Modeling Students' Learning Behaviors in Open Ended Learning Environments

By

Yi Dong

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

September 30, 2018

Nashville, Tennessee

Approved:

Gautam Biswas, Ph.D.

Akos Ledeczi, Ph.D.

Maithilee Kunda, Ph.D.

Douglas Fisher, Ph.D.

Enxia Zhang, Ph.D.

ACKNOWLEDGMENTS

I would like to express the deepest appreciation to my advisor Professor Gautam Biswas, who is also the committee chair of my dissertation. He has continuously supported my work and provided valuable suggestion and feedback about the research as well as my writing of the dissertation. Without his guidance and persistent help, this dissertation would not have been possible.

I would like to also thank my committee members, Professor Akos Ledeczi, Professor Douglas Fisher, Professor Maithilee Kunda, and Professor Enxia Zhang, for their help as well as the insightful comments about the research work of this dissertation. Their questions and suggestions about my work have encouraged me to reinforce the research and performed more robust analyses from various perspectives.

I thank all my fellows in our group who helped me in solving various challenging problems I encountered during my research. Also, I thank the Vanderbilt University for admitting and providing me the opportunity for pursuing my degree.

Last but not the least, I would like to thank all my family members for supporting me spiritually throughout my research work and my life abroad.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

In recent work on computer-based learning environments, there has been a shift in focus from learning specific domain knowledge to systems that combine learning of domain knowledge with thinking and problem-solving skills, learning strategies, metacognition, and self-regulation [1, 2]. The goal of these environments is to help students learn the broader and more transferable processes, such as how to learn and how to solve problems, not just to become proficient in a single learning topic. Open-Ended Learning Environments (OELEs) [3, 4] are a class of environments developed with these goals in mind, providing students with a learning goal, usually in the form of a complex problem or a modeling task, and a set of tools that support the problem-solving task [5, 6, 7]. Students are given freedom and choice with regard to how they combine the use of the tools to acquire and review the required knowledge, how they use this knowledge to construct their solutions, and how they monitor their progress by checking their evolving solutions to ensure their correctness [8, 9].

However, open-ended problem solving can present significant challenges for novice learners [10, 11]. To succeed, learners need to make choices regarding how to structure the solution process, explore alternative solution paths, develop awareness of their own knowledge and problem-solving skills, apply strategies that support more effective learning and problem-solving, and monitor progress as they work toward a solution [12, 13]. Given the complexities involved in working with OELEs, students may need guidance and support to help them progress in their learning and problem-solving tasks and to eventually achieve their learning goals. Researchers have suggested that guidance focused on helping learners to develop good learning behaviors and strategies may be more effective, especially as it pertains to supporting preparation for future learning [14, 15]. Therefore, a

very important component for the success of OELEs is the scaffolding they provide to help students improve their learning behaviors and strategies and, as a result, their performance. An important component of effective scaffolding is to deri learner models that can capture learners' cognitive and metacognitive processes from their interactions with the learning environment. This motivates the design of sophisticated and appropriate methodologies for the next generation of learner modeling.

Another important aspect of OELEs is that data regarding students' learning activities can be logged and stored, and this provides opportunities for developing data-driven modeling methods for constructing rich and accurate learner models. Data mining and machine learning techniques have been applied by the Educational Data Mining (EDM) community to analyze students' learning processes and behaviors in a data-driven manner. It is essential to implement appropriate algorithms on data collected from OELEs and embed them into learner modeling methods in such a way that the resulting learner model can capture learners' cognitive and metacognitive processes. Depending on the purpose, various machine learning algorithms can be applied to learner modeling, including categorizing learners into groups (by unsupervised learning or clustering), differentiating between different types of behaviors among learners (by supervised learning or classification), and predicting learning outcomes (predictive modeling).

The goal of this thesis research is to develop data-driven learner modeling methods using data mining and machine learning techniques that accurately model and assess students' learning behaviors. Typically, to attain accuracy, data-driven modeling methods require large volumes of data because of the complexities of learning and therefore, the wide variety of behaviors learners can exhibit in OELEs. Many OELE-based studies collect students' data in small samples (e.g., data from a class of 100 or fewer students learning within an OELE). This results in the *data impoverishment problem*, and mitigating this problem is one of the primary contributions of this research. Much like computer programs developed for finding useful moves or patterns in complex, adversarial game environments, we apply

reinforcement learning techniques to build and analyze models of learner behaviors [16]. Starting with "weak" learner models constructed from the originally collected data, reinforcement learning is employed to iteratively simulate a larger variety of students' learning behaviors. The learner model is then updated using the additional learning behaviors, and the result is likely to be a more accurate and complete learner model.

In the rest of this chapter, we apply our understanding of student problem-solving in this environment to generate a more formal problem statement in Section 1.1. We then discuss the challenges associated with the learner modeling method in Section 1.2 and the main contribution of this dissertation in Section 1.3.

1.1 Problem Statement

Since OELEs are learner-centered learning environments, actions and their corresponding contexts vary among individuals. Some of the traditional learner modeling methods derive learner models by representing students' knowledge as a subset of the expert model that is typically a representation of the domain knowledge to be learned. This expert-centered methodology for learner modeling fails to better adapt to individual approaches to learning and problem solving that may differ in significant ways from experts. In order to develop good instructional strategies, empirical assessment and understanding of students' learning behaviors are critical. This requires the instructor to assess and understand the individual differences between learners, and thus, be able to provide adaptive scaffolding to the learners.

Learner-centered data-driven approaches for learner modeling can overcome these limitations as mentioned above. Experts can use information discovered from students' data to build learner models for different groups of students. These data-driven models are then applied as the basis for providing feedback to different individuals. However, learner data collected from OELEs can be complex and hard to interpret, and manual assessment of the learner becomes intractable in many cases. Even for moderate amounts of OELE-collected

data, it is difficult for domain-experts to discover all the information that is important for understanding learners' behaviors and for developing analytic measures for characterizing user behaviors.

Therefore, developing algorithms that are generally applicable to learner modeling, especially those that can utilize advanced data mining and machine learning techniques, plays an important role in assisting researchers to assess and improve intelligent tutoring systems so that users can learn more effectively and efficiently. These learner modeling methods need to satisfy a number of requirements, that include:

1. The ability to discover implicit and useful information (e.g., determining students' cognitive and metacognitive states from their activity data).

2. Adaptability to different OELEs and learning purposes.

3. The ability to help in making decisions on providing scaffolding (e.g., determining the appropriate estimation to measure students' status and performance and to predict learning outcomes).

4. The ability to address the data impoverishment problem.

The effort made to develop and improve learner modeling can be regarded as an important and useful step toward developing more effective OELEs.

To meet these requirements, a novel set of techniques that combine the use Hidden Markov Modeling (HMM) [17], Coherence Relations (CR) [7], and Monte Carlo Tree Search (MCTS) [18] coupled with a reinforcement learning methodology [19] is proposed as the basis to form a data-driven learner modeling method that helps decision making in OELEs for helping to identify under-performing students, assessing their cognitive and metacognitive processes, and providing scaffolding to improve their overall learning performance.

The core representation of the learner model in this research is the Hidden Markov

Model (HMM), which is a generative stochastic model for modeling students' activity sequences. The reasons for choosing HMM are:

1. HMMs in learner modeling are known for their abilities to model learning patterns and strategies that students employ [20, 21, 22]. They can form the basis for discovering explicit and implicit cognitive and metacognitive processes.

2. HMMs are generally applicable for modeling learning behaviors in different OELEs as long as the behavioral actions in OELE are well defined and logged as action sequences.

3. The HMM representation can also be used for predictive purposes [23, 24].

4. The typical training algorithm of HMMs (i.e., Baum-Welch re-estimation) is known to be polynomial in complexity [25], which means that it can be applied to process data at scale.

In order to address heterogeneity among students' learning behaviors, in previous work, we have extended HMM learning to include an HMM-based clustering method to help characterize students' activity sequences into groups of similar behaviors [26]. HMM clustering produces a set of clusters where each cluster, represented by a unique HMM, captures similar behavior among students.

To improve the accuracy of the HMM-based learner model, as well as to handle the potential data impoverishment problem, we propose a reinforcement learning approach to iteratively improve the HMM model(s). For each iteration of the reinforcement learning algorithm, we apply Monte Carlo Tree Search (MCTS) using the HMM and coherence metrics to simulate students' activities and measure the corresponding learning outcomes as the basis for model refinement. The original activity data, combined with the simulation results of the MCTS (i.e., sequences of simulated students' actions) are used to re-learn HMMs as refined learner model representations. The MCTS, which has been widely applied to design evaluation functions for complex game play (e.g., MCTS for AI design in

Go [16]), is a novel application for simulating learning behaviors for learner modeling. In this thesis, the MCTS is configured to adapt the model representation (i.e., the HMM), and the search policies within the MCTS are governed by the original HMM structure, as well as coherence relations to capture students' learning patterns. At each iteration of reinforcement learning, the simulation results combined with the original behavior sequences form the basis for iterative refinement of HMMs and reconfiguration of the clusters. The resulting learner model is expected to (1) better categorize students' behaviors based on the refined HMM-based clusters, (2) better assess learning behaviors, and (3) predict potential performance outcomes based on HMMs within corresponding clusters to assist scaffolding.

1.2 Challenges in Data-Driven Learner Modeling

Recent developments of hardware (e.g., CPUs) and software (e.g., cloud computing) have significantly increased computational power, which enables researchers to apply data mining and machine learning algorithms to large volumes of data collected from OELEs. However, this also creates the following challenges to data-driven learner modeling methods:

- *Data collection challenges*. Data-driven methods require large volumes of rich data to support accurate and robust learner modeling. However, collecting such data from OELE environments can be a complex and time-consuming process:

    1. It is not easy to define a general data format that every OELE can follow to log learners' learning process data because of the differences between OELEs. For example, information acquisition, solution construction, and solution assessment actions can be defined differently depending on the different design of the learning environments and learning goals. Moreover, there can be addon tools allowing learners to take on additional actions that do not exist universally across OELEs (e.g., starting a conversation with the virtual agent in Betty's

Brain).

2. To design an OELE that successfully implements a specific learning purpose is itself a big challenge. Resources and tools should be designed appropriately for different purposes, and the tools developed must fit within the scope of K-12 learning and problem-solving. OELE design should also have a robust and highly reusable mechanism for data collection that ensures high data availability for every individual and/or longitudinal study.
3. Finding instructors and a sufficient number of students to work in OELE-based studies in short periods of time are hard. The OELE should be able not only to collect learning data but also to help students acquire knowledge and develop or improve their cognitive and metacognitive skills to attract and convince schools to participate in OELE-based studies.

- *Modeling challenges*. To choose an appropriate modeling method among the various modeling techniques available [27] is quite challenging as it requires expertise in learning analytics. For example, to analyze action transitions of students in Betty's Brain, a Markov model is promising as it takes into consideration the student's last action and its subsequent status changes to determine the probability of the next action. However, students can behave differently, which leads to the situation where a single Markov model is insufficient to capture their action transitions in general. In these situations, one may use a *stereotypes* modeling method [28] to characterize students into different groups. In addition, a single Markov model in a sense shows how a student reacts to the immediate prior action and its consequences, but it is unable to capture action patterns that reflect students' cognitive strategies made up of multiple actions (e.g., a strategy where students apply acquired information to construct the solution but on a subsequent solution assessment action discover an error and delete the additions made to the causal map). On the other hand, student behaviors also evolve over time, and the modeling representations must accommodate the temporal

evolution of learning behaviors.

- *Machine learning challenges*. Although many advanced machine learning techniques have been developed and shown to be useful in practice, it is still challenging to apply them to derive learning analytics measures. For example, in the *stereotypes* modeling method [28], which applies a clustering algorithm to characterize learners into groups based on their activity data, feature selection becomes a challenge. Unlike some other applications, many different forms of features can be derived from OELE data. Therefore, feature selection is not predefined but depends on the particular properties and behaviors of the students that one wants to derive from the analysis. Besides, relations between learning activities and final outcomes can be affected by learners' psychological and mental status, which is not explicitly interpretable from the logged data. In such situations, explicit assumptions about missing features should be made before carrying out clustering analysis. Despite this, researchers may face the data impoverishment problem (e.g., building learner models using data collected from a class of students working in an OELE over a short period of time). Therefore, it becomes interesting and challenging to apply machine learning algorithms (e.g., reinforcement learning) to solve the data impoverishment problem and generate good enough modeling results.

- *Verification and validation challenges*. Verification and validation is an important process for learner modeling. Learner modeling should be verified based on two perspectives:

    1. For different purposes and circumstances, the chosen modeling methods should be verified as among the best methods for helping to achieve the modeling goals and resolve the modeling difficulties.
    2. Any data mining and/or machine learning algorithms applied in learner modeling should be verified for their technical correctness and also robustness if

needed (e.g., the algorithms should be able to resist changes in the application or data format).

Verification approaches require expertise in modeling algorithms and understanding of the data collected from the corresponding learning environment. And, to ensure the robustness of the machine learning methods, it is important to collect data from multiple classrooms, which may be hard to achieve. Therefore, it becomes challenging to implement verification and/or validation methods, such as cross-validation and predictive model validation.

1.3 Contributions of the Dissertation

The main contributions of this dissertation can be summarized as follows:

1. We introduce a novel data-driven learner modeling approach. This approach is general and can be applied to different OELEs using Reinforcement Learning to generate learner models that are more accurate than traditional methods.

2. The learner modeling approach combines the use of multiple modeling and machine learning techniques, such as HMM, HMM clustering, Reinforcement Learning, and Monte Carlo tree search. Some other techniques have also been applied to help enhance the effectiveness of the modeling approach (e.g., Bayesian Information Criterion for determining the best number of hidden states in HMM and coherence relations to help performing MCTS).

3. The learner modeling approach is generally applicable to different OELEs. We convert data collected from different learning environments into a general action-view representation, where actions capture students' interactions with the learning environments, and views capture the context in which students perform the actions.

1.4 Organization of the Dissertation

The next chapter reviews the state of the art in learner modeling as well as the data-driven approaches for learner modeling. Chapter 3 presents the learner modeling approach at a higher level. The details of the learner modeling methods and the reinforcement learning algorithm applied to enhance learner modeling are illustrated in Chapter 4. Chapter 5 presents experiments and analyses by applying the learner modeling methods to data collected from two different OELEs. Chapter 6 presents the discussions and conclusions from this work.

Chapter 2

Background

To derive effective and accurate learner models, it is essential to understand the current state of the art in learner modeling. In this chapter, we review such learner modeling methods and present a summary of the state of the art and use it to motivate the approach that we will be developing in this thesis.

A learner model for Intelligent Learning Environments (ILEs) that includes OELEs is typically, an inferred model of individual learners' cognitive and metacognitive states. The overall goal of the learner modeling is to help the systems adapt to the learners' needs [29], which is the basis for personalization in computer-based educational applications[27]. To successfully construct learner models for OELEs, it is important to be clear about (1) the **purposes of the leaner models**, (2) the **learner modeling methods** and (3) the **applications of the learner models**. These are discussed in detail in the following subsections.

2.1 Purposes of Learner Modeling

For learner modeling, people may want to ask what it is about the learner that we want to model so we can (1) adapt to the learner's state of knowledge, (2) track their learning and problem-solving behaviors, and (3) provide feedback when they are unable to progress in their learning and problem-solving tasks. For example, in addition to their state of knowledge, we can model learners' approaches to learning, i.e., their cognitive and metacognitive processes that can include understanding domain content, critical thinking, perception, problem-solving, and decision-making abilities. Additionally, researchers have also established that learners' affective states, motivation, and engagement that influence their learning performance and behaviors [27]. Based on the various characteristics to be modeled, we can establish six main purposes of learner modeling [30]:

1. *Corrective*. The learner model aims to identify learners' errors and misconceptions so that the feedback provided can help them to realize their errors and make corrections. This requires an analysis of expert-provided or domain-related information and learners' understanding of the corresponding knowledge by deriving the differences between the information acquired by the learner and the expected information. A challenge one faces in this approach is to determine to what extent the feedback being provided helps the learner to understand their error or misconception rather than just passively accepting the feedback. The corrective learner model is expected to provide an immediate determination of errors so that feedback can be provided in time for the user to reflect on the error and make the necessary corrections.

2. *Elaborative*. In this case, learner modeling is used to determine the nature of the scaffolds or feedback that will help the learner extend their domain knowledge and problem-solving skills. Elaborative scaffolding typically provides new materials to support specific knowledge learning and to help learners improve their understanding. The purpose is to re-introduce the knowledge or provide refinements of the knowledge to help learners overcome their weaknesses and misconceptions. This approach should have the ability to capture information for detecting learners' understanding, especially when they demonstrate weaknesses. Unlike the analyses in *corrective* model, more sophisticated techniques are needed to understand the level of the learner's understanding.

3. *Strategic*. The objective is to assist in decision making for switching instructional strategies at a higher level than local tactics. This learner model should have the ability to gather information about learners' behaviors and performance for different teaching strategies and contexts and to measure how successful each strategy is in enabling learning gains for an individual. Based on the cumulative data from the model, mechanisms are designed to decide the best teaching strategy in different

circumstances.

4. *Diagnostic*. This approach focuses on analyzing the state of learners, which is the general purpose of many learner modeling methods. It can be used to analyze any subset of a learner's characteristics in order to help decision making in teaching. The results of the analysis can also help in other modeling purposes in terms of providing measurements for corresponding learning states.

5. *Predictive*. A predictive model predicts the outcomes of the actions taken by learners, where outcomes are often a measure of performance. The model derivation may require large volumes of cumulative data and the use of appropriate data mining and machine learning techniques to accurately simulate learning behaviors based on the current states and previous actions of a learner group. Based on the association between actions related to solution construction and performance measures, the model provides estimated outcomes for the prediction.

6. *Evaluative*. This approach is often used with other modeling techniques to evaluate corresponding metrics for decision making and prediction. In addition, the evaluative learner model should be able to gather aggregated information and provide assessments of learners' behavior and performance for different learning environments and under different requirements.

An accurate learner model may include more than one of the modeling purposes discussed above. It can be a challenging task to design mechanisms for multi-purpose learner models.

2.2 **Learner Modeling Methods**

Mark Elsom-Cook [30] characterizes most existing learner modeling methods into *expert-based modeling* and *learner-based modeling*.

Expert-based modeling

In this modeling schema, the learner model is constructed from an expert model that is typically a representation of domain knowledge. The learning objectives are a set or subsets of the whole expert knowledge representation. *Expert-based modeling* is further divided into two categories [30]:

1. Subset-based methods. The expert model is defined as a set of atomic knowledge representation units where for each unit it is assumed that a learner either understands or does not understand the unit. Therefore, the representation of student knowledge is always a subset of the expert model. The expert model can be:

    - A network representation: the knowledge is represented in a graph where the nodes are entities that can be concepts and the edges indicate relations between the concepts. The learner model is a subgraph of the network representation, and the process of knowledge understanding can be simulated as a traversal of the network from multiple starting points.
    - Rule-based representation: the knowledge is represented as a set of constraints that are usually in a logical format, and learners' understanding of the knowledge is represented by the subset of constraints that are not violated by them.

2. Perturbation-based methods. Unlike subset-based methods where the learner's understanding toward each atomic knowledge unit is defined as either knowing or not knowing that component of the expert model, perturbation-based methods assume that the learner can have an understanding of the knowledge unit that is not covered by the expert model. An example is the *buggy model* [30]. This methodology is more comprehensive for capturing learners' misconceptions but, on the other hand, results in a more complex measure of understanding for each knowledge unit. For example, the expert model can be a sub-graph or subset of a network representation or constraints that covers both correct and incorrect knowledge, while the learner model

is developed based on this broader knowledge representation and is not necessarily subject to the expert model.

Learner-based modeling

In the expert-based modeling methods, learners succeed when their knowledge representations clone the corresponding expert models. However, this methodology is incomplete because it does not take into consideration the many different ways learning and problem-solving can occur. The wide range of possibilities can be attributed to individual differences, incorrect paths chosen to solve a problem, and the alternative paths leading to success that are not captured by expert models. In OELEs, learners are free to construct solutions in various ways, and there can be a number of different correct solutions (e.g., different computational models in CTSiM [1]). This results in big behavioral differences between individuals, which means that an expert model is insufficient for capturing the learning behaviors of all learners. Therefore, learner-based modeling methods, which focus on individual differences, are more appropriate for deriving learner models in OELEs.

In expert-based modeling, the learner model (i.e., the domain representation of expert knowledge) is defined in advance of any learning process. However, learner-based modeling is different in that the expert knowledge is used merely as the basis for assessing learners' performance while the learner model itself is built from learners' behaviors. The learner-based modeling methods focus on analyzing data collected from learning environments to discover implicit cognitive and metacognitive processes that represent the mechanisms whereby individual learners approach knowledge understanding through various actions. Compared to expert-based modeling, this methodology is more amenable to methods in exploratory data mining and machine learning techniques, and it better utilizes modern computational mechanisms and capabilities. Thus, learner-based modeling methods, which are mostly considered to be data-driven approaches, have been drawing more attention in recent times.

Existing Learner Modeling Methods Some of the most well-known learner modeling

methods within the last two decades are summarized in [27]. We briefly review the primary methods below.

1. *Overlay model* (expert-based). This is a typical expert-based learner modeling method that was first introduced in [31]. The learner model in this method is built as a subset of the expert model. In the original version of the overlay model, a Boolean value of True/False associated with a knowledge element indicates whether the student knows or does not know the corresponding piece of knowledge. The knowledge elements are generated by decomposing the expert model. In later developments of this modeling method, qualitative measures such as level of understanding or probability of understanding were assigned to knowledge elements [32]. Basically, the overlay model captures learners' understanding with respect to independent knowledge elements, and this requires a decomposition of the expert model and an accurate estimation of the students' knowledge at these levels. Although it has been used extensively [33, 34, 35], the overlay model is insufficient because its scope is limited to expert concepts. It does not accommodate students' alternate conceptions and misconceptions; therefore, it may be hard to model students' prior knowledge and individual differences.

2. *Constraint-based models* (expert-based). First proposed in [36], constraint-based models (CBM) capture aspects of both the domain and the student model. A constraint is characterized by a relevance condition and a satisfaction clause that each constraint fails to hold when the satisfaction clause is false for the relevance condition. For the expert model, CBMs represent the set of all constraints, while for the student model, they represent the set of all violated constraints. This modeling method provides a way to handle the variations from the expert model in a tractable manner.

3. *Perturbation models* (expert-based). As another expert-based modeling method, this

is the extension of the original overlay model by including not only students' understanding of the expert model but also their misconceptions. This allows a better representation of students' knowledge level. However, this approach may still have the same problems as the overlay model.

4. *Stereotypes* (learner-based). First introduced in [28], stereotypes are generated by clustering all possible users of a learning environment according to certain characteristics, where the homogeneity of given characteristics for all students within each cluster is maximized. In stereotypes modeling, there is no need for an explicit definition of the knowledge elements as in the overlay model. Applications of stereotypes modeling can be found in [37]. However, stereotypes modeling suffers from the problem of inflexibility as stereotypes need to be pre-defined for all possible "students" before actually being used.

