.0
v_g	0.1	0.4	0.4	0.1	0.0
g	0.1	0.33	0.53	0.02	0.02
aver	0.0	0.1	0.39	0.33	0.18
fail	0.0	0.0	0.12	0.43	0.45

If the observed value at time t is judged as interval 2, then the observation matrix O_t as follows.

$$O_t = \begin{pmatrix} 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0.4 & 0 & 0 & 0 \\ 0 & 0 & 0.33 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0.0 \end{pmatrix}$$

5.1.2. Collection of Values of Evidence Variables

(a) Value-gathering algorithm of evidence variable

The learning state variable of the Hidden Markov model reflects the learning state of the student, which cannot be directly identified. If we can identify directly, we are not the subject of the Hidden Markov model. The state identification is confirmed by the time interval probability inference using the Hidden Markov model after indirect evidence is obtained from the answers to the students' perceptions of the concept. The algorithm for collecting the value of the evidence variable is implemented in four steps as follows.

[Algorithm 2. Value Collection Algorithm of Evidence Variables]

Step 1: Presently, elicit query presentation that enables the student to derive the basic concept of time t by himself. If the answer score for the question is 10, then move to step 3, otherwise move to step 2.

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 complete lecture on the basic concept of time t (the intelligent teacher rehearses the concept comprehensively, assuming that the student has derived the basic concept of his own but is not yet fully understood).

Step 4: Give a question to understand the basic concept of time t.

(b) Method of Values Collection of Evidence Variables.

We fix the proof of time t by setting the result of step 4 of Algorithm 2 as follows. In Step 4 of Algorithm 2, let us assume that the student's perception of the basic concept of time t is TD.

If the score TD = 1.0, we call the <A_M= interval 1>, if TD = [0.8,1], then <A_M=interval 2>, if TD = [0.6,0.8), then <A_M=interval 3>, if TD = [0.4,0.6], then <A_M=interval 4>, if TD = [0.4], then<A_M=interval 5>.

(c) Example of collecting values of evidence variables.

/*--Guided questions and supplementary lecture on basic concept 3------*/

(Basic concept 3) Relationships between agent and envi-

ronment, classification of environment.

[Problem 3-1, 1, Audio] When we look at the concept definition of an agent, is there no connection between the environment and the agent?

ⅰ. connection, ⅱ. no connection

[Answer] ⅰ. 10 ⅱ. 0

(Supplementary lecture_slide 3-1) Slide1.png.

(Comment 3-1) There is a connection. Agents that do not receive percept data from the environment and do not act on the environment is little need to develop.

(Video 3-1)

[Problem 3-2,4, Audio] What is the relationship between an agent and the environment? Try to write your ideas.

Correct: There is a percept relationship between an agents and an environment that uses a sensor to sense information, and then, by making appropriate reasoning, acts on his environment and affects it.

[Answer] ⅰ Percept ⅱ Action

(Supplementary lecture_slide 3-2) Slide1.png.

(Comment 3.2) There is a percept relationship between an agent and an environment that is sensed by the use of sensors, and then acts in its environment by making appropriate reasoning.

(Video 3-2)

/*----------A complete lecture on basic concept 3----------*/

(slide 3-8) Slide1.png.

(Comment 3-8) Why should we deal with the environment in the design of an actor?

(Video 3-8)

(slide 3-9) Slide2.png.

(Comment 3-9) The table is shown in the figure.

(Video 3-9)

/*-------A question to understand the perception of Concept 3----*/

[Problem 1,1] What is an environment? Choose the right ones.

ⅰ. Environment? Where an agent resides. ⅱ. Instead of the term environment, we also use the term world. ⅲ. The environment is assumed to change by an agent.

[Answer] ⅰ 3 ⅱ 4 ⅲ 3

[Problem 2, 1, Audio] When we look at the concept definition of an agent, is there no connection between the environment and the agent?

ⅰ. connection ⅱ. no connection

[Answer] ⅰ 10 ⅱ 0

5.1.3. Limitations of the Hidden Markov Model

(a) Evaluation(correctly estimated) of the learning state sequence and prediction of the next-time learning state only.

In intelligent tutoring systems, Hidden Markov models are used to evaluate current learning situations, predict future learning situations, and estimate performance in previous learning. That is, the Hidden Markov model can estimate a student's learning sequence of states through the evidence values from the past to the current time, but it cannot directly generate the best learning sequences to achieve the overall goal of learning. These learning sequences are the learning processes, and the learning flows inherent to each student that can maximize the degree of student awareness.

There are sequential decision-making methods, Markov decision-making or partially observable Markov decision-making, that are methods to calculate a sequence of actions that maximizes the learning state of each student, i.e., generate a learning flow that is specific to each student.

(b) Learning path generation principle using hidden Markov model.

The Hidden Markov model can probabilistically derive estimates and predictions of current student's learning state, future state. Using this value, we have to develop a specific learning path generation algorithm separately.

5.2. Building a Tutorial Database

The tutoring database is as follows.

Table 4. *Tutorial Database.*

record name	data type	main key	explanation
Concept table			
ID	INTEGER	√	unique identifier number
ConceptName	TEXT		concept name
ConceptType	INTEGER		concept type: basic concept-1, sub-concept-2, help-concept-3
RootConcept	INTEGER		root basic concept ID: this value is all zero, otherwise, the unique number of the root basic concept of the concept. For example, if a subconcept B is a subconcept between the basic concept A and the basic concept C, then the ID value of the basic concept A is the value of this field.
basic concept relational table			

record name	data type	main key	explanation
Each row of the table represents each edge(relation) of the concept graph.			
ID	INTEGER	√	unique identifier number
BeginVertex	INTEGER		precedence ID
EndVertex	INTEGER		ID of the posterior concept
RootConcept	INTEGER		fundamental concept of the relation
Relationship	INTEGER		What are the relationships between concepts. If either the front or rear nodes is a subconcept, then 2. if one of the two is the basic concept, and one is the subconcept, then 1. If both concepts are basic, then 0
Lecture table			
Comment: The slide are represented as a line of this table and are separated by the values of the ConceptID field. That is, to get the slide information for concept A, we get the unique identifier (ID) of the row corresponding to concept A in the concept table and return only the rows that are equal in value and ConceptID.			
ID	INTEGER	√	unique identifier of slide
ConceptID	INTEGER		Unique number of concepts belonging to slide is it an slide that reflects which concept.
SlideName	TEXT		The name of the picture file (including extension) in the path of "the subject name/ individual teaching/slide".
Comment	TEXT		comment
VideoName	TEXT		The name of the movie file (including extension) in the path of "the subject name/ individual teaching/slide".
ShowOrder	INTEGER		Numerical values for slide display sequences
hasTest	INTEGER		If there is a question for slide, then 1, 0, if no.
ReTest			
ID	INTEGER	√	unique identifier
XMLData	TEXT		XML data for testing
ParentConcept	INTEGER		The ID of the parent concept, which concept is the question of understanding.
PreTest			
ID	INTEGER	√	unique identifier
XMLData	TEXT		XML data for testing
ParentConcept	INTEGER		In the Concept Table, the ID of the concept, i.e., is it an leading question that reflects which concept
SlideName	TEXT		The name of the picture file (including extension) in the path of "the subject name/ individual teaching/slide".
Comment	TEXT		comment
VideoName	TEXT		The name of the movie file (including extension) in the path of "the subject name/ individual teaching/slide".
SlideTest (A understanding rate for a slide)			
ID	INTEGER	√	unique identifier
XMLData	TEXT		XML data for testing
ParentSlide	INTEGER		The ID of the slide, i.e., is it an leading question that reflects which slide.

5.3. Learning State Modeling Using Smoothing Algorithm

Learning state modeling using Hidden Markov models involves tasks such as filtering, state estimation, prediction, smoothing, generation of most likely explanation, and learning, and we set only the prediction of the learning state for student model generation for research and development purposes.

5.3.1. Learning State Prediction Method

Prediction using the Hidden Markov model may be considered as a filtering process without the presence of new evidence. The filtering process already implements a one-step prediction process and can easily derive recursive prediction computations for the state at t+k+1 from the prediction for t+k.

We first consider the filtering process and then consider the prediction method using this result. Then, the method proposed in Russell et al. [14] is applied to our learning state prediction process and an example of computation is presented.

(a) Filtering

Given the filtered result at time t, we have to calculate(estimate) the result for time t+1 from the new evidence A_M_{t+1}. This is equivalent to computing the following function f:

$$P(V_{basic\,t+1} \mid A\_M_{1:t+1}) = f(A\_M_{t+1}, P(V_{basic\,t} \mid A\_M_{1:t}))$$

Here $V_{basic\,t+1}$ is the learning state variable at time t+1, and $A\_M_{1:t+1}$ is the observation data at time t+1.

This recursive estimation calculation can be divided into two steps. First, we represent the distribution of the current state forward from t to t+1, and then update it using the new evidence A_M_{t+1}.

$$P(V_{basic\,t+1} \mid A\_M_{1:t+1}) = P(V_{basic\,t+1} \mid A\_M_{1:t}, A\_M_{t+1}) \text{ (separating evidence)} = \alpha\, P(A\_M_{t+1} \mid V_{basic\,t+1}, A\_M_{1:t})\, P(V_{basic\,t+1} \mid A\_M_{1:t}) \text{ (Using Bayesian Rules Given an Evidence Value } A\_M_{1:t}) = \alpha\, P(A\_M_{t+1} \mid V_{basic\,t+1})\, P(V_{basic\,t+1} \mid A\_M_{1:t}) \text{ (By the Sensor Markov assumption) (1)}$$

In the above equation, the first term is the sensor model and the second term is the prediction. α is a normalization constant that makes the sum of probability values equal to unity. Let us further develop Eq. (1).

$$P(V_{basic\,t+1} \mid A\_M_{1:t+1}) = \alpha\, P(A\_M_{t+1} \mid V_{basic\,t+1}) \sum_{V_{basic\,t}} P(V_{basic\,t+1} \mid V_{basic\,t}, A\_M_{1:t}) P(V_{basic\,t} \mid A\_M_{1:t}) = \alpha P(A\_M_{t+1} \mid V_{basic\,t+1})$$

$$\sum_{V_{basic\,t}} P(V_{basic\,t+1} \mid V_{basic\,t}) P(V_{basic\,t} \mid A\_M_{1:t}) = \text{(Markov assumption) (2)}$$

In the above equation, the first term on the left is the sensor model, the first term on the left in the additive term is the transition model, and the last term is the recursive term.

Also, all terms are model or previous state estimates. Here, we consider the filtered estimate $P(V_{basic\,t} \mid A\_M_{1:t})$ as a "message" $f_{1:t}$, corrected by each forward-propagating transition model in the sequence and updated by each new observation. This process is formalized as follows.

$$f\_m_{1:t+1} = \text{FORWARD}(f\_m_{1:t}, A\_M_{1:t+1})$$

$$f\_m_{1:t+1} = \alpha O_{t+1} T^{T} f\_m_{1:t} \quad (3)$$

Here, FORWARD implements the update process described in Eq. (2), which starts with $f\_m_{1:0} = P(V_{basic\ 0})$. When all state variables are discrete, each update time is constant (i.e., dependent on t), and the required space cost is also constant. These constants, of course, depend on the size of the state space and the type of the corresponding specific time-interval model.

Let us consider only two steps of filtering in an intelligent tutoring system. Let us compute $P(V_{basic2} \mid A\_M_{1:2})$. At this time, refer to the values in Tables 2 and 3.

(b) Computational examples of filtering processes

In Figure 15, there is no evidence for the 0th learning state variable, and only the prior confidence of the observer exists.