5. *Machine learning techniques* (learner-based). Learner modeling using machine learning techniques can extend from the analysis of students' learning performance given their prior knowledge to more comprehensive approaches that study the relations between students' prior knowledge and behaviors in the system and their performance on the learning task. According to [38], learner modeling using machine learning techniques can be characterized into: (1) methods to induce a single, consistent student model from multiple observed student behaviors; and (2) automatically extending or constructing from scratch a library of student models. The versatility of machine learning techniques allows for the development of a number of learner modeling applications. For example, the k-nearest neighbor algorithm has been applied to determine the stereotypes in [39]. Machine learning techniques have also been applied to detect students' off-task behaviors in [40].

6. *Cognitive theories* (learner-based). In this modeling methodology, cognitive theories are employed to model students' learning behaviors drawing upon our understanding

of the processes of human thinking and understanding.

7. *Fuzzy student modeling* (learner-based). In intelligent tutoring systems, human instructors do not directly interact with the students, which cause uncertainty of data collected from the systems. For example, the issues such as network congestion or software crash can cause difficulties and problems in gathering information about students mental state and learning behavior. To solve the uncertainty in this modeling method, *fuzzy* logic methods can be applied to learner modeling [41, 42, 43].

8. *Bayesian networks* (learner-based). Bayesian networks (BN) can be another solution for capturing the uncertainty in the relations between the nodes that make up the learner model [44]. The nodes in a Bayesian network can represent different students' states, such as learning phases and cognitive and affect states, while the causal links between the nodes can represent the relations between the states. The BN model can be (1) an expert-centric model built up by experts, (2) a data-centric model statistically based on observable data collected from intelligent tutoring systems, or (3) an efficiency-centric model that incorporates from models (1) and (2) to optimize efficiency. Examples that apply BN in learner modeling can be found in [45, 46].

Each of these learner modeling methods can achieve one or more modeling purposes (subsection 2.1), and they are summarized in Table 2.1. As we can see from the table:

- *Overlay*, *Constraint-Based* and *Perturbation* models are *corrective* due to their ability to identify error or misconceptions in different ways. They are also *evaluative* if the designed system can ensure the aggregation of the corresponding information.

- The *stereotypes* method is *strategic* due to its ability to change teaching strategies so that they adapt to pre-defined student groups with different characteristics.

- Because of the various algorithms developed for *machine learning techniques*, they

Table 2.1: Summary of Existing Learner Modeling Methods

	Modeling Methods	corrective	elaborative	strategic	diagnostic	predictive	evaluative
Expert-based Modeling	Overlay model	√					√
	Constraint-Based Model	√					√
	Perturbation	√					√
Learner-based Modeling	Stereotypes			√			
	Machine learning techniques	√	√	√	√	√	√
	Cognitive theories				√		
	Fuzzy modeling		√		√		
	Bayesian networks		√		√		

can be used for many different modeling purposes by choosing appropriate algorithms. For example, we can run clustering algorithms similar to *stereotypes*, and classification algorithms can be used to *evaluate* learners by comparing their behaviors in different groups.

- *Cognitive theories* focus on analyzing learners' learning behavior, which is a good way for *diagnosing* their learning states.

- Because *fuzzy modeling* and *Bayesian networks* are able to handle uncertainty in learner modeling, they are able to detect weak understanding and make decisions on providing extra resources. Therefore, they are useful for *elaborative* purposes. With fuzzy techniques and diagnostic reasoning, respectively, they can also be used for *diagnosing* learners' characteristics.

These existing learner modeling methods have covered all the modeling purposes for assessing learner behaviors. But it is also essential to solving practical problems by knowing *how to apply learner models*.

2.3 How to Apply Learner Models

This section discusses how to apply learner modeling given the categorizations of modeling purposes and modeling methods presented in the previous sections. Besides the "built-in" modeling challenges discussed in section 1.2, there are other challenges that need to be resolved in order to construct an appropriate and accurate learner model:

- The composition of learner modeling purposes. We can choose a subset of the modeling purposes described in section 2.1, as well as extensions of these purposes, to form a general goal for the learner modeling method. For example, in a learner model targeted at providing scaffolding to improve learning effectiveness, the learner model needs to have *predictive* and *evaluative* abilities so as to identify under-performing learners that need potential assistance. Also, depending on the way scaffolding is

provided, the learner model can be *corrective* in providing direct feedback on misconceptions, *elaborative* in providing alternative reading resources, or *strategic* in providing learning strategies that are considered more effective based on the collected data in the study.

- The composition of learner modeling methods used. The modeling methods chosen should serve to achieve the selected learning purposes and should be systematically and consistently composed. For example, *stereotypes* methods that are based on clustering can be *evaluative* in identifying under-performing learners if the features selected are good differentiators of performance. But they are not *predictive* unless other mechanisms have been added for doing prediction. Additional approaches may need to be developed when methods are combined. For example, a clustering algorithm that is combined with a Bayesian network model to represent uncertainty in the learner model structures, can generate groups of students with similar Bayesian network representations.

A good composition of modeling purposes and modeling methods serves as the basis for accurate and robust learner models.

2.4 **Summary**

In this chapter, we have presented the state of the art in learner modeling by introducing modeling purposes, categorizing modeling methods, and presenting existing modeling methods. The learner model to be developed in this thesis consists of multiple modeling purposes (i.e., *strategic*, *diagnostic*, *predictive* and *evaluative*). Our goal in this thesis is to develop an integration of machine learning and data mining techniques to construct learner modeling methods that are relevant in OELEs. It is essentially a learner-based model, which also adopts ideas from expert-based modeling for evaluating learners' performance. For example, we use a causal map created by experts as the basis to evaluate the learner-

created causal map in Betty's Brain. The details of the proposed learner modeling approach are discussed in Chapters 3 and 4.

Chapter 3

Approach

As discussed in the earlier chapters, the learner modeling approach should adapt to different OELEs and be able to generate accurate and robust learner models. In this chapter, we present a higher level description of our learner modeling approach and discuss how it can resolve the potential challenges for generating accurate learner models. The chapter is organized as follows: we first introduce an example OELE system (i.e., Betty's Brain OELE) in section 3.1, and use it as the basis to illustrate the learner modeling approach developed in this research (section 3.2). Section 3.3 discusses the challenges and the potential solutions for generating accurate learn models. Section 3.4 presents a summary of this chapter.

3.1 An Example of Learning in OELEs

We present the Betty's Brain learning environment, developed by our lab at Vanderbilt University, as an example OELE [47]. Betty's Brain satisfies the two most important aspects of OELEs: (1) it provides a set of tools to support students' learning and problem-solving (model building in this case) and (2) it provides learners with an "open-ended" approach to problem-solving [48]. The "open-endedness" promotes exploratory learning, which can lead to learning with understanding in contrast to rote learning that is known to cause the inert knowledge problem [49].

In Betty's Brain, students learn about a science topic (e.g., climate change) by constructing a causal map, where the nodes represent scientific concepts (e.g., vehicle use, global temperature) and the directed and labeled edges are the causal relations between concepts. Such representations are known to support explanations and understanding of scientific phenomena (e.g., an increase in vehicle use may increase global temperature) [47]. Learn-

ers can acquire information about each of the concepts and convert the information they read into causal links (e.g., deforestation increases carbon dioxide) to build their causal maps. They are also provided with tools to test the correctness of their evolving models. They can do this by asking the virtual agent, Betty, to answer and explain answers to specific quiz questions using the causal map they built to teach Betty. These functionalities, as well as the other tools (e.g., requesting a conversation with Betty or the mentor agent and taking notes) that are available in Betty's Brain, are all designed to support a user-centered learning process, giving the students the ability to explore and *learn by construction*.

Based on the different tools and the task to solve, students' activities in Betty's Brain are categorized into three broad classes of OELE-related tasks: (i) *information seeking and acquisition*, (ii) *solution construction and refinement*, and (iii) *solution assessment* [7].

We describe each of these tasks in greater detail:

- *Information acquisition (IA)*: This relates to actions such as reading to learn new information (***read***) and using ***search*** methods to find specific knowledge. Taking and viewing notes is also considered to be useful for information acquisition (***notes***). Figure 3.1 shows the interface for reading the science book (a set of hypertext pages) in Betty's Brain.

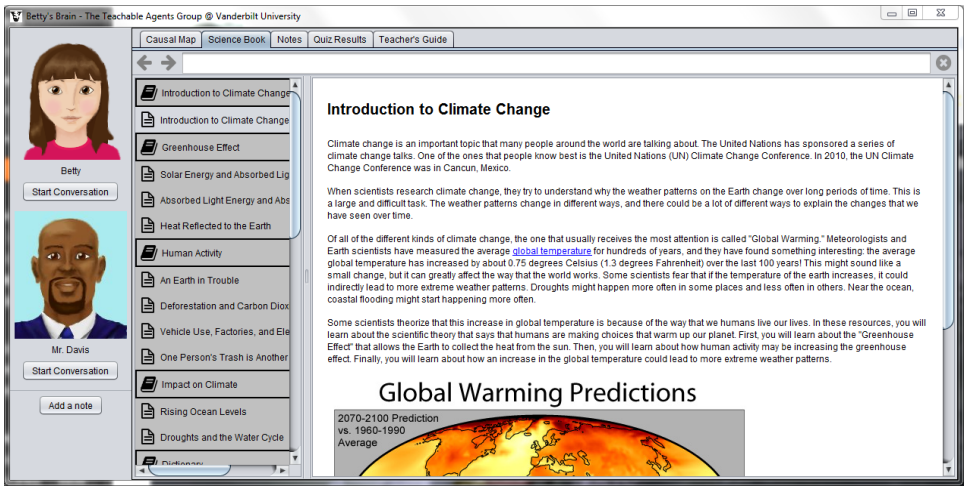


Figure 3.1: Interface for Reading Science Book in Betty's Brain

- *Solution construction (SC)*: In Betty's Brain, SC actions are causal map editing actions (***mapedit***), which consist of ***linkedit*** actions, such as add, change, or remove causal links (***linkadd***, ***linkchg***, ***linkrem***) and ***concedit*** actions such as add or remove concept entities (***concadd***, ***concrem***). Besides, students can mark the correctness of the causal links in their causal map to assist their solution construction (***clmark***) process. Figure 3.2 shows the interface that students use to build their causal maps to teach Betty.

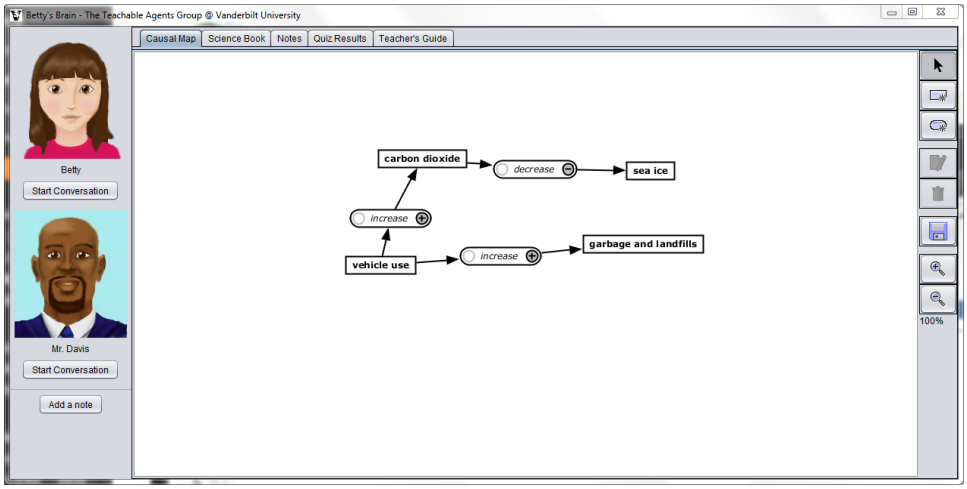


Figure 3.2: Interface for Solution Construction Actions in Betty's Brain

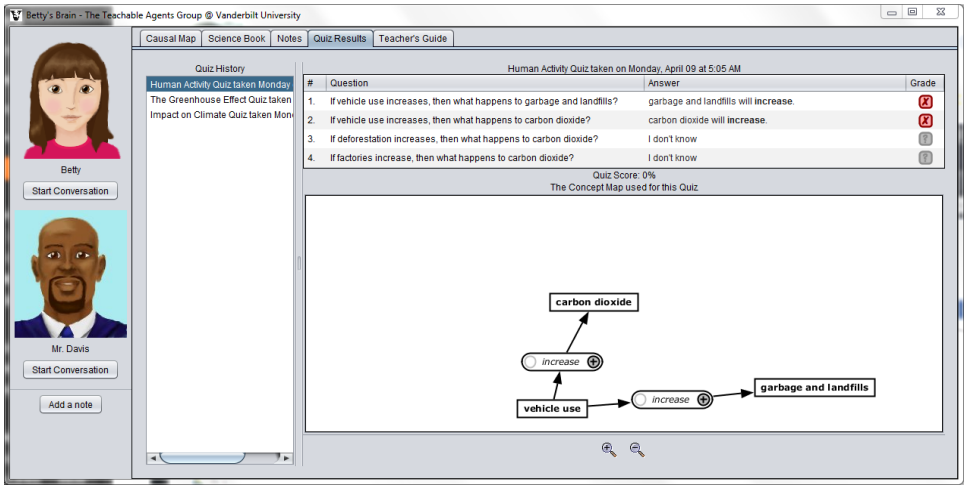


Figure 3.3: Interface for Solution Assessment Actions in Betty's Brain

- *Solution assessment (SA)*: This consists of asking Betty to take a quiz (***quiz***), answer

Table 3.1: The list of all meaningful actions students can perform in Betty's Brain

Action category	Action type	Action name/description
IA	**read**	ReadScienceBookAction
	search	SearchScienceResourceAction
	notes	AddNoteAction
		DeleteNoteAction
		ChangeNoteAction
		ViewNoteAction
SC	**mapedit**	AddConceptAction
		RemoveConceptAction
		AddCausalLinkAction
		RemoveCausalLinkAction
		ChangeCausalLinkAction
	clmark	MarkCausalLinkCorrectAction
		MarkCausalLinkWrongAction
		ClearCausalLinkMarkAction
SA	**quiz**	QuizTakenAction
		QuizViewAction
	quer	AnswerQuestionAction
	expl	ExplainAnswerAction

questions (***query***), and explain how she derived her answers using qualitative reasoning methods (***expl***). Figure 3.3 shows the interface for taking solution assessment actions in Betty's Brain.

Table 3.1 summarizes all of the primary actions available to students in the Betty's brain environment.

As students work on the system, all of their actions, as well as the progression of the causal model they build to teach Betty are logged. We use this information, in addition to other information derived from querying the user, to construct data-driven models of learner performance and behaviors. The next section provides a formal definition of the learner modeling problem and outlines the general modeling methods we have developed in this thesis.

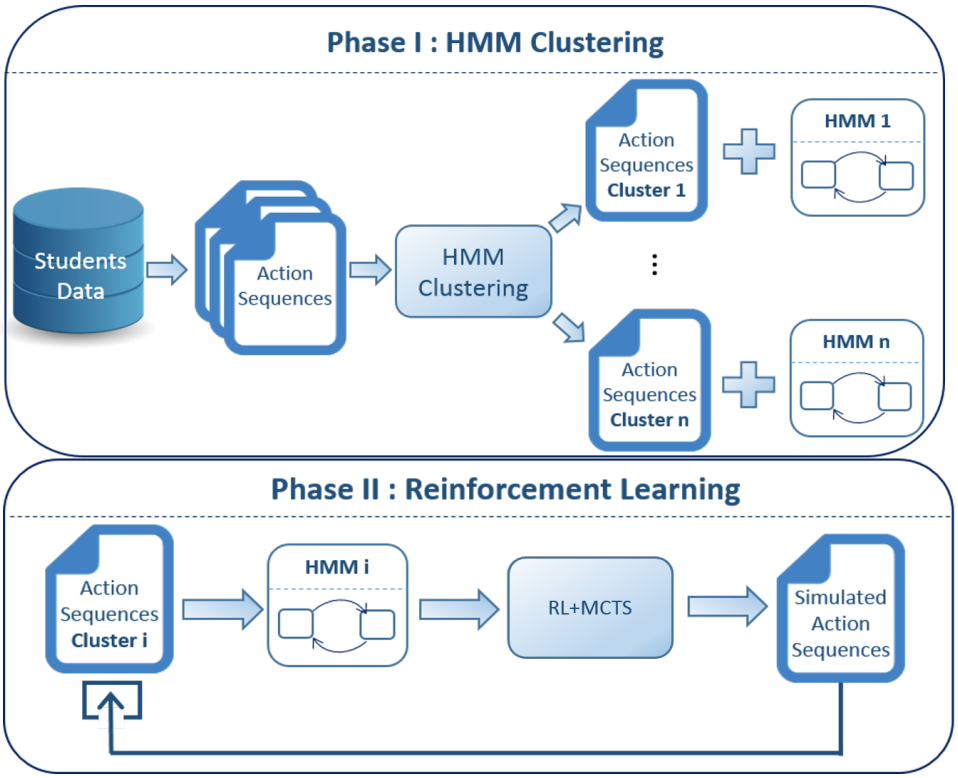


Figure 3.4: Architecture of the overall learner modeling method

3.2 Learner Modeling Approach

Figure 3.4 illustrates the approach that we have developed for our learner modeling method. As discussed earlier in section 1.1, we use HMMs to model students' learning behaviors because HMMs are able to capture explicit and implicit cognitive and metacognitive processes students employ in their learning tasks [20, 21, 22]. The HMM technique also allows researchers to go beyond simple frequency and sequence analyses, to using exploratory methods to examine how students' activities cohere in larger patterns over time [50]. And since students may exhibit a variety of behaviors and strategies, a single HMM is usually insufficient to model students' learning behaviors comprehensively. As a result, we first apply the HMM clustering method [26] to derive groups of students who show similar learning behaviors as represented by their action sequences. Corresponding to each group of students, we derive a HMM as the generative stochastic model of the group's

behaviors, and simultaneously also derive the set of coherence relations, using a set of support and effectiveness measures related to information acquisition, solution construction, and solution assessment that was developed in [7] and further expanded in [51].

To address the potential data impoverishment problem which causes inaccurate HMMs, as a next step, we iteratively generate a more accurate HMM model corresponding to each cluster. We propose a Monte Carlo Tree Search (MCTS) approach [18] and generate a search tree of student activities, starting with the current HMM, and then combine a reinforcement learning scheme (as indicated in the dashed box) to extend students' action sequences or generate additional sequences with similar behaviors. We perform the MCTS simulations to compute the rewards associated with the actions to be taken so that long-term consequences are taken into consideration. These reward values are used in the reinforcement learning scheme to generate new or to extend existing action sequences. These new action sequences are combined with the original student data to generate a newer, more complete and more robust HMM model.

The HMMs and the coherence relations are derived from students' action sequences, and are representative of their learning behaviors. On the other hand, MCTS which is used for reinforcement learning, is based on both students' behaviors and performance. In MCTS, the expansion and simulation policy is constrained by the HMMs and coherence relations while the selection policy takes students' performance measures into account. We discuss each component of the learner modeling method in the next chapter.

3.3 Challenges and Solutions

As discussed earlier in Chapter 1, the major focus of this dissertation is to develop algorithms for deriving learner models that can be used to model students' learning behaviors in terms of the cognitive and strategic processes that they employ so as to provide adaptive scaffolding to students as they learn in these environments. The learner modeling approach needs to adapt to different OELEs and be able to handle potential data impoverishment

problems. To illustrate the effectiveness of the modeling approach, we list the following problems and discuss how they are resolved at a higher level:

1. *Data representation problem*. Data collected from different OELEs can vary significantly, which causes difficulties for applications of learner modeling methods. So we propose to use an *action-view* representation, which provides a semantic framework that is applicable to different OELEs and addresses the data heterogeneity problem. This will be discussed in section 4.1.1.

2. *Model Representation Problem*. In general, the more effective model representations are a function of the purpose for which they will be used. For example, as discussed earlier, the overlay model, provides a convenient representation to compare the learners' knowledge against a chosen expert subset. The Hidden Markov Model (HMM) is our choice of representation to model the temporal characteristics of learners' behaviors in a compact manner. The HMM has been applied to model the learning patterns and strategies of students [20, 21, 22], which can be the basis for the interpretation of students' behaviors as well as their cognitive and metacognitive states. Section 4.2.1 discusses the details of HMMs applied to our work.

3. *Use of Machine Learning Approaches to support Data-Driven Modeling.* To develop a framework for adaptive scaffolding, the learner model should adapt to students that exhibit different learning and problem-solving behavior patterns. We propose to apply an HMM clustering algorithm [26] to divide students into groups of similar behaviors to increase within-cluster homogeneity for HMM representation. This will be discussed in Section 4.2.3.

4. *Data impoverishment problem.* Data collected from a single study with OELEs is likely to be incomplete, and not cover all of the different aspects of learner behaviors. This data impoverishment problem makes it difficult to derive accurate and robust learner models. For example, the data collected from one of our studies on the Betty's

Brain OELE included 98 sixth grade students. It is likely that the HMMs derived will overfit the data, especially because they are complex and students depict a variety of behaviors in the learning environments. So we propose to apply a combination of the Monte Carlo Tree Search (MCTS) and Reinforcement Learning (RL) approaches to generate simulated data to enrich an original sample of student performance and behavior data (Section 4.3.3 and 4.3.3), and, therefore, derive more accurate and more complete learner models.

5. *Verification and validation problems.* The verification and validation of our learner modeling approach will be carried out by analyzing the original HMMs as well as comparing them against the HMMs generated after reinforcement learning. During the process, cross validations and sequence mining will be applied to support our analysis and interpretations. We present some examples for adaptive scaffolding and perform empirical analyses on them.

3.4 **Summary**

In this chapter, we introduced an OELE example, the Betty's Brain system, and summarized the students' activities that can be logged during their learning process. After proper data processing, we can derive the data set of action sequences corresponding to students' learning behaviors in the OELE.

These data can then be used to generate learner model according to the approach presented in section 3.1. The approach applies reinforcement learning and MCTS to iteratively learn more accurate and complete HMMs that are derived from the original data set. Besides the learner modeling approach, we have also presented the challenges in data-driven learner modeling and how the proposed learner modeling method can resolve each of them. Details of each component of our learner modeling scheme derived using machine learning methods are discussed in the next chapter.

Chapter 4

Details of the Approach

This chapter presents our approach to enhancing learner modeling using machine learning techniques. Chapter 4.1 discusses our initial work in pre-processing and structuring the data in log files, and then applying a Hidden Markov Modeling (HMM) approach to building learners' behavior models from their activity sequences. We also discuss coherence measures that we have developed to further characterize learner behaviors. Chapter 4.2 discusses the Reinforcement learning approaches along with Monte Carlo Tree Search (MCTS) methods we have developed to enhance learner models. Chapter 4.3 provides a summary of our overall approach.