$P(V_{baisc0})$ = <0.1, 0.3, 0.42, 0.1, 0.08>

The evidence value of the first learning state variable is 0.7, with A_M = interval 3. The state estimation from 0 to 1 is performed as follows.

$$P(V_{basic1}) = \sum_{V_{basic0}} P(V_{basic1} \mid V_{basic0}) P(V_{baisc0}) = \text{<0.85, 0.1, 0.02, 0.02, 0.01>} \times 0.1 + \text{<0.2, 0.4, 0.2, 0.1, 0.1>} \times 0.3 + \text{<0.3, 0.2, 0.3, 0.15, 0.05>} \times 0.42 + \text{<0.01, 0.1, 0.3, 0.3, 0.29>} \times 0.1 + \text{<0.0, 0.0, 0.2, 0.3, 0.5>} \times 0.08 = \text{<0.272, 0.224,0.234,0.149, 0.121>}$$

$$P(V_{basic1} \mid A\_M_1) = \alpha\, P(A\_M_1 \mid V_{basic1}) P(V_{basic1}) = \alpha\, \text{<0.1, 0.4, 0.53, 0.39, 0.12> <0.272, 0.224,0.234,0.149, 0.121>} = \alpha\, \text{<0.0272, 0.0896, 0.12402, 0.05811, 0.01452>} \approx \text{<0.0868, 0.2859, 0.3957, 0.1854, 0.0462>}$$

The evidence corresponding to the second learning state variable is 0.5, with the A_M_2 = interval 4. The state estimation from 1 to 2 is performed as follows.

$$P(V_{basic2} \mid A\_M_1) = \sum_{V_{basic1}} \mathbf{P}(V_{basic2} \mid V_{basic1}) P(V_{basic1} \mid A\_M_1)$$

$$= \text{<0.85, 0.1, 0.02, 0.02, 0.01>} \times 0.0868 + \text{<0.2, 0.4, 0.2, 0.1, 0.1>} \times 0.2859 + \text{<0.3, 0.2, 0.3, 0.15, 0.05>} \times 0.3957 + \text{<0.01,}$$

0.1, 0.3, 0.3, 0.29>×0.1854 + <0.0, 0.0, 0.2, 0.3, 0.5>×0.0462 = <0.2515, 0.2207, 0.2425,0.1592, 0.1261>

The estimate for the current learning state is g.

$P(V_{basic2} \mid A\_M_1, A\_M_2) = \alpha\ P(A\_M_2 \mid V_{basic2})P(V_{basic2} \mid A\_M_1) = \alpha$<0.1, 0.1, 0.02, 0.33, 0.43><0.2515, 0.2207, 0.2425,0.1592, 0.1261> = α <0.02515, 0.02207, 0.00485, 0.05254, 0.05422> ≈ <0.1583, 0.1390, 0.0305, 0.3308, 0.3414>

When the second learning phase was followed, the student's cognitive state became worse and worse, and became aver.

(c) Prediction

Prediction is a problem that predicts what the student's cognitive state value will be when passing through the third learning state when it comes to the second learning state. The evidence is given until the second lecture. The prediction task may be considered as a filtering process without the presence of new evidence. The filtering process already implements a one-step prediction process and can easily derive recursive prediction computations for the state at time $t+k+1$ from the prediction for $t+k$.

$$P(V_{basic\ t+k+1} \mid A\_M_{1:t}) = \sum_{\mathbf{V}_{basic t+k}} P(\underbrace{\mathbf{V}_{basic\ t+k+1} \mid \mathbf{V}_{basic\ t+k}}_{\text{transition model}}) \underbrace{P(V_{basic\ t+k} \mid A\_M_{1:t})}_{\text{recursive}} \quad (4)$$

In this calculation, only the transition model is included and the sensor model is not included.

5.3.2. Smoothing algorithm

The filtering and prediction in the Hidden Markov model proposed above are implemented by a smoothing algorithm.

[algorithm 3. smoothing algorithm]

function SMOOTHING(A_M_t, *hmm*, *d*) returns a distribution over $V_{basic\ t-d}$

inputs: A_M_t, // the current evidence for time step *t*

hmm, // a hidden Markov model with S× S transition matrix T

d, // the length of the lag for smoothing

persistent: *t*, // the current time, initially 1

f_m, // the forward message $P(V_{basic\ t} \mid A\_M_t)$, initially *hmm*.PRIOR

B, // the *d*-step backward transformation matrix, initially the identity matrix*d*

$A\_M_{t-d:t}$, // double-ended list of evidence from $t - d$ to *t*, initially empty

local variables: O_{t-d}, O_t, diagonal matrices containing the sensor model information

add A_M_t to the end of $A\_M_{t-d:t}$

$O_t \leftarrow$ diagonal matrix containing $P(A\_M_t \mid V_{basic\ t})$

if $t > d$ then

f_m$\leftarrow$ FORWARD(f, A_M_{t-d})

remove A_M_{t-d-1} from the beginning of $A\_M_{t-d:t}$

$O_{t-d} \leftarrow$ diagonal matrix containing $P(A\_M_{t-d} \mid V_{basic\ t-d})$

$B \leftarrow O_{t-d}^{-1} T^{-1} B T O_t$

else $B \leftarrow BTO_t$

$t \leftarrow t + 1$

if $t > d + 1$ then return NORMALIZE(f_m × B1) else return null

5.4. Learning Path Generation

5.4.1. Learning Path Generation Algorithm

[algorithm 4. Learning path generation algorithm]

1: if ($V_{basick+1}$ = ex)

2: {

3: (If the student's learning state prediction result is ex, then the derived query for the k+1th concept is repeated until the score is 1.0, then a rich and complete lecture on the k+1th concept is given);

4: }

5: else if($V_{basick+1}$ = v_g)

6: {

7: (Choose the shortest path among paths consisting of the sub-concepts between the kth concept and the k+1th concept);

8: }

9: else if(V_{basick} = g)

10: {

11: (choose a path among m-1 paths consisting of sub-concepts between kth concept and k+1th concept using probabilistic inference algorithm);

12: }

13: else if(V_{basick} = aver)

14: {

15: For a student with this value, we generate paths that include the subconcept and help-concept, and then also divide the number of total sub path as above and generate the learning path according to the corresponding probability value.

16: }

17: else

18: {

19: Make a choice of the k-1th basic concept.

20: }

5.4.2. Description of Line 12 of Algorithm 4

A description of the stochastic path selection method for intervals 3. For the A_M=interval 3(0.6<= TD < 0.8), we randomly choose a path among m-1 paths consisting of the sub-concepts between the nth concept and the n+1th concept. If the distance is between 0.6 and 0.8, divide 1.0 by (n = total sub-learning path m −1) and find the ratio value that one

learning path occupies in the whole interval 1. For example, if we have five sub-learning paths, then dividing 1.0 by 4 yields a value of 0.25. This means that each learning path (n = total sub-learning path m-1) has a ratio of 0.25 in the interval (actually, the interval between 0.6 and 0.8). Then, n sub-learning paths are ordered from one to the next, depending on the number of arcs present in each path, from the number of arcs to the least. The probability values of each of the four intervals are expressed as

$0.6+0.25\times0.2\times1=0.6+0.05=0.65$,
$0.6+0.25\times0.2\times2=0.6+0.1=0.7$,
$0.6+0.25\times0.2\times3=0.6+0.15=0.75$,
$0.6+0.25\times0.2\times4=0.6+0.2=0.8$.

The value of 0.25 is the ratio of one path to the path consisting of n (m-1) subconcepts. This number varies with the number of paths that consist of subconcepts. The value of 0.2 is the value that makes the interval between 0.6 and 0.8 equal to 1.

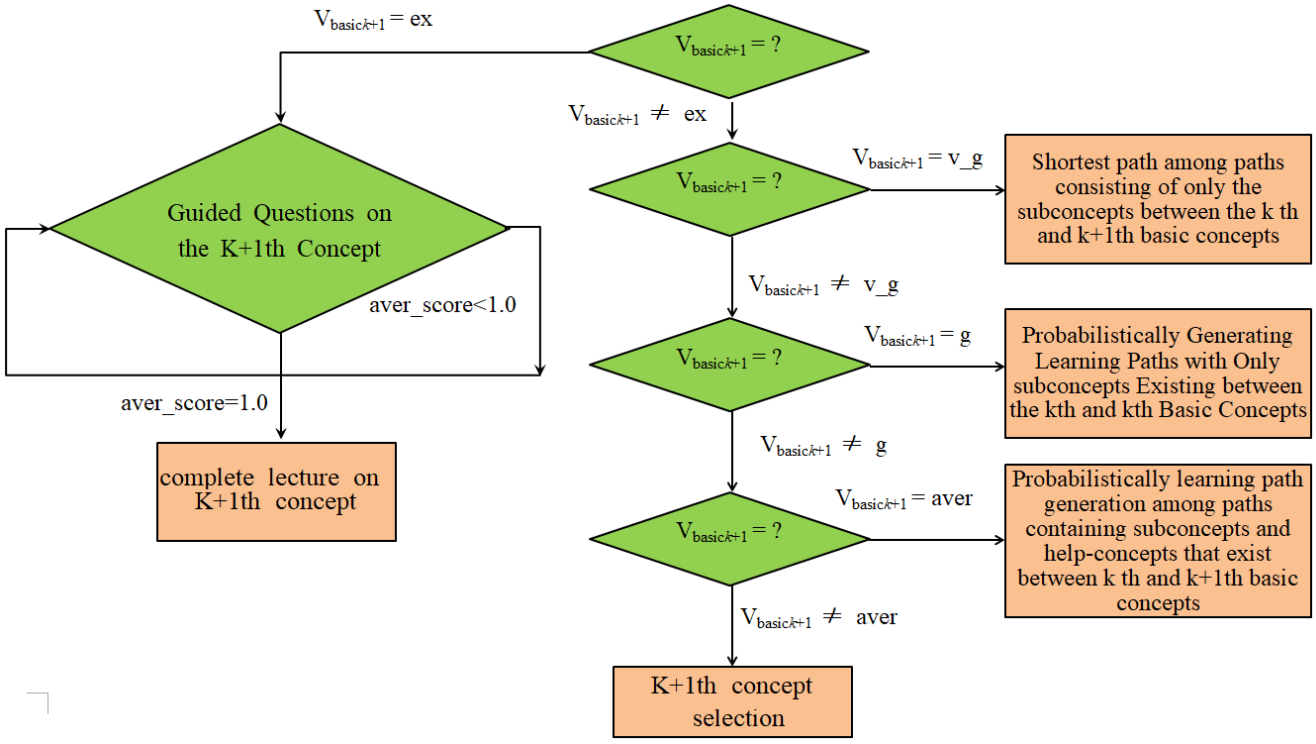

Figure 16. *Learning Path Generation Flow Diagram.*

6. 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.

6.1. Formalization of Markov Decision Process Model

6.1.1. General Concept

1) Maximum expected utility value principle.

The maximum expected utility principle is described in Russell et al. [14] as follows. An agent that must make a decision has actions a and state 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. $P(\text{RESULT}(a) = s')$ 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.

$$EU(a) == \sum_{s'} P(RESULT(a) = s')U(s') \quad (5)$$

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*' | *s*, *a*), and a reward function *R*(*s*, *a*, *s*')([14]).

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 Russell et al. [14].

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:

$$Q_t(b,a) = \sum_s B_t(s) Q(s,a) \quad (6)$$

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*.

6.1.2. Formalization- 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 LS_t consists of a set of slides(S, Slides), comments(Co, Comments), videos(V, Videos), questions and answers(QA, Questions and Answers).

$LS_t = <S_t, Co_t, V_t, QA_t>$, where t is the specific time.

LS_t=<$Slide_t$($Slide_t^1$, $Slide_t^2$,...,$Slide_t^{ns}$), Co_t(Co_t^1, Co_t^2,..., Co_t^{ne}), V_t(V_t^1, V_t^2,..., V_t^{nv}), QA_t(QA_t^1: QA_t^1, QA_t^2: QA_t^2,..., QA_t^1: QA_t^{nq})>

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 [18, 17]. 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 C_{basict} 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 LSV_t = {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 17.

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 $P(s' \mid s, a)$, 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 5 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 5. *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 5. 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 6. 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 7. 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 8. 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 17 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, B_t indicates the basic concept state variable(abbreviated Ba), Sb_t indicates the subconcept state variable(abbreviated Sub), A_t indicates the subconcept state variable(abbreviated Aux), and S_t 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.