4.1 Background Techniques

In this section, we first describe the pre-processing step that we apply for generating initial learner behavior models from log data collected from classroom experiments. We then describe the use of HMM clustering techniques to generate users learning behaviors from their activity sequences.

4.1.1 The Action-View Representation and Coherence Relations

To resolve the data heterogeneity problem across different OELEs, we have developed a data representation scheme called the action-view representation [7]. The action-view representation captures information about the context in which users perform actions. The representation helps us to better interpret how students may have performed different actions. Tracking and interpreting individual actions and relations between actions in this manner provides cues to the cognitive, strategic, and metacognitive processes learners may be employing to accomplish their learning goals.

The <*action*, *view*> has been developed in [52, 7, 53]. When applied to OELEs that focus on learning by modeling, learner's *action*s can be broadly classified into (1) actions that help them to acquire the information they need to build, check, and verify their models (information acquisition); (2) actions that are related to building and refining their models (solution construction); and (3) actions that are related to checking and verifying their models (solution assessment). On the other hand, the *view* captures the corresponding context in which an action was performed.

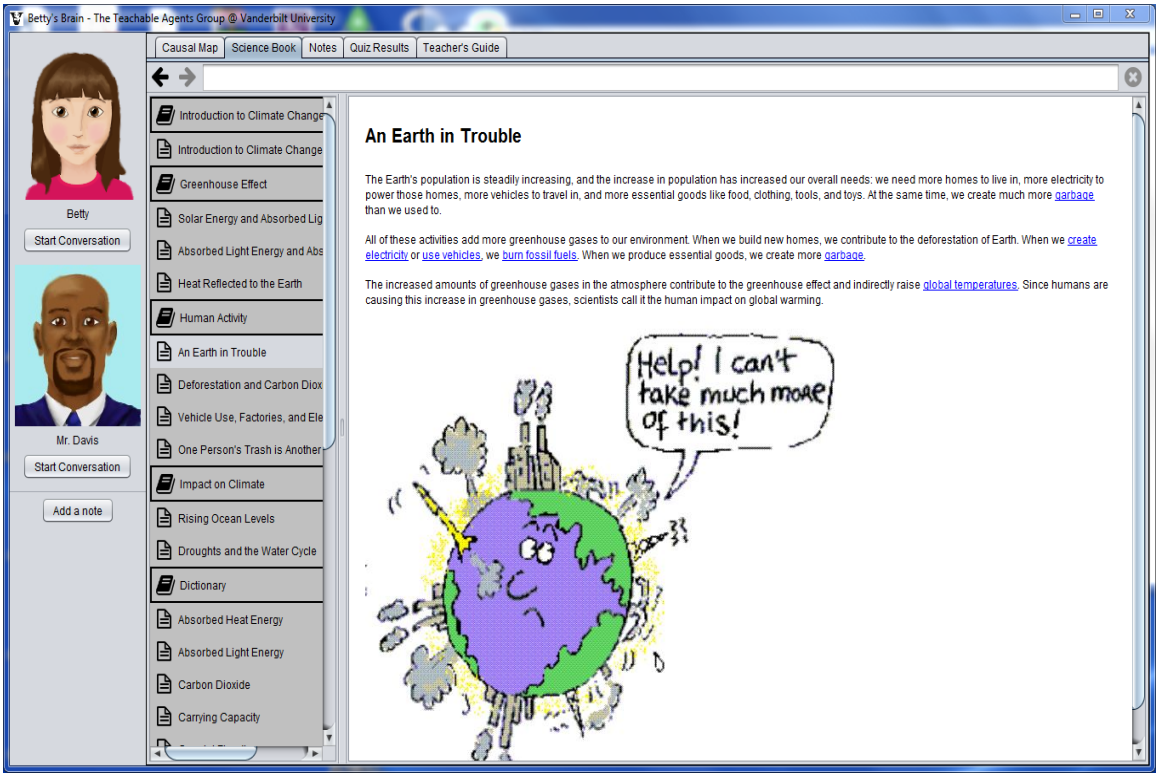


Figure 4.1: The *view* of the book page "An Earth in Trouble"

Generating action-view representation

Consider the Betty's Brain OELE that was introduced in Section 3.1. The system can generate event logs that capture every *action* taken by the student. A logged *action* corresponds to an atomic expression of intent, such as deleting a causal link or reading the science resources. The logs also contain information on every *view* that was displayed when the system was running. A logged *view* captures the information visible to a user

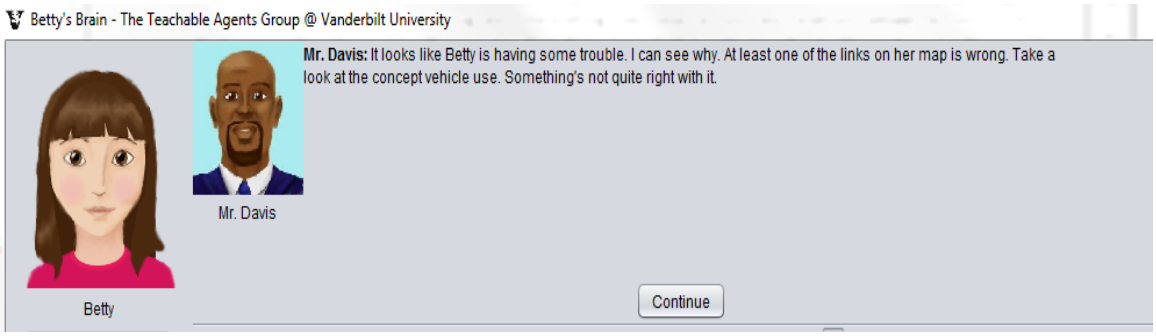


Figure 4.2: The *view* of the feedback for a quiz result

during a specific time interval. Unlike actions, which are distinct and orderable, *views* can overlap each other and span across multiple actions.

When an action is taken by the students, the system can log this *action* as well as the corresponding *view*(s) that are associated with that action. For example, when a student navigated a science book page named "An Earth in Trouble" and stayed on that page long enough (e.g., stayed for more than 5 seconds) in Betty's Brain, a "read" action is logged. Meanwhile, the *view* corresponds to the material or information on this page that the student can read is also logged (Figure 4.1). As another example, after letting Betty take a quiz and getting feedback from Mr. Davis, an action named "quiz" is logged, and meanwhile, the content of the feedback is logged as the *view* (Figure 4.2). Figure 4.3 shows an example of a sub-sequence of actions along with the corresponding views that are generated by a student who worked in Betty's Brain. The description of these actions in Betty's Brain can be found in Section 3.1.

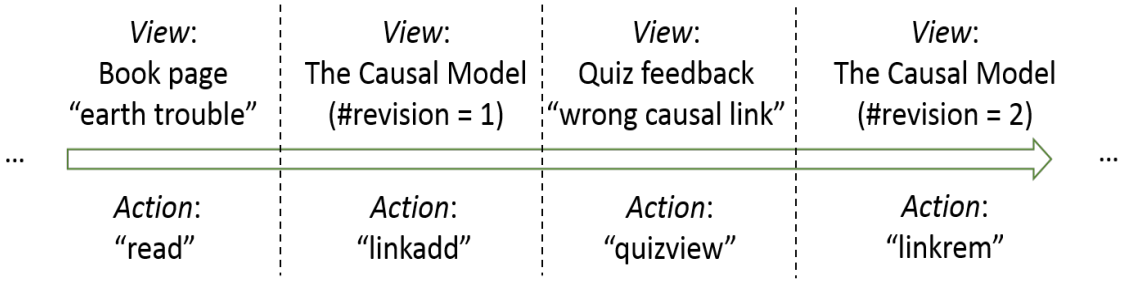


Figure 4.3: A sub-sequence of *actions* and the corresponding *views* in Betty's Brain

Each action can be associated with one or more views. For example, the action to delete the causal link "vehicle use increases garbage and landfills" can be supported by views that contains relevant information such as the science resource with page id "*earth_trouble*" and the correctness feedback (i.e., "This link might be wrong") on this causal link as shown in Figure 4.4. On the other hand, actions that start these views are considered as potential actions supporting this "DeleteCausalLinkAction" action (i.e., the two "ViewPageAction" in Figure 4.4).

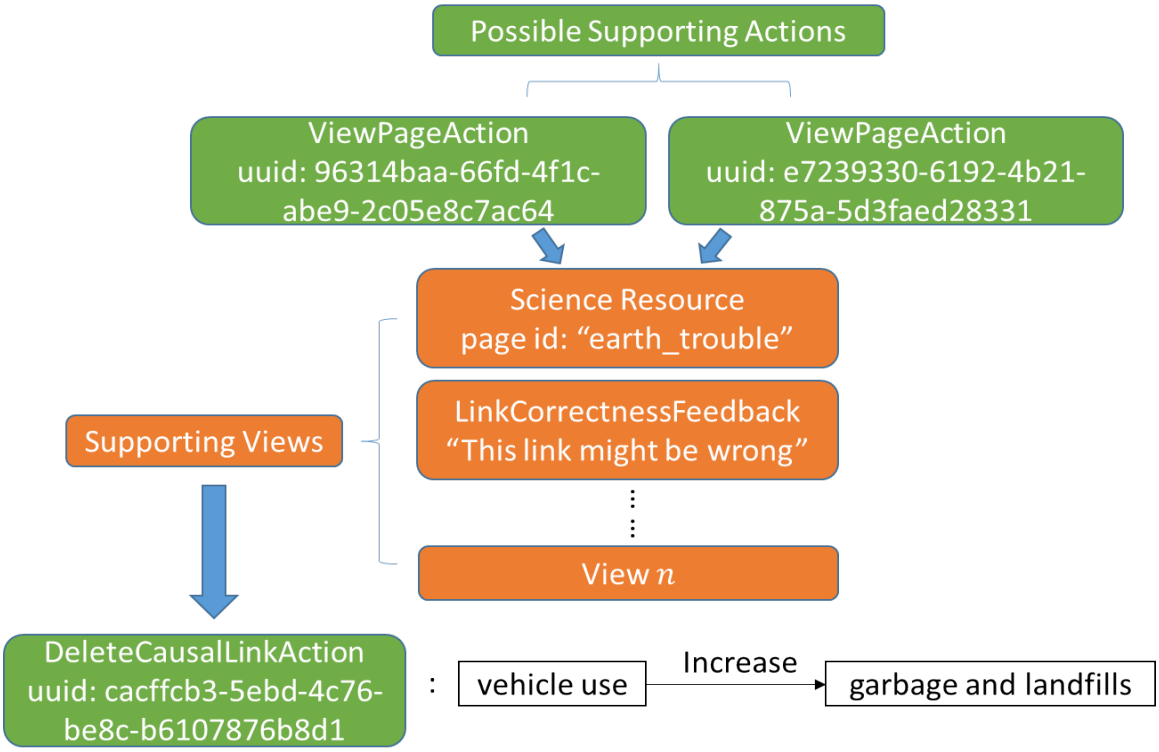


Figure 4.4: Possible actions that can support *DeleteCausalLinkAction* action.

4.2 Coherence Relations

The above discussion leads to the notion of action coherence which is formally defined in [7] as:

Definition 1 (Coherence Relation)**.** "Two ordered actions ($x \rightarrow y$) taken by a student in an OELE are action coherent if the second action, y is based on information generated by

the first action, x. In this case, x provides support for y and y is supported by x. Should a learner execute x without subsequently executing y the learner has created unused potential in relation to y. Note that actions x and y need not be consecutive."

In practice, we use a time window to determine whether an action is supported. The prior time window is chosen empirically (e.g., 2 minutes time window), and its representation is shown in Figure 4.5, where the two potential supporting actions as illustrated in Figure 4.4 are not supporting the "DeleteCausalLinkAction" because they were taken more than 2 minutes before.

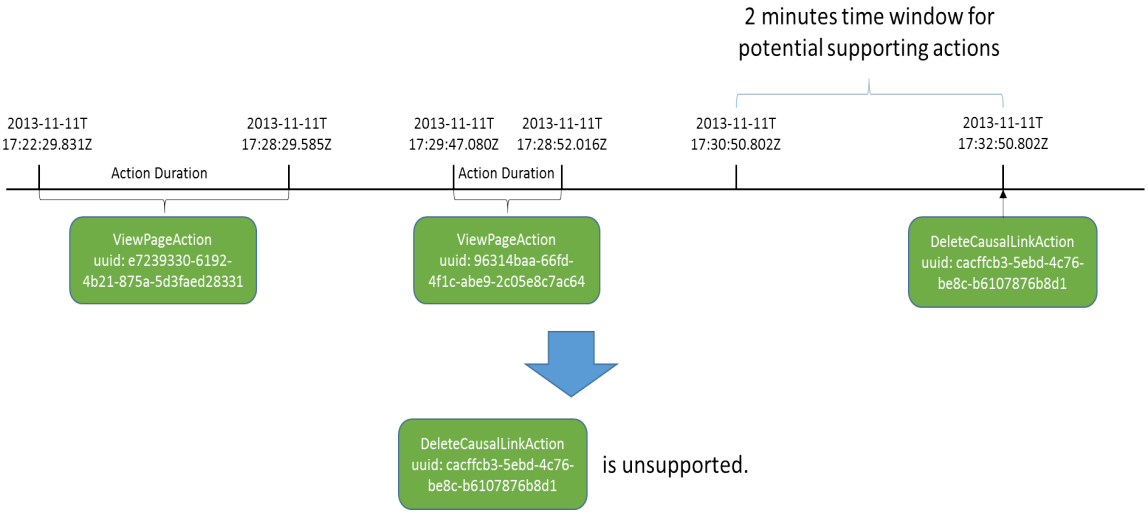


Figure 4.5: Determine supporting feature of *DeleteCausalLinkAction* action.

In addition to the coherence relation, we also define the effectiveness of solution construction actions as:

Definition 2 (Effectiveness)**.** "In an OELE, a solution construction action **A** is effective if the change made by the solution construction action is correct according to the expert model predefined for the OELE."

Examples of effective solution construction actions in Betty's Brain are removing incorrect concept entities and causal links from the causal map, or adding correct concept entities and causal links to the causal map. In other words, effective solution construction

actions made by the students can improve their model to get closer to the expert model. It is the criterion we used to derive the students' modeling performance (e.g., causal map score in Betty's Brain) in OELEs [7, 54, 55].

In some of the previous work, the coherence relations are used to characterize self-regulated Learning behaviors in OELEs [7], and are used as the basis to understand the change of students' problem-solving behaviors over time [56]. We've also applied the coherence relations to perform unsupervised learning and separate out learners into groups based on their overall learning behaviors [54].

The support/effectiveness features associated with solution construction actions are very important if we want to track learning performance measure of any artificially generated/extended action sequences, that are used to learn the reinforced HMM models. However, the HMM representation is defined at a level of abstraction that doesn't take into consideration these features. So we applied the coherence relations derived from the original data, to determine the support/effectiveness features when generating/extending action sequences using reinforcement learning.

We collect the coherence relations (i.e., support measure) using the method described above from the original data set to include coherence relations between sets of actions (e.g., IA $\rightarrow$ SC and SA $\rightarrow$ SC in Betty's Brain). The action coherence values are used to decide the relations between pairs of actions that help to determine the rewards for actions selected by the reinforcement learning algorithm. For example, actions that lead to more coherent subsequent actions can be rewarded over other actions, and thus, be chosen for generating new or extending existing action sequences.

4.2.1 **Hidden Markov Model**

Hidden Markov Models, which were initially introduced in the late 1960s, have been widely used in temporal pattern recognition applications such as Speech recognition [57] and DNA Motif Discovery [58]. The Hidden Markov Model (HMM) is a statistical model

which can be considered as a Markov process with hidden states. Formally, an HMM is defined as a five-tuple [57]:

$$\lambda = \{\mathbf{A}, \mathbf{B}, \pi, N, M\}, \tag{4.1}$$

where ***A*** and ***B*** represent the state transition probability distribution and emission probability distribution matrices, respectively, π is the initial state probability distribution, N is the number of hidden states, M is the number of distinct observations in the data set. Figure 4.6 shows a generic HMM model, where *x(t)* represents the hidden states with state transitions represented as *x(t)* $\rightarrow$ *x(t+1)*. Associated with each hidden state *x(t)*, there is a random variable *y(t)*, that has an associated probability distribution over all distinct observation symbols.

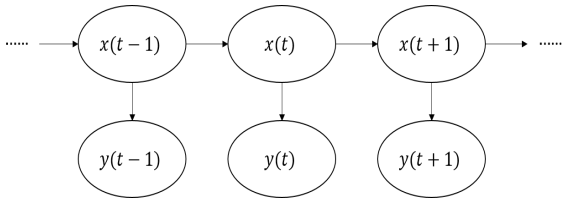


Figure 4.6: General Architecture of Hidden Markov Model

Figure 4.7 (a) presents a simple HMM example trained on two action sequences S_1 and S_2 with only 4 action types from Betty's Brain system (described in section 3.1). The HMM can also be represented by a state diagram [17] as shown in Figure 4.7 (b). Although not explicitly shown in the action sequences, the hidden states h_1 and h_2 can be interpreted, based on expert knowledge, as Information Acquisition state (searching for and reading resources) and Solution Construction state (editing concept entities and causal links), respectively.

For a more realistic example, such as the Betty's Brain environment, the students' model building tasks require about 26 action types, and the resulting HMM becomes very complex and hard to interpret if we use each individual action as a separate observation.

Sequence S_1: $search; read; search; read; concedit; linkedit$
Sequence S_2: $read; read; concedit; linkedit$

The HMM $\lambda = \{A, B, \pi, N, M\}$ can be represented as:

A:

	h_1	h_2
h_1	0.67	0.33
h_2	0	1.0

,

B_{h_1}:

$p(read)$	0.67
$p(search)$	0.33
$p(concedit)$	0
$p(linkedit)$	0

,

B_{h_2}:

$p(read)$	0
$p(search)$	0
$p(concedit)$	0.5
$p(linkedit)$	0.5

,

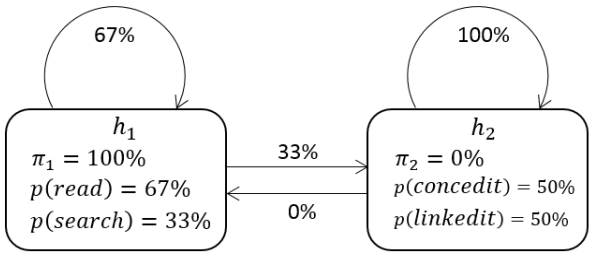
π:

h_1	1
h_2	0

,

N = 2, and M = 4

(a) HMM derived from the two action sequences



(b) State diagram of the HMM

Figure 4.7: Simple HMM example.

Therefore, we apply aggregation methods to combine sets of actions into a common descriptor. For example, we aggregate *linkedit* actions and *concedit* actions into a single *mapedit* action category.

Based on the different probability distributions for each observation (action), the hidden states are labeled by the primary actions associated with that state. The transitions between states capture the evolution of learning activities across time, e.g., transitions from information acquisition phase to solution construction and refinement phase.

Another approach that we have developed for characterizing learners' behaviors, sequence mining algorithms [59, 60] find patterns represented as subsequences of frequently occurring actions that capture snapshots of students' learning behaviors. On the other hand, the HMMs provide a generative and comprehensive model of the students learning behav-

iors across time. Furthermore, since students typically work on a modeling task for 3-4 days, it is reasonable to believe that students behaviors evolve as their learning improves, and it may fluctuate under different circumstances (e.g., when students are confused and stuck versus when they are making good progress). Therefore, HMMs, which capture students learning behaviors across time, are a more expressive temporal model of students' behaviors.

4.2.2 **Deriving Hidden Markov Models**

For deriving HMMs, the Baum-Welch re-estimation procedure [57] is applied to adjust model parameters($\mathbf{A}$, $\mathbf{B}$, π) to generate local maximal of the likelihood of action/observation sequences given the model λ. This re-estimation procedure implements the EM (Expectation-Maximization) algorithm, which starts with random parameters($\mathbf{A}$, $\mathbf{B}$, π) and iteratively updates them based on training data. This procedure works only when the number of hidden states N is known. However, N is often unknown in learner modeling because the students' behavior characteristics are unknown. In our work, we have adapted the Bayesian Information Criterion (BIC) to determine the best number of hidden states for HMM [61]. For different number of hidden states, the BIC value is computed by:

$$BIC = 2 \cdot \ln \hat{L} - k \cdot \ln(n) \tag{4.2}$$

where

- $\hat{L}$ is the maximized value of the likelihood function of the model λ, i.e., $\hat{L} = P(O|\hat{\lambda})$, where $\hat{\lambda}$ is the HMM model with maximized likelihood function.
- n is the number of observation sequences
- k is the number of free parameters to be estimated which we configured to be a polynomial function of N and M as a measure of model complexity.

So the first term $\ln\hat{L}$ is likely to promote a larger and more detailed model, e.g., a model with more hidden states, whereas the second term $k \cdot \ln(n)$ adds a penalty to complex models. Essentially, the BIC measure provides a trade-off between accuracy and complexity of the HMM model and tries to find models that are not overly complex while also maintaining sufficient accuracy. The refined HMM learner can be described by algorithm 1 (n_s is the max number of hidden states allowed).

Algorithm 1 ***Refined HMM Learner***

Given dataset $\mathscr{D}$
for $i := 1$ **to** n_s **do**
  $N_i \leftarrow i$
  Generate $\lambda_i = \{N_i, M, \mathbf{A}, \mathbf{B}, \pi\}$ with random $\mathbf{A}, \mathbf{B}$ and π
  $\bar{\lambda}_i \leftarrow$ ***Baum-Welch Learner***$(\lambda_i, \mathscr{D})$ [57]
  $bic_i \leftarrow$ ***BIC Computation***$(\bar{\lambda}_i, \mathscr{D})$ use formula (4.2) [61]
end for
return $\bar{\lambda}_i = \{N_i, M, \bar{\mathbf{A}}, \bar{\mathbf{B}}, \bar{\pi}\}$ where bic_i is maxmized

In our work, we choose the number of hidden states where the BIC value starts to level off. Because the Baum-Welch re-estimation procedure may converge to local optimal after random parameters' initialization, one can employ the *Baum-Welch Learner* multiple times in order to get potentially better results.

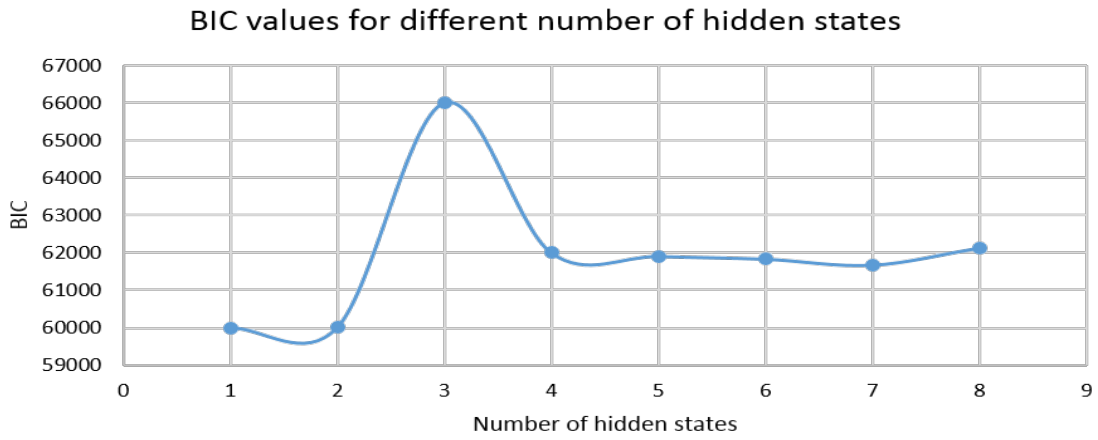


Figure 4.8: Bayesian Information Criterion (BIC) values calculated for different numbers of hidden states. The BIC values were re-scaled to fit the window.