$Up_{3,t}^{1}$ is an Up_3 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 B_t as the initial state in the network of Figure 17, and completes its actions in the B_{t+1} state. Each performance is considered probabilistic, determined through experiments, empirically or through reinforcement learning. Up_1, Up_2, Up_3, $Down_1$, $Down_2$, $Down_3$, 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. P_{t-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 $Down_{1,\,t-1}^{3}$ on the S_t state variable, the probability of reaching a state $P_{t,1}$ 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 $Down_{1,\,t-1}^{3}$ are {0.91, 0.03, 0.03}.

Table 6. *Transition model.*

S_{t-1}	$P(S^1_t\|S_{t-1}) = 0.91$			$P(S^2_t\|S_{t-1})=0.03$			$P(S^3_t\|S_{t-1})=0.03$			$P(S^4_t\|S_{t-1})=0.03$		
	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 B_0, B_1, …, B_t, …, B_n, where the value of initial state B_t is H and the value of target state B_{t+1} is H. The sequence of actions that the intelligent tutoring system performs is as follows: In the initial state B_t with value H, the intelligent tutoring reaches state $Sb_{t,1}$ with value M by executing $Down_{1,\,t-1}^{3}$, and then performing $Down_{2,t}^{1}$ reaches state $S_{t,2}$ with value L, and reaches the target state B_{t+1} with value H by executing $U_{p3,t}^{2}$.

$$B_t(H) \xrightarrow{Down_{1,t-1}^{3}} Sb_{t,1}(M) \xrightarrow{Down_{2,t}^{1}} S_{t,2}(L) \xrightarrow{Up_{3,t}^{2}} B_{t+1}(H) \quad (7)$$

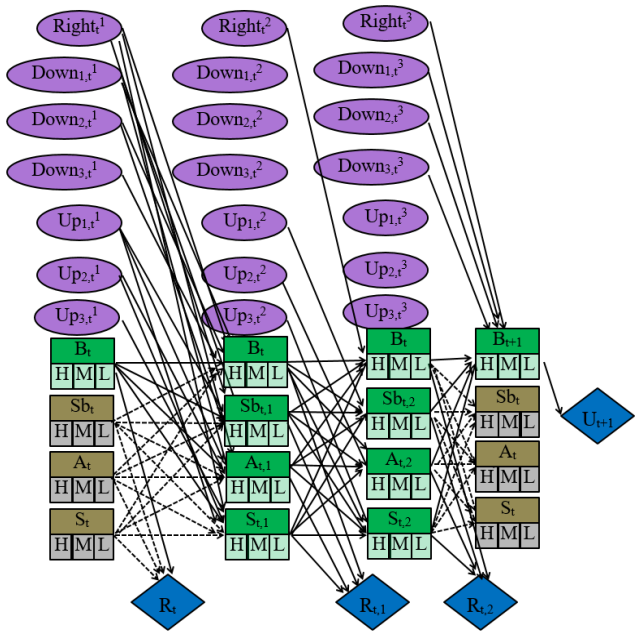

Figure 17. *Dynamic Decision Networks for Intelligent Tutoring Systems.*

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 R_t and the utility function value by U_t.

6.2. Computational Methods

6.2.1 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 $0.91^3 = 0.7536$, 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.

6.2.2. 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 $U_h([s_0,a_0,s_1,a_1,\ldots,s_n])$.

Then, we have to establish how to calculate the utility value for the state sequences. We use additive discounted rewards method [14]. The utility value for a certain history is calculated as follows.

$$U_h([s_0,a_0,s_1,a_1,s_2,\ldots]) = R(s_0,a_1,s_1)+\gamma R(s_1,a_2,s_2)+\gamma^2 R(s_2,a_3,s_3)+\ldots$$

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.

6.2.3. 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 policies 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(B_t), subconcept state variables(Sb_t), auxiliary concept state variable(A_t), supplementary explanation state variable(S_t) 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 s_t. When the initial state is $S_0 = s_t$, the probability distribution of the state sequences (S_1, S_2,..) 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 [14], the following action selection method using expected utility is described. The expected utility value ob-

tained when executing the policy π starting from the basic concept state variable B_t is. The state notation uses s, the general notation of the state variable.

$$\mathrm{U}^{\pi}(S_t) = E\left[\sum_{t=0}^{\infty}\gamma^t \mathrm{R}(S_t, \pi(S_t), S_{t+1})\right] \tag{8}$$

The expected value E depends on the probability distribution for the state sequences determined by S_t and π. Starting at the initial state S_t, 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 $\pi^*_{S_t}$.

$$\pi^*_{S_t} = \arg\max_{\pi} U^{\pi}(S_t) \tag{9}$$

$\pi^*_{S_t}$ also recommends some action for every state as a policy, especially if S_t 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 S_t 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 $U(S_t)$.

This utility function $U(S_t)$ 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.

$$\pi^*_{S_t} = \arg\max_{a\in A(S_t)} \sum_{S'_t} P(S'_t \mid S_t, a)[R(S_t, a, S'_t)] \tag{10}$$

b) Method of calculation of utility function using Bellman equation

As above the utility function $U(S_t)$ 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

$$U(s) = \max_{a\in A(s)} \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma U(s')] \tag{11}$$

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. (10) is a solution to the set of Bellman equations. In fact, the solutions are unique. In Figure 17, 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 $Down^3_{1,t-1}$, $Down^3_{2,t-1}$, $Down^3_{3,t-1}$, and $Right^3_{t-1}$.