For example, we can apply this algorithm on a set of 28 students' action sequences from

a Betty's Brain study and derive the plots of BIC values for different numbers of hidden states as shown in Figure 4.8. According to the plot, we choose 3 as the best number of hidden states because the BIC value reaches its max value with 3 hidden states. Using this result, we can then apply the Baum-Welch Learner to generate a three-states HMM which is shown in Figure 4.9.

$\boldsymbol{\pi}$:

π_1	π_2	π_3
1.0	0	0

$\boldsymbol{A}$:

	h_1	h_2	h_3
h_1	0.69	0.27	0.04
h_2	0.22	0.76	0.02
h_3	0.04	0.08	0.88

$\boldsymbol{B}$:

		h_1	h_2	h_3
IA	$p(read)$	0.46	0.12	0.03
	$p(search)$	0.02	0	0
	$p(notes)$	0.06	0.01	0.02
SC	$p(mapedit)$	0.30	0.52	0.04
	$p(clmark)$	0.02	0.13	0
SA	$p(quiz)$	0.07	0.16	0.22
	$p(query)$	0.01	0.02	0.01
	$p(expl)$	0.06	0.04	0.68

Figure 4.9: Example HMM for the set of 28 students' action sequences. The HMM is represented by vector of initial probabilities π, matrix of state transition probabilities **A**, and matrix of emission probabilities **B**.

4.2.3 **Sequence Clustering using HMMs**

As discussed earlier in Section 3.2, students exhibit a variety of behaviors, and their behaviors evolve when they are learning in an OELE environment. So a single HMM is insufficient to model students' learning behaviors comprehensively. As a result, we apply clustering method to derive groups of students who show similar learning behaviors as represented by their action sequences. And in order to maintain the consistency between

the clustering results and the model representation, we apply the HMM clustering algorithm which uses the differences in the HMMs as the basis for separation [26].

In this methods, a variation of the K-Means clustering algorithm is applied, which instead of computing a traditional centroid, applies the *Refined HMM Learner* algorithm (illustrated in Algorithm 1) and uses the resulting HMM model as the surrogate for the cluster centroid. Then, we employ the log-likelihood function as a measure of the cohesiveness of a cluster.

For each action sequence O in the data set, a sequence-to-HMM likelihood is given by $P(O|\lambda)$ which measures the probability of sequence O be generated by the HMM model λ. $P(O|\lambda)$ can be computed by using Forward-Backward Procedure [57]. In this procedure, however, the forward variable which stores the probability of the partial observation sequence is in a product form of state transition probabilities and emission probabilities, where each probability term is less than or significantly less than 1. For sufficiently long data sequences, the dynamic range of the forward variable will exceed the precision range of any machine, resulting in an underflow in computing the sequence-to-HMM likelihood. A scaling method [57] is required to normalize the forward variable and computes sequence-to-HMM log-likelihood.

We use the Partition Mutual Information(PMI) [62] to determine the optimal number of clusters [26]. The clusters are most separated when the PMI value is maximized. And in *Baum-Welch learner*, ($\mathbf{A}$, $\mathbf{B}$, π) is modified to maximize the likelihood function, which in other words is maximizing homogeneity within each cluster. Thus, finding the number of clusters, which results in highest PMI potentially gives an optimal clusters' configuration. Partition Mutual Information is given by:

$$PMI = \frac{\sum_{j=1}^{J} \sum_{i=1}^{n_j} MI_{i,j}}{J}, \tag{4.3}$$

where J is the number of clusters, n_j is the number of data sequences in cluster j and $MI_{i,j}$

the average mutual information between observation sequence O_i and a HMM models λ_j is the logarithmic value of the posterior probability of model λ_j, trained on data O_i:

$$\begin{aligned} MI_{i,j} &= \log P(\lambda_j|O_i) = \log \frac{P(O_i|\lambda_j)P(\lambda_j)}{P(O_i)} \\ &= \log \frac{P(O_i|\lambda_j)P(\lambda_j)}{\sum_{j=1}^{J} P(O_i|\lambda_j)P(\lambda_j)}, \end{aligned} \tag{4.4}$$

where $P(\lambda_j)$ is the prior probability. And as discussed above, the dynamic range of the sequence-to-HMM likelihood $P(O_i|\lambda_j)$ may be intractable, we need to find alternative ways to compute $MI_{i,j}$. We apply the logarithmic identity:

$$\log_b \sum_{i=0}^{N} a_i = \log_b a_0 + \log_b \left(1 + \sum_{i=1}^{N} b^{(\log_b a_i - \log_b a_0)} \right), \tag{4.5}$$

where $a_0 > a_1 > \ldots > a_N$ are sorted in descending order. We compute $MI_{i,j}$ as:

$$MI_{i,j} = \log P(O_i|\lambda_j)P(\lambda_j) - \log \sum_{j=1}^{J} P(O_i|\lambda_j)P(\lambda_j), \tag{4.6}$$

and

$$\begin{aligned} \log \sum_{j=1}^{J} P(O_i|\lambda_j)P(\lambda_j) &= \log P(O_i|\lambda_0)P(\lambda_0) \\ + \log [1 + \sum_{j=1}^{J} e^{\log P(O_i|\lambda_j)P(\lambda_j) - \log P(O_i|\lambda_0)P(\lambda_0)}], \end{aligned} \tag{4.7}$$

where $\log P(O_i|\lambda_0)P(\lambda_0) > \log P(O_i|\lambda_1)P(\lambda_1) > \ldots > \log P(O_i|\lambda_J)P(\lambda_J)$ are sorted in descending order. The HMM clustering algorithm can be described by Algorithms 2 and 3.

Algorithm 2 ***HMM Clustering***

Given dataset $\mathscr{D}$
for $i := 2$ **to** n_c **do**
  $\{\lambda_1, \lambda_2, ..., \lambda_i\} \leftarrow$ ***Standalone HMM Clustering***($\mathscr{D}$, i)
  Compute PMI_i according to formula (4.3)(4.6)(4.7)
end for
$n_{best} \leftarrow i$ where PMI_i is maxmized
return ***Standalone HMM Clustering***($\mathscr{D}$, n_{best})

Algorithm 3 ***Standalone HMM Clustering***

Given dataset $\mathscr{D}$ and number of clusters n
Randomly distribute $\mathscr{D}$ into n clusters, $\mathscr{C}_1$,$\mathscr{C}_2$,...,$\mathscr{C}_n$.
repeat
  for each cluster $\mathscr{C}_i$ ($i = 1, 2, ..., n$) **do**
    $\lambda_i \leftarrow$ ***Rrefined HMM Learner***($\mathscr{C}_i$)
  end for
  for each sequence O in $\mathscr{D}$ **do**
    Redistribute O into cluster $\mathscr{C}_i$ where $P(O|\lambda_i)$ is maximized
  end for
until Convergence
return $\{\lambda_1, \lambda_2, ..., \lambda_n\}$

As we can see from the algorithms, the ***Refined HMM Learner*** will be performed multiple times till convergence. In our work, we had to carefully set the number of repetitions for the ***Baum-Welch Learner*** to constrain the run time of the HMM clustering procedure. To achieve this, we applied a convergence criterion that terminated when the likelihood increment associated with the clusters in a new configuration was smaller than a pre-specified threshold. Figure 4.10 shows an example of computing PMI values to determine the number of clusters for a data set consists of 98 students' action sequences from a Betty's Brain study.

We then perform the HMM clustering algorithm on this data set, which produced three HMMs shown in Figure 4.11 (cluster 1), Figure 4.12 (cluster 2), and Figure 4.13 (cluster 3). We can also represent the HMMs in state diagram as shown in Figure 4.14.

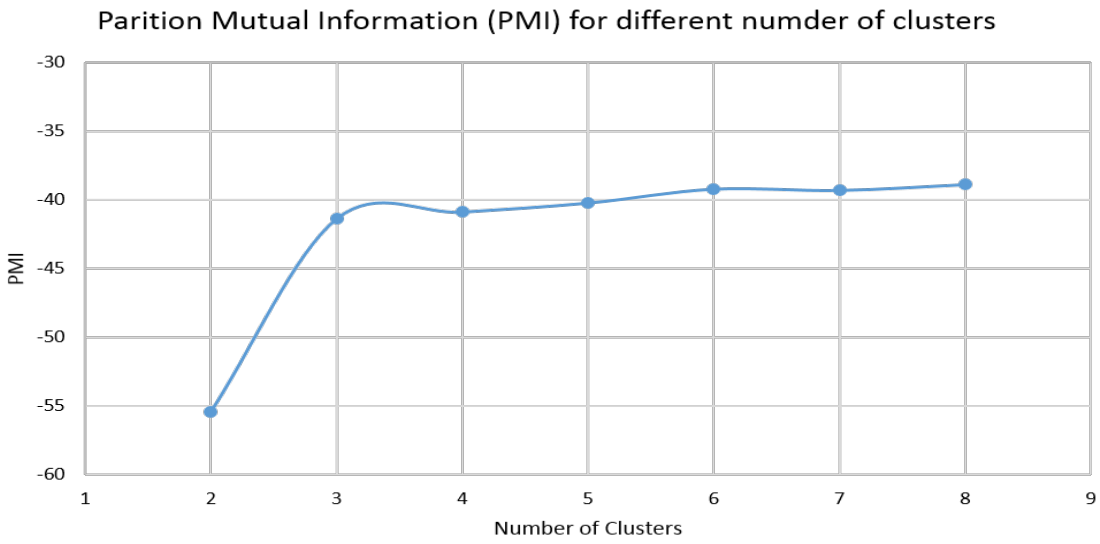


Figure 4.10: Partition Mutual Information values calculated for the Betty's Brain data

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4
0.77	0.23	0	0

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4
h_1	0.65	0.35	0	0
h_2	0.41	0.31	0.28	0
h_3	0.10	0.18	0.34	0.38
h_4	0.05	0.05	0.07	0.83

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4
IA	$p(read)$	0.57	0.39	0.15	0.06
	$p(search)$	0.07	0	0	0
	$p(notes)$	0.16	0.08	0.12	0.03
SC	$p(mapedit)$	0.20	0.45	0.21	0.06
	$p(clmark)$	0	0.03	0.15	0.03
SA	$p(quiz)$	0	0.03	0.26	0.33
	$p(query)$	0	0.01	0.06	0.02
	$p(expl)$	0	0.01	0.05	0.47

Figure 4.11: The HMM of Cluster 1 (22 students) derived for the Betty's Brain study

4.3 Applying Reinforcement Learning to Generate Updated HMMs

Generating HMMs that accurately represent students' behaviors requires large amounts of data (action sequences) to converge to their true behaviors. However, given that we can

$\boldsymbol{\pi}$:

π_1	π_2	π_3
1.0	0	0

$\boldsymbol{A}$:

	h_1	h_2	h_3
h_1	0.69	0.27	0.04
h_2	0.22	0.76	0.02
h_3	0.04	0.08	0.88

$\boldsymbol{B}$:

		h_1	h_2	h_3
IA	$p(read)$	0.46	0.12	0.03
	$p(search)$	0.02	0	0
	$p(notes)$	0.06	0.01	0.02
SC	$p(mapedit)$	0.30	0.52	0.04
	$p(clmark)$	0.02	0.13	0
SA	$p(quiz)$	0.07	0.16	0.22
	$p(query)$	0.01	0.02	0.01
	$p(expl)$	0.06	0.04	0.68

Figure 4.12: The HMM of Cluster 2 (28 students) derived for the Betty's Brain study

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4
0.52	0.31	0.18	0

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4
h_1	0.78	0.11	0.08	0.03
h_2	0.35	0.36	0.27	0.02
h_3	0.27	0.22	0.14	0.37
h_4	0.02	0.02	0.13	0.82

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4
IA	$p(read)$	0.32	0.41	0.29	0.03
	$p(search)$	0.04	0.01	0	0
	$p(notes)$	0	0.02	0.03	0
SC	$p(mapedit)$	0.58	0.44	0.32	0.06
	$p(clmark)$	0	0	0.01	0.06
SA	$p(quiz)$	0.06	0.10	0.26	0.31
	$p(query)$	0	0.01	0.02	0.01
	$p(expl)$	0	0.01	0.07	0.53

Figure 4.13: The HMM of Cluster 3 (48 students) derived for the Betty's Brain study

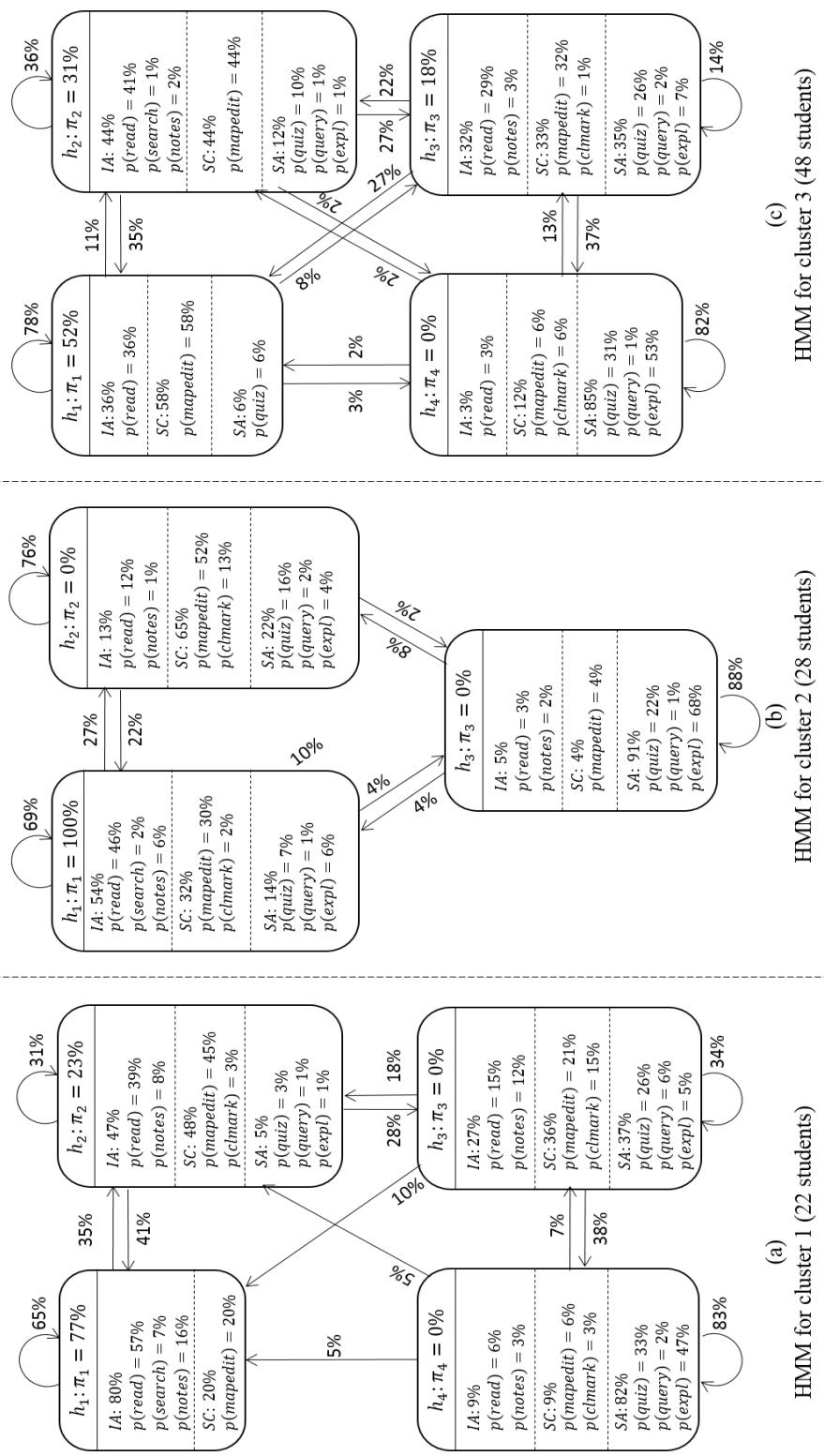


Figure 4.14: HMMs for the three clusters

only collect limited amounts of student data from our classroom studies, we suffer from the data impoverishment problem. To address this problem, we have adopted a novel reinforcement learning method combined with Monte Carlo Tree Search (MCTS) to generate additional data that starting from initial HMM models of students' learning behavior with the goal of learning more robust and accurate models.

The extended data generates additional/extended action sequences that (1) better represent students' learning behaviors, and (2) optimize the sequences based on specific learning performance measures. Section 4.3.1 delves deeper into our approach for reinforcement learning and how the two purposes are achieved. Section 4.3.2 presents the technical definition and details of the MCTS approaches to derive reward functions for reinforcement learning. Section 4.3.3 presents the application of these combined techniques in generating artificial action sequences. The generated action sequences are combined with original students' data to learn new HMMs.

4.3.1 **Reinforcement Learning**

The reinforcement learning procedure can be modeled as a Markov Decision Process [19, 63], which consists of:

- A set of environment states, S_e, and agent states, S_a.
- A set of available agent actions, A.
- Functions that define state transitions from s to s_{new} under action a.
- The reward generated after the agent takes an action a.

Figure 4.15 shows the reinforcement learning scheme using the Betty's Brain learning environment as an example. As we can see from the figure, a virtual agent can interact with the environment by taking actions. This agent corresponds to a student who interacts with the Betty's Brain System. At each time t, the expected reward of all available actions can

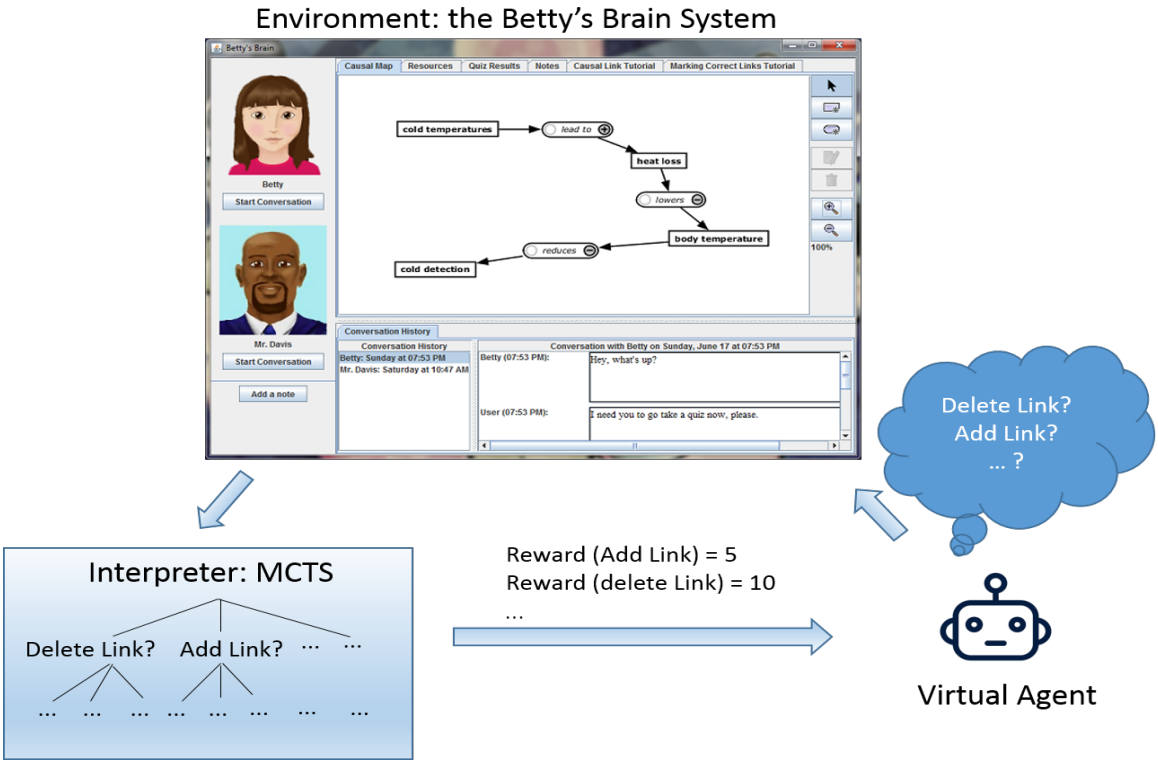


Figure 4.15: The scheme of reinforcement learning using Betty's Brain environment as an example

be derived by the interpreter (e.g., MCTS), and the RL approach is aimed to choose the one that maximizes a long-term reward. So the agent should be able to take into consideration the long-term consequences of the action to be chosen, so as to maximize the overall reward.

This reinforcement learning scheme is suitable for generating additional or extending existing action sequences of students learning behaviors in OELEs, in a way that the agent can be equated to a virtual student whose action sequences may match the students on the OELE system. Because we use HMMs to model agents' learning behaviors, the agent states S_a correspond to the hidden states in the corresponding HMM. And, the environment states S_e can be determined by a set of constraints defined in the OELE system. For example, in the Betty's Brain system, there are constraints such as "no *linkadd* action can be taken before at least two concept entities exist in the causal map" (mark as constraint C_1)

and "no *linkrem* or *linkchg* action, which is enforced till there are at least one causal link has been added to the causal map" (mark as constraint C_2). An environment state can be determined by concatenating the binary values of all constraints' conditional clauses, and the size of environment states set equals to 2^n where n is the number of binary constraints. A conditional clause is assigned a value of 1 when it holds, and 0 otherwise. Suppose there are only two constraints and the conditional clause of C_1 holds (i.e., at least two concept entities exist in the causal map) while the conditional clause of C_2 does not hold (i.e., no link exists in the causal map), the corresponding binary values are 1 and 0, respectively. Then, the environment state is set to 10, and this state enables *linkadd* actions while *linkchg* and *linkrem* are disabled.

So the available actions that the agent can take at time t are determined by both agent states and environment states. The agent states which are represented by hidden states in HMM, determine the available actions A_1 according to learner's behavior, while the environment states determine available actions A_2 that are enabled by the system. The set of all available actions A_t at time t is the intersection of A_1 and A_2.

The reinforcement learning is applied to generate action sequences to satisfy two major modeling purposes in this work. They are:

- (**Purpose I**): generate additional action sequences that simulate the real students' learning behaviors so that the reinforced HMMs can have higher accuracy for classifying students into the correct groups, and

- (**Purpose II**): extending existing action sequences to optimize specific learning performance measure so that the reinforced HMMs can form the basis for generating appropriate scaffolding .

The interpreter for reinforcement learning (Figure 4.15) should be designed to derive the reward values for the virtual agent's actions for the two different purposes (i.e., **Purpose I** and **Purpose II**). For example, an action that is frequently taken by the students or an

action that may result in similar performance measure can be rewarded for purpose **I**; and an action that results in an improvement of performance measure can have higher reward compared to other available actions for purpose **II**. For deriving aggregated reward of long-term consequences for taking actions, we apply Monte Carlo Tree Search (MCTS) as the interpreter, which produces the projected reward after taking an action through the use of simulations (i.e., multiple fast rollouts).

4.3.2 Monte Carlo Tree Search

As a heuristic search algorithm, MCTS has been applied widely to solve decision problems, e.g., in developing AI techniques to model game play. The recent success of AlphaGo [16], which employed the MCTS algorithm and reinforcement learning further supports the power of this technique in solving practical problems. Although learner modeling is different from games, we can adopt the MCTS methodology for the decision making processes, e.g., deciding the next best action(s) and studying the effectiveness of feedback provided to students when they encounter problems in their learning in OELEs.

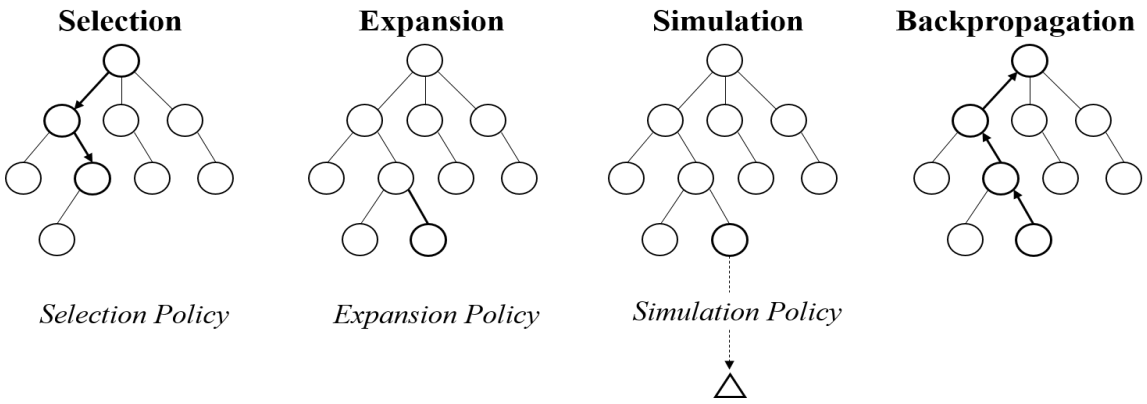


Figure 4.16: Four steps for each iteration of MCTS algorithm

Monte Carlo Tree Search basically performs a recursive search on a tree to make decisions on what is the next best node to choose based on a pre-defined criterion. Each iteration of its recursive search consists of four steps [64]:

1. ***Selection***: Starting from root, recursively apply a *selection policy* until the algorithm reaches the most promising expanded node.

2. ***Expansion***: If a termination condition is not reached, generate a new node according to an *expansion policy*.

3. ***Simulation***: Apply a *simulation policy* that results in a fast "rollout" from current node to a game's end state.

4. ***Backpropagation***: Use simulation results to update parent node statistics, e.g., the likelihood of win for a parent node.

Figure 4.16 shows how the MCTS is applied in a general way. Most MCTS implementations apply *selection policy* by applying the *Upper Confidence Bound applied to trees* (UCT) [65] criterion, which creates a balance between exploitation and exploration. *Simulation policy* in the original MCTS is implemented as a purely random sampling on all possible nodes. After a number of simulations, we can determine the reward for actions (children of the root) accordingly and apply it for extending action sequences in reinforcement learning.

4.3.3 **Reinforcement Learning using MCTS for generating action sequences**

At each iteration of the reinforcement learning approach (shown in Figure 4.15), we build the search tree and apply MCTS to pick the best node (that is generated by the largest number of simulations). This corresponds to an action that is added to the tail of the current action sequence. It can either start with an empty action sequence to generate additional action sequences, or start with an existing student's action sequence to generate an extended action sequence.

Figure 4.17 shows a simple example for generating artificial action sequences by applying reinforcement learning and MCTS. The MCTS basically acts as the interpreter for

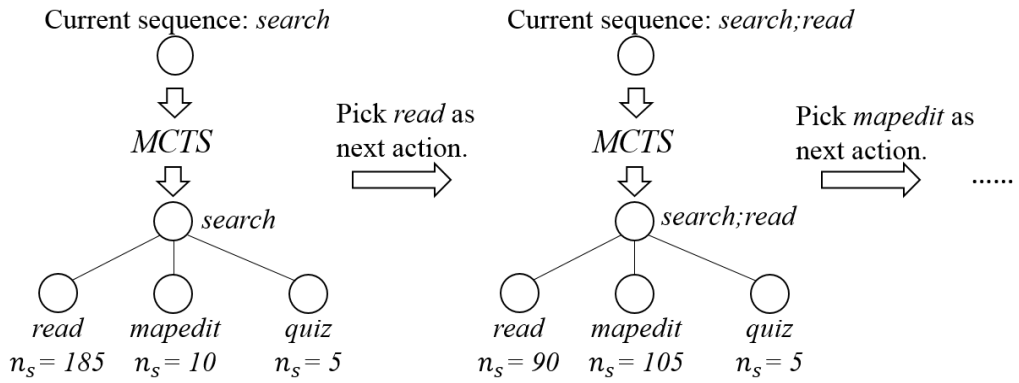


Figure 4.17: Simple example of applying RL and MCTS for generating action sequence. n_s is the number of simulations performed during MCTS.

the reinforcement learning algorithm, and does not derive immediate rewards for available actions, but rather provides the long-term reward of taking specific actions. As we can see from Figure 4.17, the *read* and the *mapedit* actions are chosen and appended to the current action sequence for providing the highest reward (i.e., number of simulations) in the two reinforcement learning iterations. In this example, the action sequence has been grown from "*search*" to "*search*;*read*;*mapedit*". If we continue this process, we can generate an action sequence of required length (e.g., 500 actions) or an action sequence that achieves expected performance (e.g., highest map score).

In the process as illustrated in Section 3.2, we repeatedly generate or extend action sequences that maximize a specified reward function, and add them to the previously generated data. The reinforced data set is used to learn a refined version of the HMMs, and the refined HMMs are called reinforced models. So, for the two different modeling purposes (i.e., **Purpose I** and **Purpose II**), we can generate two reinforced models, namely a *reinforced classification model* and a *reinforced scaffolding model*. In order to satisfy these two modeling purposes, the artificial action sequences are generated by configuring the three search policies of MCTS shown in Figure 4.16 (i.e., *selection policy*, *expansion policy* and *simulation policy*). They are summarized as follows:

Selection

In most MCTS implementations for designing AI systems, the Upper Confidence bounds applied to Trees (UCT) is applied as the reward function for node selection:

$$UCT = \frac{w_i}{n_i} + c\sqrt{\frac{\ln t}{n_i}}, \tag{4.8}$$

where n_i is the number of simulations performed after adding the ith action; c is the exploration parameter with a typically chosen empirical value of $\sqrt{2}$; t is the total number of simulation runs for the parent node, which is equal to the sum of all the n_i values; w_i is the sum of wins (1's) for all simulations after adding the ith action.

The UCT criterion is known to balance exploitation and exploration for running simulations [66]. The left term $\frac{w_i}{n_i}$, which calculates the win ratio of all simulations is the exploitation measure for all the simulations performed. It is high for actions with high average win ratio. On the other hand, the right term $c\sqrt{\frac{\ln t}{n_i}}$ is the exploration term that is high for actions with fewer simulations.

We adopt a similar reward function in this work. However, since students can be assigned partial points based on their performance in OELEs, w_i can not be computed by the sum of wins (1's). Therefore, we derive the w_i by the sum of a value v_w which is in the range of [0, 1]. The value of v_w measures how successful a single simulation matches the modeling goal we targeted. In this work, we have developed two different schemes for computing the v_w value for the UCT functions according to the two reinforcement learning purposes. They are:

(1) *Reinforced classification model*. We compute v_w by:

$$v_w = av_l + (1-a)v_p \tag{4.9}$$

where

- v_l is the normalized log-likelihood value of the action sequence generated by the corresponding HMM. This is essentially a homogeneity measure of generated action sequences to the original HMM. For example, a simulation which generates an action sequence with the highest log-likelihood value has its $v_l = 1$.

- v_p is the normalized value of the similarity between the **simulation** performance and the **expected** performance measure. For example, a simulation has $v_p = 1$ when its resulting action sequence achieves the exact same performance measure with the expected value.

- a is a bias factor in the range of [0, 1]. For this reinforced model, we set $a = 0.5$ so that both terms are equally important.

So v_w results in the range of [0, 1], and the sum of v_w for all performed simulations produces the value for w_i. Using this measure, we can generate action sequences that are aimed at simulating students' behaviors based on existing HMMs and the corresponding performance, measured by the model score achieved.

(2) *Reinforced scaffolding model*. We compute v_w using the same equation as (4.9) but configure the v_p as the normalized value of the similarity between the **simulation** performance and the **maximum** performance measure. So, a simulation has $v_p = 1$ when the its resulting action sequence achieves the best performance measure (e.g., a model score of 15 in a Betty's Brain study about "climate change"). We set $a = 0.2$ to bias toward better performance (i.e., lower distance from optimal performance measure) over the homogeneity measure.

This allows MCTS to better utilize the coherence relations to generate action sequences with more effective solution construction actions and the resulting HMMs can evolve and favor the use of more coherent actions. Furthermore, this approach assumes the students are learning, and, therefore, their map scores get better with time. However, we define an upper limit on the length of the action sequences that, not all of them can achieve the

maximum map score.

Expansion and ***Simulation***

Both expansion and simulation in MCTS are constrained by the available actions that are determined by the environment state and the agent state of reinforcement learning as described in Section 4.3.1. The MCTS will only expand to nodes that are in the set of corresponding available actions, and the simulation will be performed through the fast rollout according to available actions at each step. With these simulation and expansion policies, we can always generate action sequences that fit the HMM within a specified variance range despite the different cases involved in the selection policy. Actions in the generated sequences will never violate the system constraints.

Algorithm 4 outlines the sequence generating procedure. And the overall reinforcement learning algorithm is summarized as Algorithm 5.

Algorithm 4 ***Reinforcement Learning for Sequence Generation***

Given a HMM model λ_i and corresponding dataset $\mathscr{D}_i$
$Sequence \leftarrow \{\}$
$root \leftarrow \{\}$
repeat
  Perform MCTS (described in section 3 and 4)
  $nextBest \leftarrow root.child$ with most simulations
  $Sequence \leftarrow Sequence \cup nextBest$
  $root \leftarrow nextBest$;
until Reaching desired length
return *Sequence*

Algorithm 5 ***Learning Reinforced HMMs***

Given dataset $\mathscr{D}$, clustered HMM model $\{\lambda_1, \lambda_2, ..., \lambda_n\}$ and corresponding data distribution $\mathscr{D}_1, \mathscr{D}_2, ..., \mathscr{D}_n$
for each cluster λ_i and $\mathscr{D}_i$ $(i = 1, 2, ..., n)$ **do**
  repeat
    $\mathscr{D}_i \leftarrow \mathscr{D}_i \cup$ ***Reinforcement Learning for Sequence Generation***$(\lambda_i, \mathscr{D}_i)$
    $\lambda_i \leftarrow$ ***Refined HMM Learner***$(\mathscr{D}_i)$
  until Sample size of $\mathscr{D}_i$ is satisfiable
end for

4.4 Summary

In this chapter, we have presented details of our learner modeling approach. We started by discussing the action-view representation, which can handle the data heterogeneity problem between different OELEs. We also presented coherence relations, which are used to determine the effectiveness of actions when generating/extending action sequences. Then, we demonstrate the learner modeling approach and described each component of the modeling approach in detail. Specifically, we presented a definition and our implementation for learning HMMs. In addition, we discussed the clustering algorithm based on the HMM representation. The last section provided a detailed description of reinforcement learning and MCTS for generating artificial action sequences to aid the learner modeling process.

We have now addressed aspects of the *data collection* challenges with the *action-view* representation, and addressed the *modeling* and *machine learning* challenges in designing our learner modeling approach that combines multiple techniques (i.e., HMM and HMM clustering, MCTS, Reinforcement Learning, and coherence relations). Compared to many other learner modeling approaches, the major contribution of our work is the combined use of MCTS and reinforcement learning for generating simulated data that can enrich and extend the initial data set, and help learn more complete and robust HMMs.

Our next task is to address the other challenge (i.e., *verification and validation* challenges). This is presented in detail in Chapter 5. We demonstrate the results of experiments run with two different OELEs developed by our group: (1) the Betty's Brain system, and (2) the CTSiM environment. We perform cross-validations on reinforced classification HMMs to show the improvements in classification accuracy. We also compare the reinforced scaffolding HMMs against the original HMMs, and use them as the basis for generating appropriate adaptive scaffolds.

Chapter 5

Experiments

In this chapter, we run experiments to demonstrate the effectiveness of our learner modeling method using data collected from classroom studies with the Betty's Brain and the Computational Thinking using Simulation and Modeling (CTSiM) environments. As discussed earlier, Betty's Brain is a learning-by-teaching environment, where students learn a scientific phenomenon by creating a causal map to teach Betty [47]. On the other hand, students learn about science topics by building simulation models in the CTSiM environment using a visual domain-specific block-structured language [1]. Through the two experimental studies, we provide evidence that our learner modeling methods using reinforcement learning are generalizable to different OELEs.

For each data set, we first briefly describe the learning environment and formally define the format of the data. Then we derive the HMMs from the original data and refine the model using the reinforcement learning methods discussed in Chapter 4, i.e., Reinforced Classification and Reinforced Scaffolding. For the reinforced classification HMMs, we run Leave-One-Out Cross Validation to show that the reinforced HMMs are better at categorizing students into correct groups. For the reinforced scaffolding HMMs, we perform empirical analyses on the results generated and compare them with the original HMMs to show the effectiveness of this method. Examples of adaptive scaffolds generated based on the reinforced models are also discussed.

5.1 Experiments with the Betty's Brain study

The first experiment uses the data collected from a Betty' Brain study run with 98 6^{th} grade middle school students in a science classroom. The Betty's Brain system and its available actions are discussed in Section 3.1. As our first step, we applied the HMM

clustering algorithm to discover groups of students based on their action sequences.

5.1.1 HMM Clustering Results

As discussed earlier, we derived the optimum number of clusters by computing the partition mutual information (PMI) for different numbers of clusters as illustrated in Section 4.2.3. Figure 5.1 shows the plot of the PMI values for different numbers of clusters. As we can see, the PMI value started to level off when the number of clusters reached 3. Considering the fact that the model can be over complex when the number of clusters increases, we chose 3 as the number of clusters for this experiment.

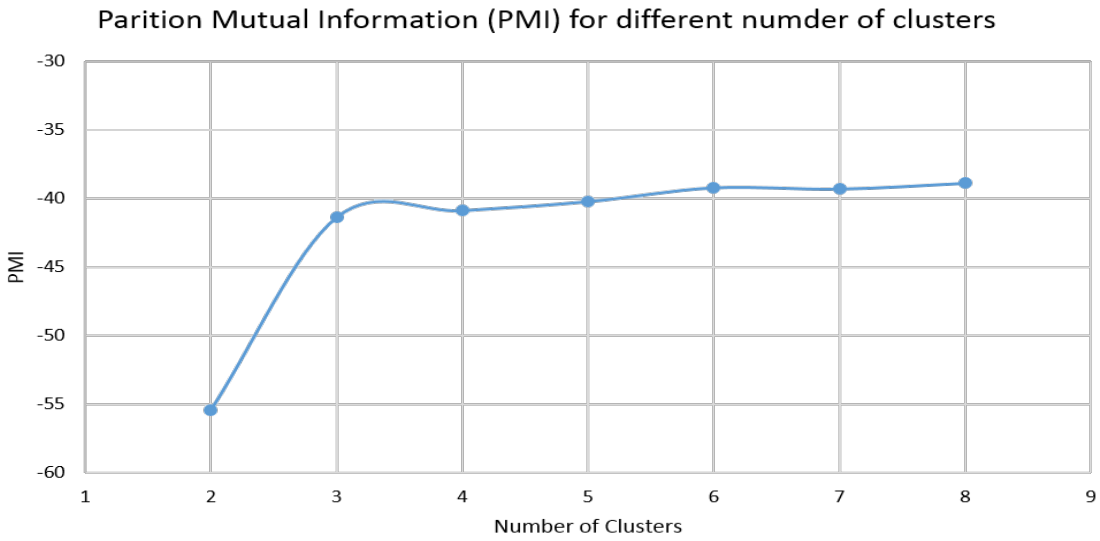


Figure 5.1: Partition Mutual Information values calculated for the Betty's Brain data

Then, the HMM clustering algorithm was run to generate three clusters of action sequences that maximizes the PMI value as described in Section 4.2.3. For each of the three clusters, we generated an HMM that models the action sequences within the cluster. As discussed earlier in Section 4.2.2, we compute the Bayesian Information Criterion (BIC) to determine the best number of hidden states for learning the HMMs. Figures 5.2 shows the plots of the BIC values computed for the three derived clusters. As we can see:

- Cluster 1 and 3 achieved best BIC value with 4 hidden states.

- Cluster 2 achieved best BIC value with 3 hidden states.

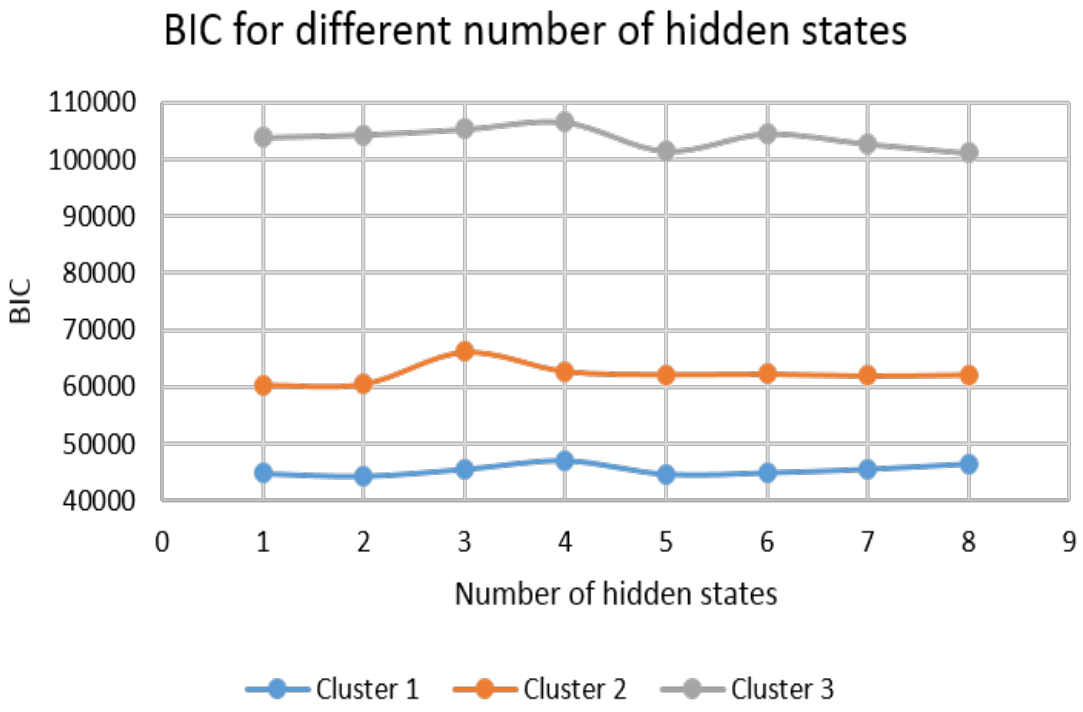


Figure 5.2: Bayesian Information Criterion (BIC) values calculated for the three clusters derived from HMM clustering on data collected from a Betty's Brain study. The BIC values were re-scaled to fit the window.

Using these results, the final HMMs derived for the three clusters are shown in Figure 5.3 (cluster 1), Figure 5.4 (cluster 2), and Figure 5.5 (cluster 3). We also show the state diagrams for the three clusters in Figure 5.6. For each of the three HMMs:

- The hidden states are denoted as $h_1, h_2, ..., h_n$, where n is the number of hidden states.
- The initial states are denoted as $\pi = \{\pi_1, \pi_2, ..., \pi_n\}$, where n is the number of hidden states. The value of π_i is shown in vector π.For example, the initial probability of h_1 in the HMM for cluster 1 is 77%.
- The state transition probabilities are shown in matrix **A**. The value in an entry $\{i, j\}$ is the state transition probability from hidden state h_i to h_j. So the ith row shows the

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4
0.77	0.23	0	0

***A*:**

	h_1	h_2	h_3	h_4
h_1	0.65	0.35	0	0
h_2	0.41	0.31	0.28	0
h_3	0.10	0.18	0.34	0.38
h_4	0.05	0.05	0.07	0.83

***B*:**

		h_1	h_2	h_3	h_4
IA	$p(read)$	0.57	0.39	0.15	0.06
	$p(search)$	0.07	0	0	0
	$p(notes)$	0.16	0.08	0.12	0.03
SC	$p(mapedit)$	0.20	0.45	0.21	0.06
	$p(clmark)$	0	0.03	0.15	0.03
SA	$p(quiz)$	0	0.03	0.26	0.33
	$p(query)$	0	0.01	0.06	0.02
	$p(expl)$	0	0.01	0.05	0.47

Figure 5.3: The HMM of Cluster 1 (22 students) derived for the Betty's Brain study

$\boldsymbol{\pi}$:

π_1	π_2	π_3
1.0	0	0

***A*:**

	h_1	h_2	h_3
h_1	0.69	0.27	0.04
h_2	0.22	0.76	0.02
h_3	0.04	0.08	0.88

***B*:**

		h_1	h_2	h_3
IA	$p(read)$	0.46	0.12	0.03
	$p(search)$	0.02	0	0
	$p(notes)$	0.06	0.01	0.02
SC	$p(mapedit)$	0.30	0.52	0.04
	$p(clmark)$	0.02	0.13	0
SA	$p(quiz)$	0.07	0.16	0.22
	$p(query)$	0.01	0.02	0.01
	$p(expl)$	0.06	0.04	0.68

Figure 5.4: The HMM of Cluster 2 (28 students) derived for the Betty's Brain study

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4
0.52	0.31	0.18	0

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4
h_1	0.78	0.11	0.08	0.03
h_2	0.35	0.36	0.27	0.02
h_3	0.27	0.22	0.14	0.37
h_4	0.02	0.02	0.13	0.82

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4
IA	$p(read)$	0.32	0.41	0.29	0.03
	$p(search)$	0.04	0.01	0	0
	$p(notes)$	0	0.02	0.03	0
SC	$p(mapedit)$	0.58	0.44	0.32	0.06
	$p(clmark)$	0	0	0.01	0.06
SA	$p(quiz)$	0.06	0.10	0.26	0.31
	$p(query)$	0	0.01	0.02	0.01
	$p(expl)$	0	0.01	0.07	0.53

Figure 5.5: The HMM of Cluster 3 (48 students) derived for the Betty's Brain study

probabilities of transitioning out of hidden state h_i, while the jth column shows the probabilities of transitioning into hidden state h_j. For example, the state transition probability from h_1 to h_2 in the HMM for cluster 1 (Figure 5.3) is 35%.