Max{ [0.91 ×0.85 ×(-0.1+γU($Sb_{t,1}$(H)))+0.91 ×0.1 ×(-0.2+γU($Sb_{t,1}$(M)))+0.91 ×0.05 ×(-0.2+γU($Sb_{t,1}$(L)))+
(0.03 ×0.85 ×(-0.1+γU(B_t(H)))+0.03 ×0.1 ×(-0.2+γU(B_t(M)))+0.03 ×0.05 ×(-0.3+γU(B_t(L))))+
(0.03 ×0.85 ×(-0.1+γU($S_{t,1}$(H)))+0.03 ×0.1 ×(-0.2+γU($S_{t,1}$(M)))+0.03 ×0.05 ×(-0.3+γU($S_{t,1}$(L))))+
(0.03 ×0.85 ×(-0.1+γU($A_{t,1}$(H)))+0.03 ×0.1 ×(-0.2+γU($A_{t,1}$(M)))+0.03 ×0.05 ×(-0.3+γU($A_{t,1}$(L))))],
[0.91 ×0.85 ×(-0.1+γU($S_{t,1}$(H)))+0.91 ×0.1 ×(-0.2+γU($S_{t,1}$(M)))+0.91 ×0.05 ×(-0.2+γU($S_{t,1}$(L)))+
(0.03 ×0.85 ×(-0.1+γU(B_t(H)))+0.03 ×0.1 ×(-0.2+γU(B_t(M)))+0.03 ×0.05 ×(-0.3+γU(B_t(L))))+
(0.03 ×0.85 ×(-0.1+γU($S_{t,1}$(H)))+0.03 ×0.1 ×(-0.2+γU($S_{t,1}$(M)))+0.03 ×0.05 ×(-0.3+γU($S_{t,1}$(L))))+
(0.03 ×0.85 ×(-0.1+γU($A_{t,1}$(H)))+0.03 ×0.1 ×(-0.2+γU($A_{t,1}$(M)))+0.03 ×0.05 ×(-0.3+γU($A_{t,1}$(L))))],
[0.91 ×0.85 ×(-0.1+γU($A_{t,1}$(H)))+0.91 ×0.1 ×(-0.2+γU($A_{t,1}$(M)))+0.91 ×0.05 ×(-0.2+γU($A_{t,1}$(L)))+
(0.03 ×0.85 ×(-0.1+γU(B_t(H)))+0.03 ×0.1 ×(-0.2+γU(B_t(M)))+0.03 ×0.05 ×(-0.3+γU(B_t(L))))+
(0.03 ×0.85 ×(-0.1+γU($S_{t,1}$(H)))+0.03 ×0.1 ×(-0.2+γU($S_{t,1}$(M)))+0.03 ×0.05 ×(-0.3+γU($S_{t,1}$(L))))+
(0.03 ×0.85 ×(-0.1+γU($S_{t,1}$(H)))+0.03 ×0.1 ×(-0.2+γU($S_{t,1}$(M)))+0.03 ×0.05 ×(-0.3+γU($S_{t,1}$(L))))],
[0.91 ×0.85 ×(-0.1+γU($B_{t,1}$(H)))+0.91 ×0.1 ×(-0.2+γU($B_{t,1}$(M)))+0.91 ×0.05 ×(-0.2+γU($B_{t,1}$(L)))+
(0.03 ×0.85 ×(-0.1+γU(S_t(H)))+0.03 ×0.1 ×(-0.2+γU(S_t(M)))+0.03 ×0.05 ×(-0.3+γU(S_t(L))))+
(0.03 ×0.85 ×(-0.1+γU($S_{t,1}$(H)))+0.03 ×0.1 ×(-0.2+γU($S_{t,1}$(M)))+0.03 ×0.05 ×(-0.3+γU($S_{t,1}$(L))))+
(0.03 ×0.85 ×(-0.1+γU($A_{t,1}$(H)))+0.03 ×0.1 ×(-0.2+γU($A_{t,1}$(M)))+0.03 ×0.05 ×(-0.3+γU($A_{t,1}$(L))))],

Now, to choose the optimal action in the initial state variable B_t, 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.

$$U(s) = \max_{a} Q(s, a) \tag{12}$$

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

$$\pi^*(s) = \arg\max_{a} Q(s, a) \tag{13}$$

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.

$$Q(s,a) = \sum_{s'} P(s' \mid s,a)[R(s,a,s') + \gamma U(s')] = \sum_{s'} P(s' \mid s,a)[R(s,a,s') + \gamma \max_{a'} Q(s',a')] \quad (14)$$

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

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

6.3. 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. (11) 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 $U_i(s)$ be the utility value for state s at the ith iteration. Such an iteration is called the Bellman update.

[Algorithm 9. 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

$U \leftarrow U'$; $\delta \leftarrow 0$

for each state *s* in *S* do

$U'[s] \leftarrow \max_{a \in A(s)}$ Q-VALUE(*mdp*, *s*, *a*, *U*)

if $|U'[s] - U[s]| > \delta$ then $\delta \leftarrow |U_0[s] - U[s]|$

until $\delta \leq \varepsilon(1-\gamma)/\gamma$

return *U*

$$U_i(s) \leftarrow \max_{a \in A(s)} \sum_{s'} P(s' \mid s,a)[R(s,a,s') + \gamma U_{i-1}(s')] \quad (16)$$

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. (10) is optimal. A detailed algorithm including the termination condition when the utility value is fully converged is shown in algorithm 9.

7. Experiment and Evaluation

Two performance evaluation experiments of intelligent tutoring systems were conducted. One is the comparison evaluation experiment between the Hidden Markov Model and the other models, and the other is the comparison evaluation experiment with Markov Decision Process model and other models.

7.1. Experiments on Hidden Markov Model

7.1.1. Analysis of Intelligent Tutoring Systems using HMM

We evaluated how students using an intelligent tutoring system constructed using a Hidden Markov Model improved their learning performance compared to those who did not use it.

Using the intelligent tutoring system, experiments were conducted on the learning performance when students had several extra-curricular sessions. Also using the intelligent tutoring system, an experiment was conducted to evaluate the amount of work done when the students were given a task after all of the learning activities or extracurricular studies on a subject.

In the first experiment, students were divided into three groups: excellent, medium, and low, and one subject was tested for extracurricular learning. The number of students who participated in the test was 350, and the test problem averaged 20 questions in each section and 450 questions in one subject. The results are shown in the following figure. Since approximately 25% of the total amount of learning was completed for each chapter or subject, the results that students had observed for each of the clause tests embedded in the intelligent tutoring system were 75% completed. The results were positive.

In the second experiment, students were divided into three groups: excellent, medium, and low, and after completing all the extra-curricular studies on one subject, the task was given and the performance was evaluated. For one subject, the average number of subjects was 20.