- The emission probabilities **B** within each hidden state are shown by the corresponding column in matrix **B** as the values of $p(a)$, which is the probability of observing action a in the corresponding hidden state. For example, the emission probability of the observation/action *read* in h_1 is 57% for cluster 1. In addition, all the actions are categorized into the three categories (i.e., Information Acquisition, Solution Construction, and Solution Assessment) as described earlier in Section 3.1.

In order to compare the HMMs between different clusters, we use some of the measures defined in our previous work [54, 60]. They are defined as:

- **IA effort**, which is the percentage of taking IA actions of all the actions performed by a student.

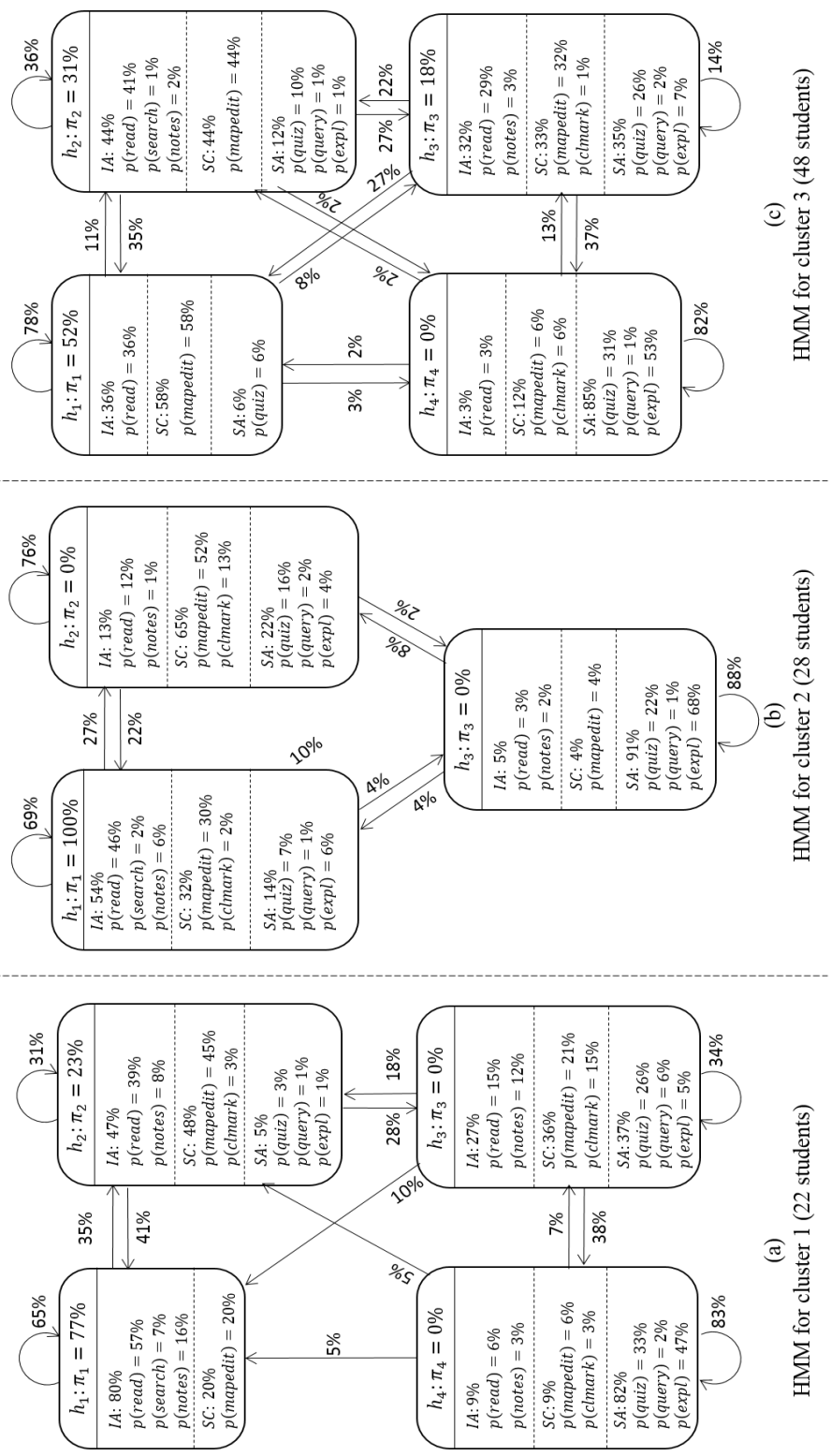


Figure 5.6: HMMs for the three clusters derived for the Betty's Brain study

- **Search Rate** , which is the average percentage of taking *search* actions of all the IA actions.

- **IA $\rightarrow$ SC transition**, which is the average percentage of IA actions that **support** later SC actions of all the IA actions performed by a student.

- **IA $\rightarrow$ SC effectiveness**, which is the average percentage of effective SC actions that are supported by IA actions.

- **SA $\rightarrow$ SC transition**, which is the average percentage of SA actions that **support** later SC actions of all the SA actions performed by a student.

- **SA $\rightarrow$ SC effectiveness**, which is the average percentage of effective SC actions that are supported by SA actions.

- **SC $\rightarrow$ IA transition**, which is the average percentage of SC actions that **support** later IA actions of all the SC actions performed by a student.

- **SC $\rightarrow$ SA transition**, which is the average percentage of SC actions that **support** later SA actions of all the SC actions performed by a student.

- $\overline{S_g}$, which is the average pre- and post-test score gain. It measures on average, how much the students within in a cluster have learned after the study (the higher, the better).

- $\overline{S_m}$, which is the average final causal map score. It measures how good their final causal maps are (the higher the better).

Note that all the transition measures correspond to the coherence framework that we have defined by Definition 1 in Section 4.1.1. The effectiveness of SC actions are defined by Definition 2 in Section 4.1.1. All the measures represent the percentage averaged across all students within the same cluster.

Table 5.1: Comparison of the three clusters of original data from Betty's Brain. Results for each measure are presented as mean (standard deviation).

	Cluster 1	Cluster 2	Cluster 3
IA effort %	42.5 (21.6)	20.8 (12.3)	24.4 (11.7)
Search rate %	5.6 (2.7)	0.3 (1.1)	3.1 (2.6)
IA $\rightarrow$ SC transitions %	31.5 (21.8)	15.9 (6.4)	22.0 (13.5)
IA $\rightarrow$ SC effectiveness %	69.5 (25.9)	63.8 (23.6)	66.7 (21.5)
SA $\rightarrow$ SC transitions %	15.3 (8.7)	9.4 (6.3)	13.5 (6.9)
SA $\rightarrow$ SC effectiveness %	82.1 (18.7)	76.6 (20.6)	75.6 (23.9)
SC $\rightarrow$ IA transitions %	28.1 (15.5)	12.1 (7.1)	19.0 (11.6)
SC $\rightarrow$ SA transitions %	13.3 (6.2)	21.3 (11.3)	18.7 (10.0)
$\overline{S_g}$	6.22 (4.3)	2.85 (1.8)	5.61 (3.2)
$\overline{S_m}$	7.50 (3.7)	-2.25 (2.6)	3.79 (2.1)

Results of these measures for the three different clusters (i.e., Cluster 1, 2, and 3) are shown in Table 5.1. In order to compare the differences of the measures between the three clusters shown in Table 5.1, we perform the pairwise none-parametric test (i.e., the MannWhitney U-Test) for each pair of the samples from the three clusters. The results are shown in Table 5.2 (The difference is significant at $p<0.05$). The **bolded** p-values are less than 0.05, showing the corresponding differences between the two clusters are significant. For example, the p-value of the **IA effort** measure between cluster 1 and 2 is 0.002, which indicates the difference of the **IA effort** measure is significant between cluster 1 and 2.

The results indicate that the students in cluster 1 achieved the best learning gains ($\overline{S_g} = 6.22$) as well as the best model building performance ($\overline{S_m} = 7.50$), whereas students in cluster 2 had the lowest performance measures ($\overline{S_g} = 2.85$, $\overline{S_m} = -2.25$). Students in cluster 3 had a decent learning gain ($\overline{S_g} = 5.61$), but unlike their learning gains, their model building performance ($\overline{S_g} = 3.79$) was not as good as the students in cluster 1. Overall, we rank the learning performance of the three clusters in descending order as cluster **1** $>$ cluster **3** $>$ cluster **2**.

As we can see from Table 5.1 and Table 5.2, for the **IA effort** measure, students in cluster 1 had higher percentage of their effort spent in information acquisition activities (42.5%)

Table 5.2: Pairwise MannWhitney U-Test result (p-value) for each pair of the samples from the three clusters. The **bolded** p-values are those less than 0.05, showing the corresponding difference between the two clusters is significant.

	p-value Cluster 1 and 2	p-value Cluster 1 and 3	p-value Cluster 2 and 3
IA effort	**0.002**	**0.005**	0.194
Search rate	$<$ **0.00001**	$<$ **0.00001**	$<$ **0.00001**
IA $\rightarrow$ SC transitions	**0.00006**	**0.0012**	**0.006**
IA $\rightarrow$ SC effectiveness rate	0.267	0.407	0.483
SA $\rightarrow$ SC transitions	**0.037**	0.047	**0.033**
SA $\rightarrow$ SC effectiveness rate	0.415	0.405	0.431
SC $\rightarrow$ IA transitions	$<$ **0.00001**	**0.00005**	**0.023**
SC $\rightarrow$ SA transitions	**0.024**	**0.039**	0.119
$\overline{S_g}$	$<$ **0.00001**	0.14	$<$ **0.00001**
$\overline{S_m}$	$<$ **0.00001**	**0.038**	$<$ **0.00001**

than students in cluster 2 (20.8%, $p = 0.002$) and cluster 3 (24.4%, $p = 0.005$). The information acquisition activities helped them gain the information/knowledge from the science resources to help them build their causal maps to teach Betty. According to the definition of coherence relations in Section 4.1.1, information acquisition activities help create potentials that can be used by the solution construction and solution assessment actions. Any generated potential can provide support to subsequent solution construction actions, and it has been shown in [7, 51, 60] that the supported solution construction actions are more likely to be effective (Definition 2). We rank the results of this measure in descending order as cluster **1** $>$ cluster **3** $>$ cluster **2**, which matched up with the performance ranking presented above.

Students in cluster 1 performed more *search* actions (5.6%) to aid their reading compared to the students in cluster 2 (0.3%, $p < 0.00001$) and cluster 3 (3.1%, $p < 0.00001$). Students with higher percentage of *search* actions are typically more efficient in looking for targeted knowledge. So, we rank the results of this measure in descending order as cluster **1** $>$ cluster **3** $>$ cluster **2**, which also matched up with the performance measures presented above.

There were no significant differences of the "IA $\rightarrow$ SC effectiveness" measure (cluster 1: 69.5%; cluster 2: 63.8%; cluster 3: 66.7%) and the "SA $\rightarrow$ SC effectiveness" measure (cluster 1: 82.1%; cluster 2: 76.6%; cluster 3: 79.6%) between the three clusters, where all the differences were not significant (p-values > 0.05). So, the differences of students' model building performance rely more on how many of the solution construction actions are coherently supported by information acquired from prior IA and SA actions. For examples, students in cluster 1 applied significantly more IA $\rightarrow$ SC transitions compared to cluster 2 (31.5% versus 15.9%, $p = 0.00006$), and so did the SA $\rightarrow$ SC transitions (15.3% versus 9.4%, $p = 0.037$). So it was expected that students in cluster 1 could do better in constructing the models than students in cluster 2 ($\overline{S_g}$ = 7.50 versus -2.25, $p < 0.00001$), given that the differences of "IA $\rightarrow$ SC effectiveness" (69.5% versus 63.8%, $p = 0.267$) and "SA $\rightarrow$ SC effectiveness" (82.1% versus 76.6%, $p = 0.415$) measures between cluster 1 and 2 were not significant.

Students in cluster 1 had the highest average percentage measure of the IA $\rightarrow$ SC transitions (cluster 1: 31.5%; cluster 2: 15.9%; cluster 3: 22.0%), where differences between clusters were significant (p-values < 0.05). They also had the highest average percentage measure of SA $\rightarrow$ SC transitions (cluster 1: 15.3%; cluster 2: 9.4%; cluster 3: 13.5%) where the differences between clusters were significant except between cluster 1 and 3 (15.3% versus 13.5%, $p = 0.469$). These measures showed how frequently students were using the information gained from the science resources or the feedback provided in solution assessment to help construct the causal model. During the model construction process, students could get a better understanding of the previously acquired information. For both measures, we rank the three clusters in descending order as cluster **1** $>$ cluster **3** $>$ cluster **2**, which also matched up with the performance ranking.

Besides, students in cluster 1 had the highest average percentage measure of the SC $\rightarrow$ IA transitions (cluster 1: 28.1%; cluster 2: 12.1%; cluster 3: 19.0%), where all the differences between clusters were significant ($p < 0.05$). This showed that they prefer

to divide their solution construction tasks into smaller sub-tasks and go back and forth between information acquisition and solution construction actions to build the model. This divide-and-conquer strategy is useful for constructing more accurate causal models. The rank of this measure also matched up with the performance measures.

Interestingly, students in cluster 1 had the lowest percentage measure of the SC $\rightarrow$ SA transitions while students in cluster 2 had the highest percentage (cluster 1: 13.3%; cluster 2: 21.3%; cluster 3: 18.7%), where differences between clusters were significant (p-values < 0.05). Although the SC $\rightarrow$ SA transitions relate to a useful strategy that uses feedback from solution assessment to correct mistakes in the causal map, it may also imply the use of a trial-and-error approach, which may eventually result in low performance [67, 68].

Overall, students in cluster 1 used more IA and SA actions to support their SC actions, while students in cluster 2 applied less coherent transitions than the other 2 clusters. The lower use of coherent transitions (e.g., students in cluster 2) may also increase the chunk size of information acquisition, solution construction and solution assessment actions which has been shown to have a negative impact on students' learning performance [55].

The measures of IA effort and transitions between action types provided empirical evidence of the differences in students' learning performance. According to the results, students in cluster 1 had the highest usage of information acquisition actions, *search* actions, and the coherent transitions. As expected, they achieved the best learning performance ($\overline{S_g} = 6.22$, $\overline{S_m} = 5.61$).

Despite of not performing as good as students in cluster 1, students in cluster 3 achieved better performance than students in cluster 2 ($\overline{S_g} = 5.61$, $\overline{S_m} = 3.79$). And their pre- and post-learning gain was similar to students in cluster 1. According to the other measures, we can see them having used more IA actions and applied more coherent transitions than students in cluster 2, which matched up with their performance ranking.

On the other hand, the measures for students in cluster 2 have shown them not spending

enough time to read the resources, and the strategies they used are ineffective. For example, they applied more "trial-and-error" strategies, where they repeatedly added causal links to the model without sufficient reading of the resources or the use of SA actions to check the correctness of the added links. Because of the complexity involved in Betty's Brain (e.g., there are 210 possible causal links in the "climate change" learning unit), it is very likely the causal links being added are incorrect in the first place, especially if the students don't understand the knowledge associated with the links. They repeatedly chose one causal link out of all 210 possible links without enough information acquired from science resource to support their choices, while expecting that the solution assessments would help them find and correct all their mistakes. However, it is hard to check all the 210 possible causal links to find out the 15 correct links within a given period of time, and students often get confused after making mistakes repeatedly [68]. Therefore, the poor performance measures ($\overline{S_g} = 2.85$, $\overline{S_m} = -2.25$) for students in cluster 2 were expected.

Next, we generate the two reinforced HMMs according to the algorithms described in Section 4.3.1 and analyze the generated models.

5.1.2 **Analysis of Reinforced Classification Model**

We learn the reinforced classification HMMs by applying the reinforcement learning algorithm that **generates additional** action sequences. This reinforcement learning implies that we are adding more "users" with similar behaviors and then re-deriving a more robust HMM model for that group. The idea here is to generate more accurate and robust classifiers for the groups of students. The action sequences were generated according to the algorithm described in Section 4.3.3. The generated action sequences were combined with the original data set to learn updated HMMs as the reinforced classification models, which are shown in Figure 5.7 (cluster 1), Figure 5.8 (cluster 2), and Figure 5.9 (cluster 3). We also show the state diagrams for the three reinforced classification HMMs in Figure 5.10 (a) (cluster 1), Figure 5.10 (b) (cluster 1), and Figure 5.10 (c) (cluster 3), respectively.

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4
0.67	0.25	0.08	0

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4
h_1	0.61	0.29	0.06	0.04
h_2	0.37	0.41	0.22	0
h_3	0.13	0.21	0.27	0.39
h_4	0.03	0.05	0.10	0.82

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4
IA	$p(read)$	0.55	0.36	0.18	0.04
	$p(search)$	0.11	0.02	0	0
	$p(notes)$	0.12	0.05	0.07	0
SC	$p(mapedit)$	0.19	0.49	0.27	0.08
	$p(clmark)$	0.03	0	0.10	0.05
SA	$p(quiz)$	0	0.08	0.26	0.27
	$p(query)$	0	0	0.08	0.05
	$p(expl)$	0	0	0.04	0.51

Figure 5.7: The reinforced classification HMM of Cluster 1 derived for the Betty's Brain study

$\boldsymbol{\pi}$:

π_1	π_2	π_3
0.88	0.11	0.01

$\boldsymbol{A}$:

	h_1	h_2	h_3
h_1	0.57	0.33	0
h_2	0.28	0.67	0.05
h_3	0.05	0.14	0.81

$\boldsymbol{B}$:

		h_1	h_2	h_3
IA	$p(read)$	0.51	0.15	0.02
	$p(search)$	0	0.01	0.01
	$p(notes)$	0.03	0.01	0.01
SC	$p(mapedit)$	0.31	0.48	0
	$p(clmark)$	0.03	0.11	0
SA	$p(quiz)$	0.05	0.18	0.15
	$p(query)$	0.02	0.06	0.08
	$p(expl)$	0.05	0	0.73

Figure 5.8: The reinforced classification HMM of Cluster 2 derived for the Betty's Brain study

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4
0.61	0.23	0.10	0.06

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4
h_1	0.69	0.21	0.06	0.04
h_2	0.33	0.41	0.21	0.05
h_3	0.19	0.31	0.19	0.31
h_4	0.6	0	0.15	0.79

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4
IA	$p(read)$	0.33	0.32	0.32	0.05
	$p(search)$	0.06	0	0.01	0.01
	$p(notes)$	0.02	0.03	0	0.01
SC	$p(mapedit)$	0.48	0.45	0.41	0.03
	$p(clmark)$	0.04	0.01	0.05	0
SA	$p(quiz)$	0.06	0.09	0.16	0.28
	$p(query)$	0	0.05	0	0.04
	$p(expl)$	0.01	0.05	0.05	0.58

Figure 5.9: The reinforced classification HMM of Cluster 3 derived for the Betty's Brain study

In the reinforced classification model, there were no significant structural changes compared to the original HMMs. For example, the number of hidden states in the reinforced HMMs for each of the groups were unchanged. However, the **A**, **B** and π matrices and vector were updated in a way that improves classification accuracy for the group.

According to our modeling purpose for the *Reinforced classification model* discussed in Section 4.3.3, the reinforcement learning algorithm is aimed to generate action sequences that simulate students' learning behaviors based on existing HMMs and the corresponding performance, measured by the model score achieved. These newly derived action sequences increase the sample size while also expanding the space of behaviors to include others that are similar in that their coherence measures are about the same, and retain the same performance characteristics. This is because of the UCT criterion balances between exploitation and exploration (Section 4.3.2), which also reduces the risk of overfitting and, therefore, increase the accuracy of classifying students into the correct clusters that were

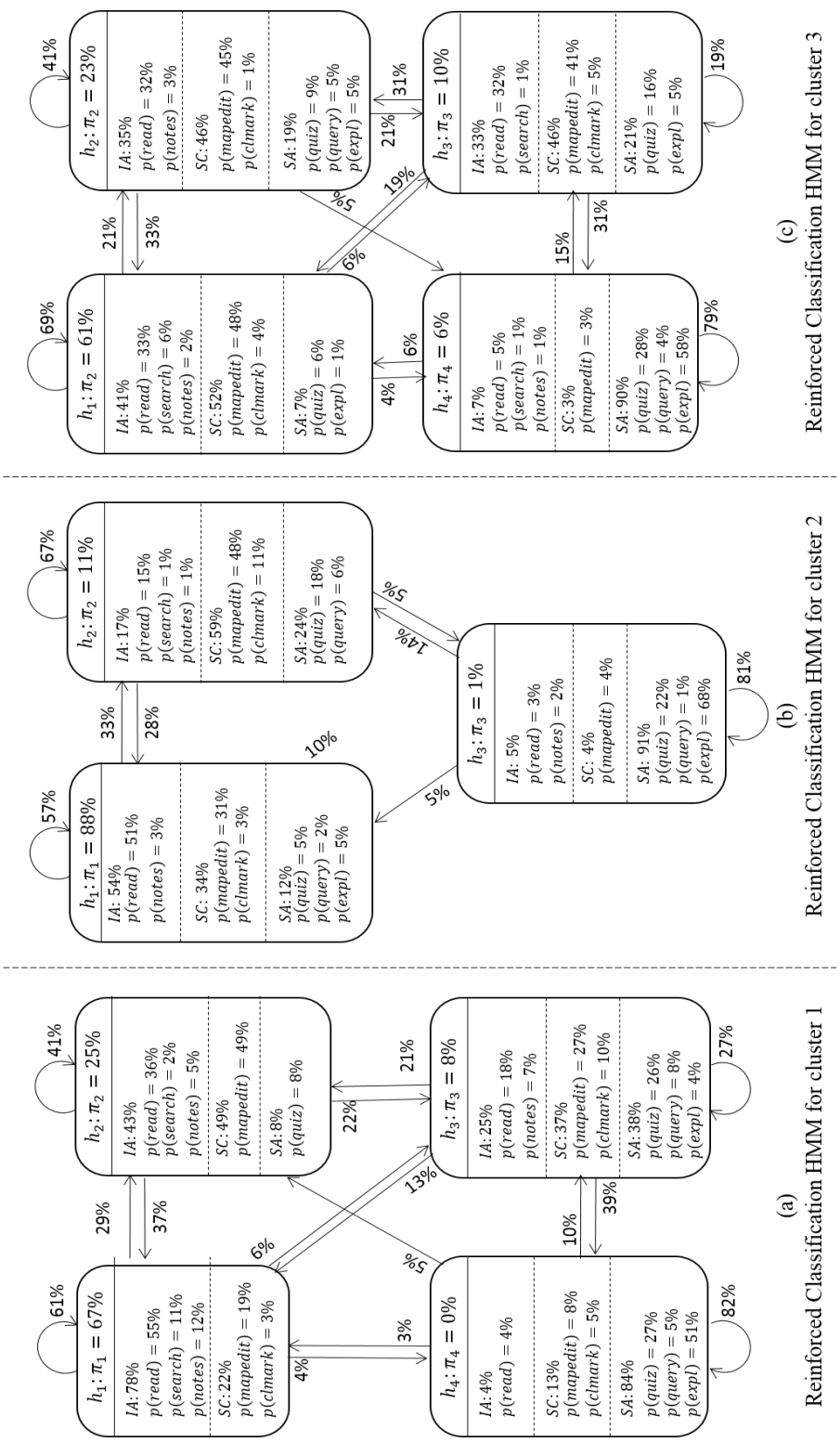


Figure 5.10: Reinforced classification HMMs for the three clusters derived for the Betty's Brain study

derived from HMM clustering.

For the classification, an action sequence, S, is classified into a cluster, whose HMM, λ, has the highest log-likelihood $P(S|\lambda)$ of generating S. So when comparing the reinforced classification models with the original models, we excluded the measures that have no impact on the log-likelihood value, namely $\overline{S_g}$, $\overline{S_m}$, and the effectiveness measures. Results of the measures are shown in Table 5.3, where the **bolded** entries are those with significant change pre- and post-reinforcement learning (p-values < 0.05).

Table 5.3: Comparison of the three clusters for the reinforced classification models in the Betty's Brain study. Results for each measure is presented as mean (standard deviation). The **bolded** entries are those with significant change pre- and post-reinforcement learning (p-values < 0.05).

	Cluster 1	Cluster 2	Cluster 3
IA effort %	43.1 (22.7)	**23.6 (11.5)**	24.9 (13.6)
Search rate %	**8.9 (4.4)**	0.5 (1.2)	**4.9 (2.9)**
IA $\rightarrow$ SC transitions %	30.7 (19.8)	16.2 (7.2)	24.2 (11.9)
SA $\rightarrow$ SC transitions %	16.8 (8.9)	8.9 (4.9)	14.1 (7.5)
SC $\rightarrow$ IA transitions %	30.3 (13.6)	11.8 (6.6)	21.5 (10.8)
SC $\rightarrow$ SA transitions %	12.6 (6.2)	22.1 (10.8)	18.4 (10.2)

As we can see from Table 5.3, the changes between pre- and post-reinforcement learning had slight differences ($p > 0.05$) for most measures, with only a few exceptions (i.e., **Search rate** in Cluster 1: 5.6 $\rightarrow$ 8.9, $p = 0.007$; **Search rate** in Cluster 3: 3.1 $\rightarrow$ 4.9, $p = 0.032$; and **IA effort** in Cluster 2: 23.6 $\rightarrow$ 20.8, $p = 0.048$). These significant changes were made to reinforce the ground truth of some behavioral differences across the three clusters (e.g., students in cluster 1 and 3 used more **search** actions, and students in cluster 2 made the fewest IA actions), which on the other hand, increased the average between-cluster variance. In order to verify the reinforced classification model, we ran the leave-one-out cross validation (LOOCV) on the original data set, and the expanded data set used to learn the updated HMMs. We then compare the results of LOOCVs performed on the two data sets to show the improvements in classification accuracy.

In every iteration of the LOOCV, we take one action sequence s_a from cluster C_i of the

original student data out and perform the reinforcement learning using the rest of the data to generate three updated HMMs (i.e., HMMs for the three clusters discussed above). The action sequence s_a is assigned with classification label j if the **reinforced** HMM of cluster C_j has the highest log-likelihood value for s_a. If j is the same as the **original** cluster label (i.e., i) that s_a belongs to, we say the classification is accurate. The average classification accuracy of all students' action sequences is computed for the LOOCVs.

Our experimental results showed that the average classification accuracy of LOOCV on original data set is **0.68**, and it was increased to **0.9** after using the reinforcement learning which grew the data set to four times of its original size. This result showed significant improvement in classifying students into the correct groups. More accurate classifications can lead to better characterization of students' behaviors, that can then provide the basis for designing targeted scaffolds or feedback that will help this group of students. Scaffolds for each group are then derived based on the reinforced scaffolding HMMs that are discussed next.

5.1.3 **Analysis of Reinforced Scaffolding Model**

We learn the reinforced scaffolding HMMs by applying the reinforcement learning algorithm to **extend** the **existing** action sequences according to the algorithm described in Section 4.3.3. The extended action sequences are used to learn updated HMMs as the reinforced scaffolding model. We show the generated HMMs in Figure 5.11 (cluster 1), Figure 5.12 (cluster 2), and Figure 5.13 (cluster 3). We also show the state diagrams for the three reinforced scaffolding HMMs in Figure 5.14 (a) (cluster 1), Figure 5.14 (b) (cluster 1), and Figure 5.14 (c) (cluster 3), respectively.

Compared to the original HMMs (Figures 5.3, 5.4, and 5.5), the HMMs for the three clusters gradually converged to an isomorphic 3-state HMM structure. However, the evolution of behavioral patterns for the three clusters was different. To show these evolving behaviors, we presented the results of the measures that were used for analyses of the orig-

$\boldsymbol{\pi}$:

π_1	π_2	π_3
0.72	0.28	0

$\boldsymbol{A}$:

	h_1	h_2	h_3
h_1	0.16	0.65	0.19
h_2	0.81	0.13	0.06
h_3	0.16	0.05	0.79

$\boldsymbol{B}$:

		h_1	h_2	h_3
IA	$p(read)$	0.47	0.42	0.12
	$p(search)$	0.11	0.05	0.13
	$p(notes)$	0.11	0.12	0.02
SC	$p(mapedit)$	0.27	0.27	0.07
	$p(clmark)$	0.04	0.04	0.02
SA	$p(quiz)$	0	0.03	0.28
	$p(query)$	0	0.03	0.05
	$p(expl)$	0	0.01	0.31

Figure 5.11: The reinforced scaffolding HMM of Cluster 1 derived for the Betty's Brain study

$\boldsymbol{\pi}$:

π_1	π_2	π_3
0.84	0.16	0

$\boldsymbol{A}$:

	h_1	h_2	h_3
h_1	0.41	0.44	0.15
h_2	0.43	0.45	0.12
h_3	0.12	0.18	0.70

$\boldsymbol{B}$:

		h_1	h_2	h_3
IA	$p(read)$	0.55	0.12	0.08
	$p(search)$	0.12	0.12	0.08
	$p(notes)$	0.05	0.03	0.06
SC	$p(mapedit)$	0.20	0.37	0.05
	$p(clmark)$	0.04	0.10	0.01
SA	$p(quiz)$	0.02	0.18	0.19
	$p(query)$	0.01	0.03	0.01
	$p(expl)$	0.01	0.05	0.42

Figure 5.12: The reinforced scaffolding HMM of Cluster 2 derived for the Betty's Brain study

inal HMMs in Table 5.4. The learning performance measures ($\overline{S_g}$ and $\overline{S_m}$) are excluded here because there is no accurate way to compute the real learning performance using a

$\boldsymbol{\pi}$:

π_1	π_2	π_3
0.77	0.23	0

$\boldsymbol{A}$:

	h_1	h_2	h_3
h_1	0.51	0.39	0.10
h_2	0.64	0.23	0.13
h_3	0.10	0.08	0.82

$\boldsymbol{B}$:

		h_1	h_2	h_3
IA	$p(read)$	0.46	0.37	0.06
	$p(search)$	0.11	0.10	0.05
	$p(notes)$	0.06	0.03	0.01
SC	$p(mapedit)$	0.30	0.27	0.06
	$p(clmark)$	0.02	0.07	0.07
SA	$p(quiz)$	0.03	0.11	0.29
	$p(query)$	0.01	0.03	0.01
	$p(expl)$	0.01	0.02	0.45

Figure 5.13: The reinforced scaffolding HMM of Cluster 3 derived for the Betty's Brain study

virtual agent that is informed by reinforcement learning. We also perform the pairwise MannWhitney U-Test for each pair of samples from the three clusters and show the results (p-values) in Table 5.5. The **bolded** p-values are less than 0.05, implying the corresponding difference between the two clusters is significant. For example, the p-value of the **IA** $\rightarrow$ **SC transitions** measure between cluster 1 and 2 is 0.03, which indicates the difference in the use of the **IA** $\rightarrow$ **SC transitions** measure is significant between cluster 1 and 2.

Table 5.4: Comparison of the three clusters of the reinforced scaffolding models for Betty's Brain. Values in the parenthesis are the measures of the original model.

	Cluster 1	Cluster 2	Cluster 3
IA effort %	38.3 (20.7)	35.8 (17.5)	37.7 (16.9)
Search rate %	10.7 (4.3)	9.3 (3.9)	9.5 (3.4)
IA $\rightarrow$ SC transitions %	44.7 (19.5)	39.5 (16.7)	41.1 (18.3)
SA $\rightarrow$ SC transitions %	36.9 (18.2)	27.5 (13.5)	26.7 (13.7)
SC $\rightarrow$ IA transitions %	41.5 (19.6)	32.6 (15.1)	39.5 (16.4)
SC $\rightarrow$ SA transitions %	17.3 (6.9)	20.3 (8.7)	22.4 (8.9)

As we can see from Tables 5.4 and 5.5, the differences between the three reinforced

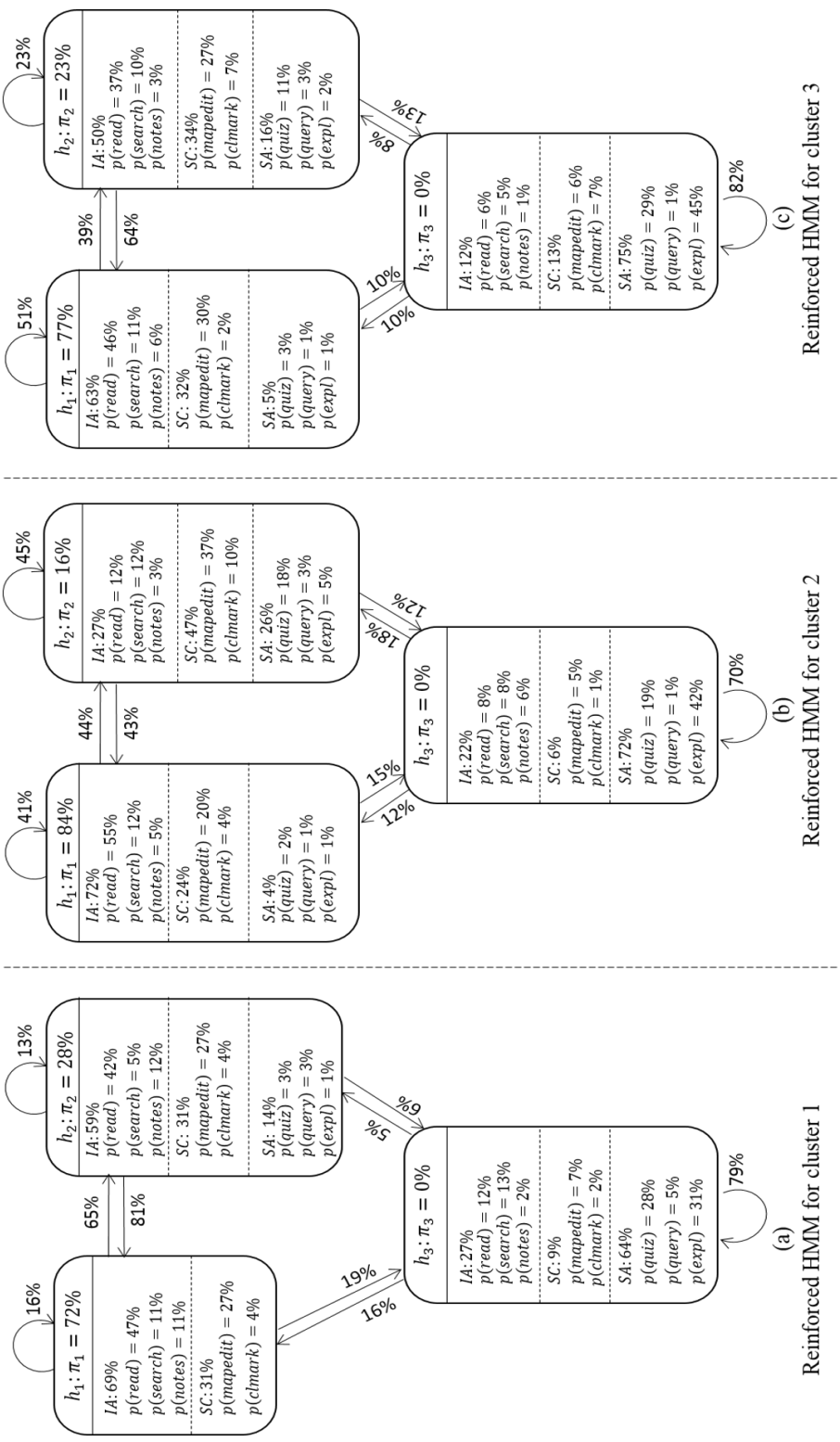


Figure 5.14: Reinforced Scaffolding HMMs for the three clusters

Table 5.5: Pairwise MannWhitney U-Test result (p-value) for each pair of the samples from the three clusters. The **bolded** p-values are those less than 0.05, showing the corresponding difference between the two clusters is significant.

	p-value Cluster 1 and 2	p-value Cluster 1 and 3	p-value Cluster 2 and 3
IA effort	0.091	0.695	0.216
Search rate	0.069	0.164	0.739
IA $\rightarrow$ SC transitions	**0.03**	0.11	0.338
SA $\rightarrow$ SC transitions	0.053	0.117	0.272
SC $\rightarrow$ IA transitions	0.109	0.704	**0.048**
SC $\rightarrow$ SA transitions	0.072	**0.004**	0.095

scaffolding models for most measures were not significant (p-values > 0.05) compared to the original models, which showed that the reinforced scaffolding models remain similar in some measures (e.g., **IA effort**, **Search rate**). However, other measures showed significant differences between the clusters (i.e., the **IA $\rightarrow$ SC transitions** measure between cluster 1 and 2, the **SC $\rightarrow$ IA transitions** measure between cluster 2 and 3, and the **SC $\rightarrow$ SA transitions** measure between cluster 1 and 3). These differences can inform the design of adaptive scaffolds that would most benefit each group, and help them become better learners.

In addition, we can also study the differences that between the reinforced scaffolding derived models as compared to the original models (Table 5.1). We summarize and interpret the differences as follows:

- **IA effort**.The average percentage of effort spent in information acquisition for action sequences in cluster 2 (20.8% $\rightarrow$ 35.8%, $p = 0.0017$) and cluster 3 (24.4% $\rightarrow$ 37.7%, $p = 0.019$) increased significantly so that sufficient information can be gained to support their solution construction actions. On the other hand, the value was slightly decreased for action sequences in cluster 1 (42.5% $\rightarrow$ 38.5%, $p = 0.091$), implying the students should redistribute their efforts to become better learners. Basically, students in cluster 1 were encouraged to translate the acquired information

into knowledge understanding by building the correct causal model.

- **Search Rate**. The average percentage of taking search actions for action sequences in cluster 1 (5.6% $\rightarrow$ 10.7%, $p = 0.002$), cluster 2 (0.3% $\rightarrow$ 9.3%, $p<0.00001$), and cluster 3 (3.1% $\rightarrow$ 9.5%, $p<0.00001$) also increased. In other words, all three groups may be guided to become more targeted information seekers.

- **IA $\rightarrow$ SC transition**. This measure for cluster 1 (31.5% $\rightarrow$ 44.7%, $p = 0.032$), cluster 2 (15.9% $\rightarrow$ 39.5% $p = 0.0002$), and cluster 3 (22.0% $\rightarrow$ 41.1%, $p = 0.0003$) increased significantly, which implied that students should be guided to use more of the information acquired by reading for solution construction activities. This change indicated more use of information acquired by reading science resources (IA) to support subsequent solution construction (SC) actions. The information is the content that can relate to the part of the causal map (e.g., concept entities or causal links).

- **SA $\rightarrow$ SC transition**. The average percentage measure of the SA $\rightarrow$ SC transitions for cluster 1 (15.3% $\rightarrow$ 36.9%, $p = 0.0006$), cluster 2 (9.4% $\rightarrow$ 27.5%, $p = 0.00008$), and cluster 3 (13.5% $\rightarrow$ 26.7%, $p = 0.0007$) also increased. This change indicated more information acquired by taking quiz-related actions (SA) should be used to support subsequent subsequent solution construction actions. The information is usually the feedback about their existing causal maps, which helps students to determine whether existing causal links are correct.

- **SC $\rightarrow$ IA transition**. This measure for cluster 1 (28.1% $\rightarrow$ 41.5%, $p = 0.033$), cluster 2 (12.1% $\rightarrow$ 32.6%, $p = 0.014$), and cluster 3 (19.0% $\rightarrow$ 39.5%, $p = 0.002$) was increased to encourage going back to reading after building parts the of model.

- **SC $\rightarrow$ SA transition**. The average percentage measure of the SA $\rightarrow$ SC transitions for cluster 1 (13.3% $\rightarrow$ 17.3%, $p = 0.082$), and cluster 3 (18.7% $\rightarrow$ 22.4%, $p = 0.064$) changed but with minor differences. Given these results and the change of

percentage measure of the SC → IA transitions, the reinforcement learning slightly favored using coherent transitions between IA and SC over the coherent transitions between SA and SC. The primary reason was to prevent the frequent use of trial-and-error approaches in building the causal model.

As we can see from the results, the data for training reinforced HMMs showed a significantly higher use of coherently supported actions compared to the original dataset, especially for cluster 2 and 3. The students in cluster 1 were doing better in making sure their strategies were coherent in the original model, which also showed why their learning performance is better than the other two clusters (Table5.1).

We can explain these changes according to the reinforced scaffolding HMMs (Figure 5.11, 5.12, and 5.13) and the original HMMs (Figure 5.3, 5.4, and 5.5), that the efforts spent in information acquisition, solution construction, and solution assessment were redistributed so that more coherently supported actions can be taken. This redistribution can lead to strategies that are similar to divide-and-conquer where IA, SC, and SA actions are used coherently to build different parts of the causal map.

This strategy and the use of more coherently supported actions would increase the chance of making correct SC actions (i.e., constructing more accurate causal maps in Betty's Brain), which can eventually lead to better performance. It is what we expected to achieve for the purpose of extending existing action sequences to optimize specific learning performance (i.e., the causal map score).

5.1.4 **Example scaffolds**

Based on these comparative and interpretive analyses, we can derive some example scaffolds that we can experiment with in the future to improve students' learning performance in the Betty's Brain system. An example for providing scaffolds to a student can be done by the following process:

1. Use the student's current action sequence s_c to classify him/her into a group based on the reinforced classification HMMs that are derived from earlier Betty's Brain studies. The student is classified into cluster i if the action sequence s_c has the highest log-likelihood to be generated by the **reinforced classification** HMM for cluster i. For example, we may classify a student into cluster 2, whose action sequence A has the highest log-likelihood to be generated by the HMM for cluster 2 shown in Figure 5.8.

2. Monitor the student's model building performance (i.e., the causal map score) when he/she is working with the Betty's Brain system. And for a period of time (e.g., 10 minutes), if the student's causal map score keeps decreasing or oscillating without improvements, we mark him/her as under-performing and in need for scaffolding.

3. Compare the behavior-related measures of the under-performing students to the measures derived from the corresponding **reinforced scaffolding** model and provide the scaffolding that will help the students improve these behaviors. Use the reinforced classification and scaffolding model we presented in previous sections as an example, if the student is classified into cluster 2 and his/her **IA** $\rightarrow$ **SC transition** measure is 10% while the same measure for the **reinforced scaffolding** model is 39.5% (10% $<<$ 39.5%), we can provide scaffolds to encourage him/her to go back and forth more between information acquisition and solution construction. On the other hand, if the student's **IA** $\rightarrow$ **SC transition** measure is 80% (80% $>>$ 39.5%), we can provide scaffolds to let him/her gain sufficient information by read more pages before switching to do the solution constructions.

The scaffolds provided to the same students depends on which clusters they are classified into. For example, a student whose **SA** $\rightarrow$ **SC transition** measure is 30% is encouraged to take more of this transition if he/she has been classified into cluster 1, where the suggested **SA** $\rightarrow$ **SC transition** measure from the reinforced scaffolding model is 36.9% (30%

$< 36.9\%$); but the student is encouraged to take less of the transition if he/she has been classified into cluster 2, where the suggested **SA** $\rightarrow$ **SC transition** measure from the reinforced scaffolding model is 27.5% (30% $>$ 27.5%).

In fact, a lot more measures can be used as the basis for providing scaffolds besides those we used for this experiment, such as the measure of average percentage for taking **IA** $\rightarrow$ **SC** $\rightarrow$ **SA** or **SC** $\rightarrow$ **SA** $\rightarrow$ **IA** transitions. The reinforced classification and scaffolding models should keep evolving by combining with newer collected data or by performing the reinforcement learning again using original data combined with newer data.

5.2 Experiments with the CTSiM study

The CTSiM learning environment is an OELE developed by our research group, which promotes synergistic learning of science and computational thinking (CT) concepts using a "learning by modeling" approach [1]. In CTSiM, students use an agent-based, block-structured visual language to build simulation models on different science topics, such as kinematics, mechanics, diffusion, and ecological systems [1]. In addition to the model building, the learning environment provides resources and tools to help students acquire information and assess their evolving models.

Students' primary learning activities in CTSiM can be categorized into the three broad classes as [1, 54]:

- *Information acquisition (IA)*: This relates to actions for viewing and acquiring information about domain content and CT-related concepts from hypertext resource libraries (**read**). The *search* action is also categorized as an IA action. Figure 5.15 shows the interface for reading the science resources in CTSiM.