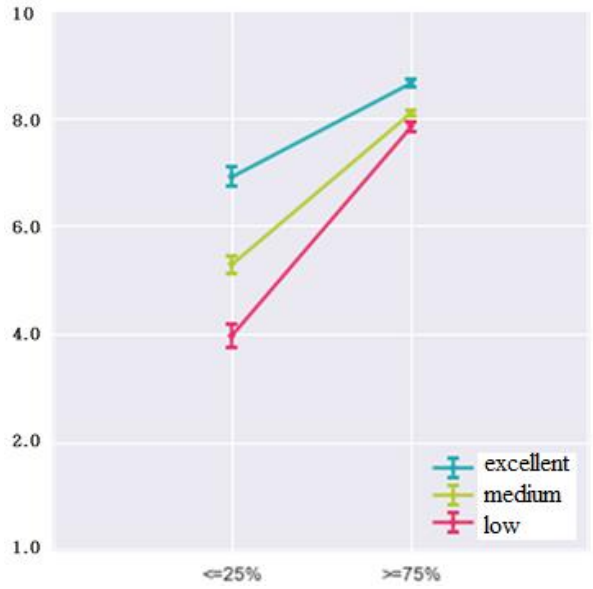

Figure 18. *When Students Were Trained for Extracurricular Learning Using an Intelligent Tutoring System(Learning Volume on the Longitudinal Axis, Student Score on the Horizontal Axis).*

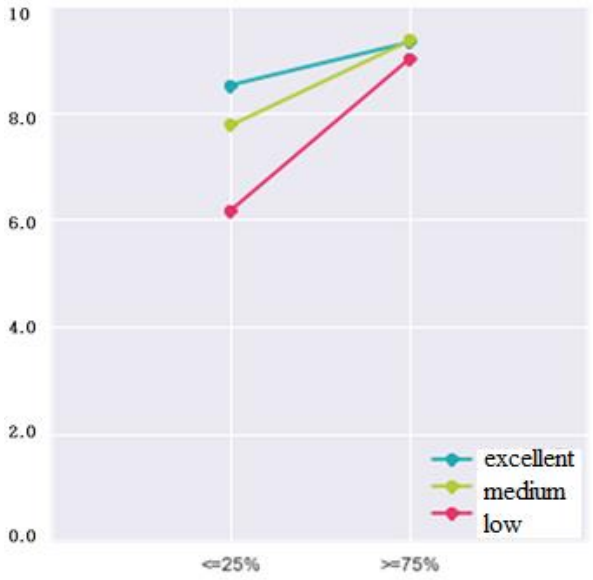

Figure 19. *The Percentage of Work Done When Assigning Students to a Task Who Completed All the Extra-curricular Learning Courses on a Subject Using an Intelligent Tutoring System (%) (the Percentage Done on the Longitudinal Axis, the Amount of Work Done on the Horizontal Axis).*

7.1.2. Comparative Evaluation Experiments with Other Models

In addition to the Hidden Markov Model, there are methods such as performance factor model(PFM), additive factor model(AFM), with partially observable Markov Decision Process. We conducted a performance evaluation experiment with the proposed intelligent tutoring system using Hidden Markov Model and intelligent tutoring system using performance factor model and additive factor model. The experiments were conducted by pre-testing and final testing and comparing the results. The test was conducted by dividing the students into high and general. The experimental results showed that the average performance of students using the Hidden Markov Model was higher than that of students using other models for intelligent tutoring systems.

Table 7. *Results of the Comparative Evaluation of the Three Models.*

policy	pre-test			final test		
	high	medium	overall	High	Medium	overall
HMM	7.5	7.2	7.1	8.4	9.2	8.8
PFM	7.4	7.0	7.4	8.6	7.4	7.9
AFM	6.8	6.0	6.7	8.9	7.6	8.2

7.2. Comparison Experiments with Markov Decision-making Models and Other Models

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.

7.2.1. 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 $O(d^{2n})$ and its size is $O(d^{2n})$ per update count [14]. On the other hand, in the Markov Decision Pro-

cess model, when the number of parents of each variable is limited to k, its size is $O(nd^k)$. 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 8. Evaluation Through Means (M) and Standard Deviations (SD).

policy	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.

7.2.2. 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.

(1) 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)

χ^2-test(χ^2 = 0.03, p = 0.86) showed no significant difference between the fast and slow classes among the three methods. Here, the χ^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 [24]. 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 9 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 9. *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)

8. Conclusion

To enable any subject teacher to develop an intelligent tutoring system for his/her subject, it requires an editor as a framework and an action platform for the data developed by the editor to operate. Our work describes some of the challenges in developing editors and action platforms.

Using the editor, we were able to build instructional resources that could implement group teaching, tutoring using HMM, and tutoring skills using MDP.

We then define and use an extended knowledge graph.

The platform uses a face recognizer recognizable student face and a speech synthesizer capable of reading text. It also allowed us to deal with the students' responses by natural language.

The HMM-based approach uses HMM for learning state prediction, additional algorithms for learning path generation, and the MDP-based approach uses learning state prediction and training path generation simultaneously in one model.

Abbreviations

AI	Artificial Intelligence
ITS	Intelligent Tutoring System
LM	Learner Model
TM	Tutor Model
DM	Domain Model
LPG	Learning Path Generator
LSEPM	Learning State Estimation & Prediction Model
GIFT	Generalized Intelligent Framework for Tutoring
TLA	Total Learning Architecture
ALOSI	Adaptive Learning Open Source Initiative
TS	Tutoring Strategy
TF	Tutoring Flow
LS	Lecture State
S	Slides
Co	Comments
V	Videos
QA	Question and Answer
V	Vertex
E	Edge
Ex	Excellent
v_g	Very Good
g	Good
aver	Average
fail	Failure
A_M	Answer_Marks
MEU	Maximum Expected Utility
MDP	Markov Decision Process
S	State
A	Action
T	Transition
R	Reward
POMDP	Partially Observable Markov Decision Process
IOHMM	Input-Output Hidden Markov Model
DBN	Dynamic Bayesian Network

DDN	Dynamic Decision Network
H	High
M	Medium
L	Low
PFM	Performance Factor Model
AFM	Additive Factor Model
M	Means
SD	Standard Deviations

Author Contributions

Song-Hwan Kwon: Conceptualization, Methodology, Writing-original draft preparation, Project administration
Yun-Ho Ko: Writing-review & editing
Chung-Song Ko: Software, Formal analysis
Jong-Nam Rim: Investigation, Software
Sin-Hye Jang: Verification
Yun-Sok Ham: Software, Data curation
Ha-Gyong Kim: Resources, Validation
Hyon-Il Son: Investigation, Software

Conflict of Interest

The authors hereby declare no conflict of interest.

References

[1] Goldberg, B., Rowe, J., Spain, R., Mott, B., Lester, J., Pokorny, B., Sottilare, R. (2018). Hands-on with GIFT: A Tutorial on the Generalized Intelligent Framework for Tutoring. Springer International Publishing AG, part of Springer Nature 2018, C. Penstein Rosé et al. (Eds.): AIED 2018, LNAI 10948, pp. 547–550, https://doi.org/10.1007/978-3-319-93846-2

[2] Sottilare, R., Brawner, K., Sinatra, A., Johnston, J. (2017). An Updated Concept for a Generalized Intelligent Framework for Tutoring. US Army Research Lab, Orlando, FL.

[3] Goldberg, B., Hoffman, M., Tarr, R. (2015). Authoring instructional management logic in GIFT using the EMAP. In: Sottilare, R., Graesser, A., Hu, X., Brawner, K. (eds.) Design Recommendations for ITS: Authoring Tools, vol. 3. US Army Research Laboratory.

[4] Chi, M. T. (2009). Active-constructive-interactive: a conceptual framework for differentiating learning activities. Top. Cogn. Sci. 1(1), 73–105.

[5] Rowe, J., Mott, B., Lester, J., Pokorny, B., Peng, W., Goldberg, B. (2015). Toward a modular reinforcement learning framework for tutorial planning in GIFT. In: Sottilare, R., Sinatra, A.,(eds.) Proceedings of the 3rd Annual GIFT Users Symposium.

[6] Sottilare, R., Robson, R., Barr, A., Graesser, A., Hu, X. (2018). Exploring Opportunities to Standardize Adaptive Instructional Systems(AISs). Springer International Publishing AG, part of Springer Nature 2018, C. Penstein Rosé et al. (Eds.): AIED 2018, LNAI 10948, pp. 547-550, https://doi.org/10.1007/978-3-319-93846-2

[7] Aleven, V. A., Koedinger, K. R. (2002). An effective metacognitive strategy: learning by doing and explaining with a computer-based cognitive tutor. Cogn. Sci. 26(2), 147-179.

[8] Craig, S., Graesser, A., Sullins, J., Gholson, B. (2004). Affect and learning: an exploratory look into the role of affect in learning with AutoTutor. J. Educ. Media 29(3), 241-250.

[9] Sottilare, R. A., Brawner, K. W., Goldberg, B. S., Holden, H. K. (2012). The Generalized Intelligent Framework for Tutoring (GIFT). Concept Paper Released as Part of GIFT Software Documentation. US Army Research Laboratory-Human Research and Engineering Directorate (ARL-HRED). US Army Research Laboratory-Human Research and Engineering Directorate (ARL-HRED), Orlando.

[10] Wetzel, J., VanLehn, K., Butler, D., Chaudhari, P., Desai, A., Feng, J., Samala, R. (2017). The design and development of the dragoon intelligent tutoring system for model construction: lessons learned. Interact. Learn. Environ. 25(3), 361-381.

[11] Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Milik, N., Holland, J., McGuigan, N. (2009). ASPIRE: an authoring system and deployment environment for constraint-based tutors. Int. J. Artif. Intell. Educ. 19(2), 155-188.

[12] Folsom-Kovarik, J. T., Raybourn, E. M. (2016). Total learning architecture (TLA) enables next generation learning via meta-adaptation. In: Proceedings of the Interservice/Industry Training, Simulation, and Education Conference, Nov.

[13] Ramirez, C., Valdes, B. (2012). A General Knowledge Representation Model of Concepts, Advances in Knowledge Representation, InTech.

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

[15] Quillian, M. R. (1968). Semantic Information Processing. In M. Minsky (Ed.), (p. 227-270.). Cambridge, Massachusetts: MIT Press.

[16] Gruber, T. R. (1993). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal Human-Computer Studies, 43, 907-928.

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

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

[19] Ehrlinger, L., Wöß, W. (2016). Towards a Definition of Knowledge Graphs. SEMANTICS 2016: Posters and Demos Track September 13-14, Leipzig, Germany.

[20] Vande Pol, J., Volman, M., Beishuizen, J. (2010). Scaffolding in teacher-student interaction: a decade of research. Educ. Psychol. Rev. 22, 271-296.

[21] Brusilovsky, P., Schwarz, E., Weber, G. (1996). ELM-ART: an intelligent tutoring system on world wide web. In: Frasson, C., Gauthier, G., Lesgold, A. (eds.) ITS 1996. LNCS, vol. 1086, pp. 261-269. Springer, Heidelberg. https://doi.org/10.1007/3-540-61327-7 123

[22] Chi, M., Jordan, P., VanLehn, K. (2014). When is tutorial dialogue more effective than step-based tutoring? In: Trausan-Matu, S., Boyer, K. E., Crosby, M., Panourgia, K. (eds.) ITS 2014. LNCS, vol. 8474, pp. 210-219. Springer, Cham. https://doi.org/10.1007/978-3-319-07221-0_25

[23] Rosen, Y., Rushkin, I., Rubin, R., Munson, L., Ang, A., Weber, G., Lopez, G., Tingley, D. (2018). Adaptive Learning Open Source Initiative for MOOC Experimentation, © Springer International Publishing AG, part of Springer Nature 2018, C. Penstein Ros éet al. (Eds.): AIED 2018, LNAI 10948, pp. 307-311, https://doi.org/10.1007/978-3-319-93846-2_57

[24] 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