- *Solution construction (SC)*. In CTSiM, there are two types of SC actions: (1) **SC_conc**. This refers to SC actions for constructing an abstract **conceptual** model of the science scenario using an agent-based framework, which includes defining the relevant

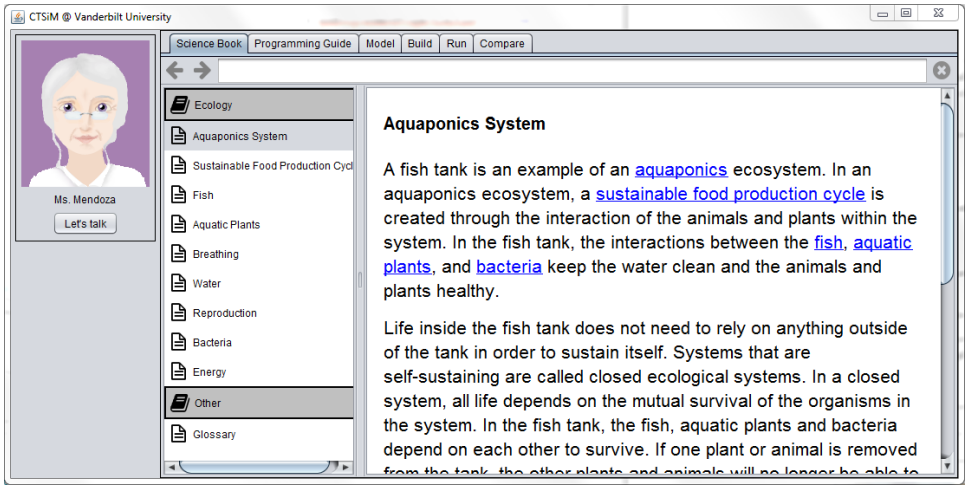


Figure 5.15: Interface for Reading Science Resource in CTSiM

properties and behaviors associated with the agents (**AgentEdit**) and the environment element (**EnvEdit**) where the agents operate in. Figure 5.16 shows the interface for taking **SC_conc** actions in CTSiM; (2) **SC_comp**. This refers to SC actions for building **computational** models of agent behaviors using a block-structured visual language (**BlockEdit**). Figure 5.17 shows the interface for taking **SC_comp** actions in CTSiM.

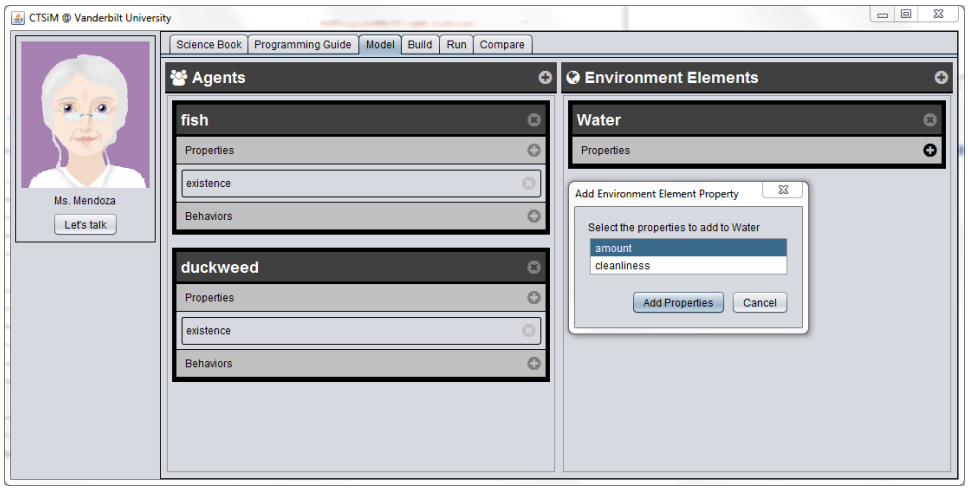


Figure 5.16: Interface for taking conceptual editing **SC_conc** actions in CTSiM

- *Solution assessment (SA)*: This consists of actions for executing their models to analyze the behaviors generated as NetLogo simulations [69], and verifying the cor-

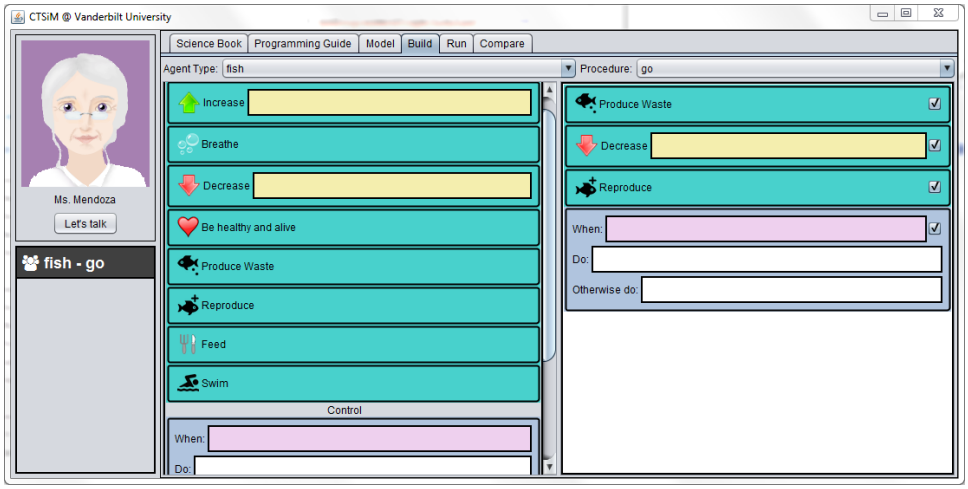


Figure 5.17: Interface for taking computational editing **SC_comp** actions in CTSiM

rectness of their models by comparing them to the behaviors generated by an expert model that runs synchronously with theirs. Students have choices to run the simulation with (**SA_compare** or **Enactment Simulation Actions**) or without (**SA_run** or **Envisionment Simulation Actions**) comparison with the expert model. Figure 5.18 and 5.19 show the interfaces for taking **SA_run** and **SA_compare** actions, respectively.

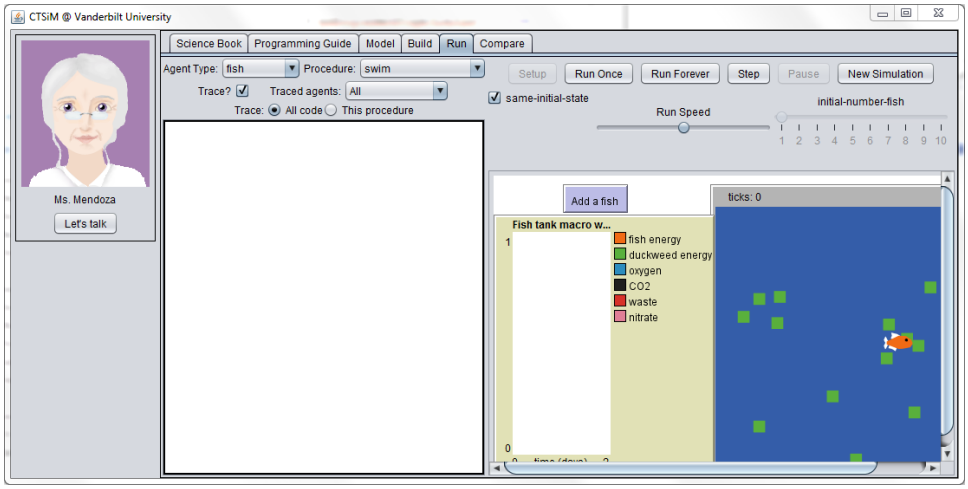


Figure 5.18: Interface for taking **SA_run** actions in CTSiM

Overall, we include 32 different actions that students can perform in the CTSiM OELE. Table 5.6 shows the list of these actions in CTSiM.

Table 5.6: The list of all meaningful actions students can perform in CTSiM

Action category	Action type	Action name/description
IA	**search**	SearchLibraryAction
		ClearLibrarySearchResultsAction
		NavigateToSearchHitAction
		FollowHyperlinkAction
	read	ReadResourceLibraryAction
SC_conc	**AgentEdit**	AddAgentAction
		RemoveAgentAction
		AddAgentPropertyAction
		RemoveAgentPropertyAction
		AddAgentBehaviorAction
		RemoveAgentBehaviorAction
	EnvEdit	AddEnvironmentElementAction
		RemoveEnvironmentElementAction
		AddEnvironmentPropertyAction
		RemoveEnvironmentPropertyAction
SC_comp	**BlockEdit**	InsertBlockAction
		RemoveBlockAction
		MoveBlockAction
		MoveBlockParameterAction
		RemoveBlockArgumentAction
		InsertBlockParameterAction
SA_run	**RunSim**	ChangeTraceModeAction
		SetupEnactmentSimulationAction
		StartEnactmentSimulationAction
		StopEnactmentSimulationAction
		StartOverEnactmentSimulationAction
SA_compare	**CompSim**	SetupEnvisionmentSimulationAction
		StartEnvisionmentSimulationAction
		StopEnvisionmentSimulationAction
		StartOverEnvisionmentSimulationAction
		ChangeModelComparisonModeAction
		SelectSimulationBehaviorAction

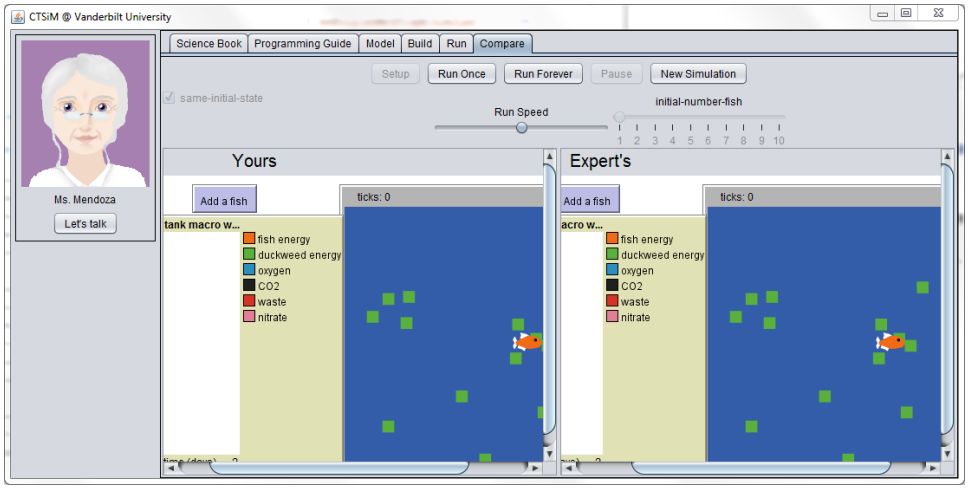


Figure 5.19: Interface for taking **SA_compare** actions in CTSiM

The CTSiM OELE can (1) leverage the use of multiple linked representations to help students become better learners [55], (2) measure effectiveness of adaptive scaffolds provided to students who worked in CTSiM [70], and (3) categorize students' learning behaviors using data collected from CTSiM [54].

During the learning process in CTSiM, all students' actions, as well as their model revision history are logged in a raw data format. We applied the methods as discussed in Section 4.1.1 using the raw data to generate the Action-View representations, which was then used as the data set to support our learner modeling approach.

Specifically, we perform the learner modeling algorithm illustrated in Chapter 4 using data collected from the CTSiM study conducted in a 6^{th} grade classroom where students worked on three units. For the first unit, students worked individually to build a **roller-coaster** model in CTSiM. They then worked on building a **macroscopic** model of a fish tank ecosystem (the second unit), which was followed by a **microscopic** model (the third unit) for the fish tank ecosystem. We have used two types of measures to compute students' learning performances:

1. The model distance from the student-built models to the canonical expert model including the conceptual and the computational parts ($\overline{S_m}$). This measure shows how

good the students perform in their model building tasks (the **lower**, the **better**).

2. The pre- and post-test learning gains in the corresponding domains (i.e., kinematics for the **rollercoaster** unit and ecology for the **macroscopic** and the **microscopic** units), as well as in computational thinking (CT) skills ($\overline{S_g}$). These measures show how much they learned as a result of the model building intervention (the **higher**, the **better**).

We first derive and discuss the learner models (i.e., the HMMs) for the original data set (Section 5.2.1) and then analyze the results derived by applying reinforcement learning (Section 5.2.2 and Section 5.2.3) as we did for the Betty's Brain data (Section 5.1).

5.2.1 Analysis of HMMs for the Original Data

Since the dataset collected from the CTSiM study consists of students' activities on different units, it is possible to observe their behavior evolution as they progress through these units. So we first run the HMM clustering algorithm to discover groups of action sequences for the first unit (i.e., **rollercoaster**). We compare these results with a later unit (i.e., **macroscopic**) and discuss how students' behaviors evolve over time. Description of possible actions that can be taken in CTSiM can be found in Table 5.6.

5.2.1.1 HMM Clustering Results for the "Rollercoaster" Unit

Similar to the experiments with the Betty's Brain data, we derive the optimum number of clusters by computing the partition mutual information (PMI) for different numbers of clusters as illustrated in Section 4.2.3. Figure 5.20 shows the plot of the PMI values for different numbers of clusters. As we can see, the PMI value started to level off when the number of clusters reached 4. To prevent the HMMs to be over-complex, we chose 4 as the number of clusters for this experiment.

The students can be characterized into four different groups using the method described

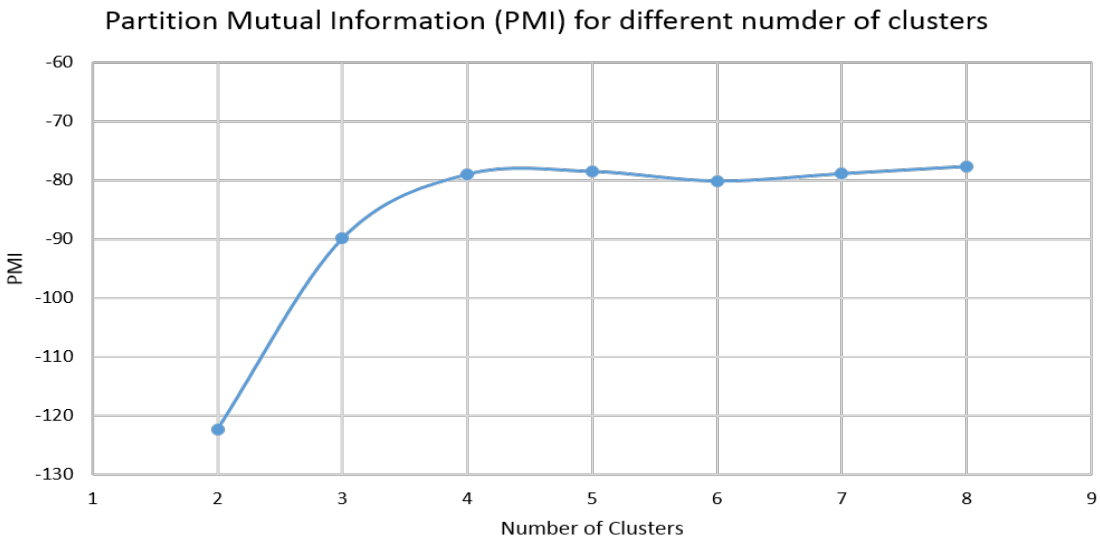


Figure 5.20: Partition Mutual Information values calculated for Data from the **Roller-coaster** Unit in CTSiM

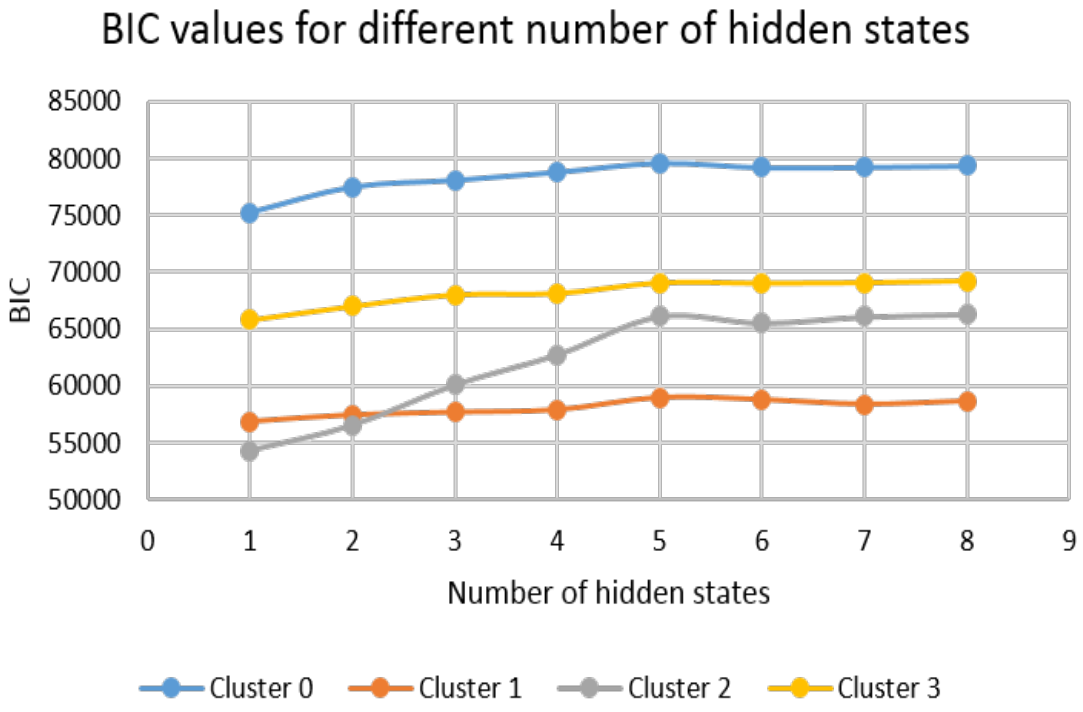


Figure 5.21: Bayesian Information Criterion (BIC) values calculated for the four clusters derived from HMM clustering on data collected from the **rollercoaster** unit in CTSiM. The BIC values were re-scaled to fit the window.

in Section 4.2.3. And for each cluster, we learn an HMM that is representative of the students' behaviors for that group. As discussed in Section 4.2.2, we also compute the Bayesian Information Criterion (BIC) to determine the best number of hidden states when learning the HMMs. Figures 5.21 shows the plots of the BIC values computed for the four derived clusters. As we can see from the figure, the BIC values for all four clusters started to level off at around 5 hidden states. Therefore, we chose 5 as the number of hidden states to generate HMMs for all four groups of students. The generated HMMs for the four clusters are shown in Figure 5.22 (cluster 0), Figure 5.23 (cluster 1), Figure 5.24 (cluster 2), and Figure 5.25 (cluster 3), respectively.

In order to better understand the four HMMs, we use some measures to compare the clustering results as we did for the Betty's Brain experiment (Section 5.1.1). Since the SC and SA actions in CTSiM are divided into further categories (i.e., SC to SC_Conc and SC_Comp, SA to SA_compare and SA_run), we extend the measures used in our Betty's Brain study with additional behavioral measures derived from our previous work [54, 68]. The measures are listed as following (description of the actions can be found in Section 5.2):

- **IA effort**, which is the average percentage of taking IA actions among all the actions performed by a student.
- **Search Rate**, which is the average percentage of taking *search* actions among all the IA actions.
- **IA $\rightarrow$ SC transitions**, which is the average percentage of IA actions that support later SC actions among all the IA actions performed by a student. This transition refers to a strategy that is described in the context of CTSiM system as " the strategy that involves acquiring information about an agent/environment behavior before modeling it" [68].
- **IA $\rightarrow$ SC effectiveness**, which is the average percentage of effective SC actions that

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4	π_5
0.09	0.89	0	0.02	0

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4	h_5
h_1	0.13	0.07	0.42	0.16	0.21
h_2	0.06	0.86	0.03	0	0.04
h_3	0.05	0.03	0.51	0.25	0.16
h_4	0.11	0.01	0.45	0.25	0.07
h_5	0.26	0.10	0.35	0.22	0.06

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0	0.05	0	0	0
	$p(search)$	0	0.01	0	0	0
SC	$p(SC\_comp)$	0.02	0.80	0	0.01	0.05
	$p(SC\_conc)$	0.01	0.11	0	0	0.01
SA	$p(SA\_compare)$	0.14	0	0.54	0.49	0.71
	$p(SA\_run)$	0.83	0.04	0.46	0.50	0.22

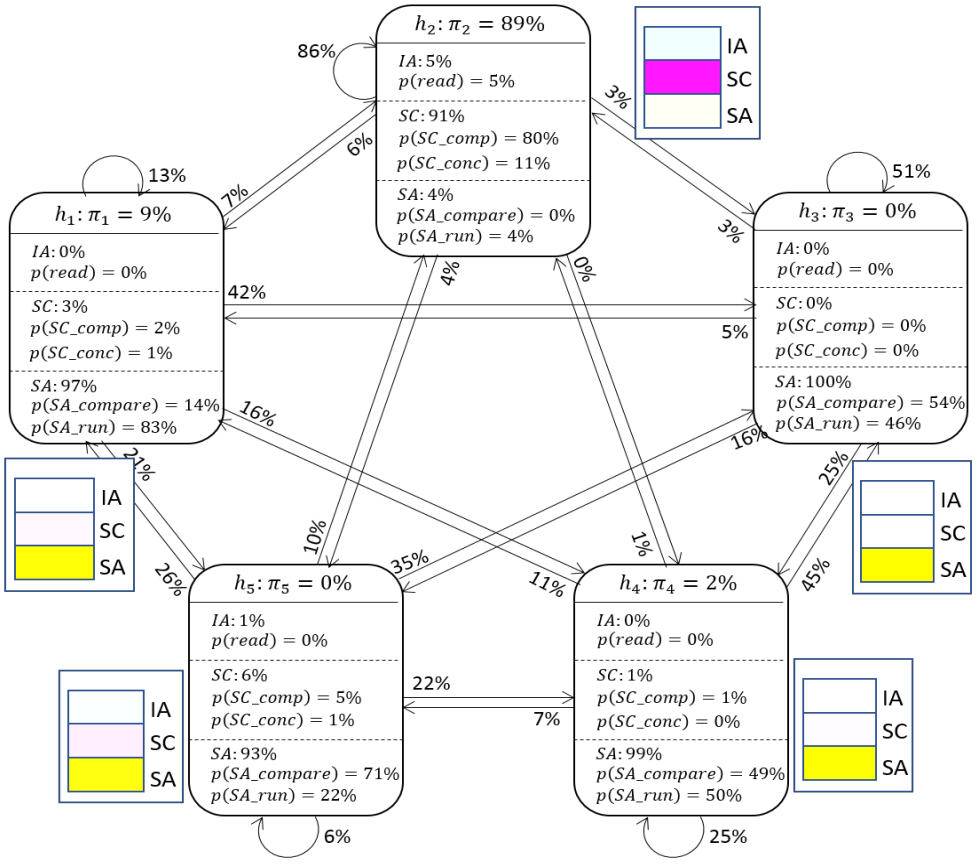


Figure 5.22: The HMM of Cluster 0 (33 students) derived for the **rollercoaster** unit

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4	π_5
0.08	0	0	0.92	0

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4	h_5
h_1	0.15	0.25	0.23	0.18	0.20
h_2	0.05	0.80	0.02	0.05	0.08
h_3	0.06	0.06	0.59	0.02	0.27
h_4	0.13	0.13	0.11	0.56	0.07
h_5	0.13	0.02	0.30	0.01	0.54

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0.04	0	0	0.13	0
	$p(search)$	0	0	0	0.02	0
SC	$p(SC\_comp)$	0.06	0.97	0	0.04	0.01
	$p(SC\_conc)$	0.03	0.01	0	0.26	0
SA	$p(SA\_compare)$	0.05	0	0.47	0.18	0.30
	$p(SA\_run)$	0.82	0.02	0.53	0.37	0.69

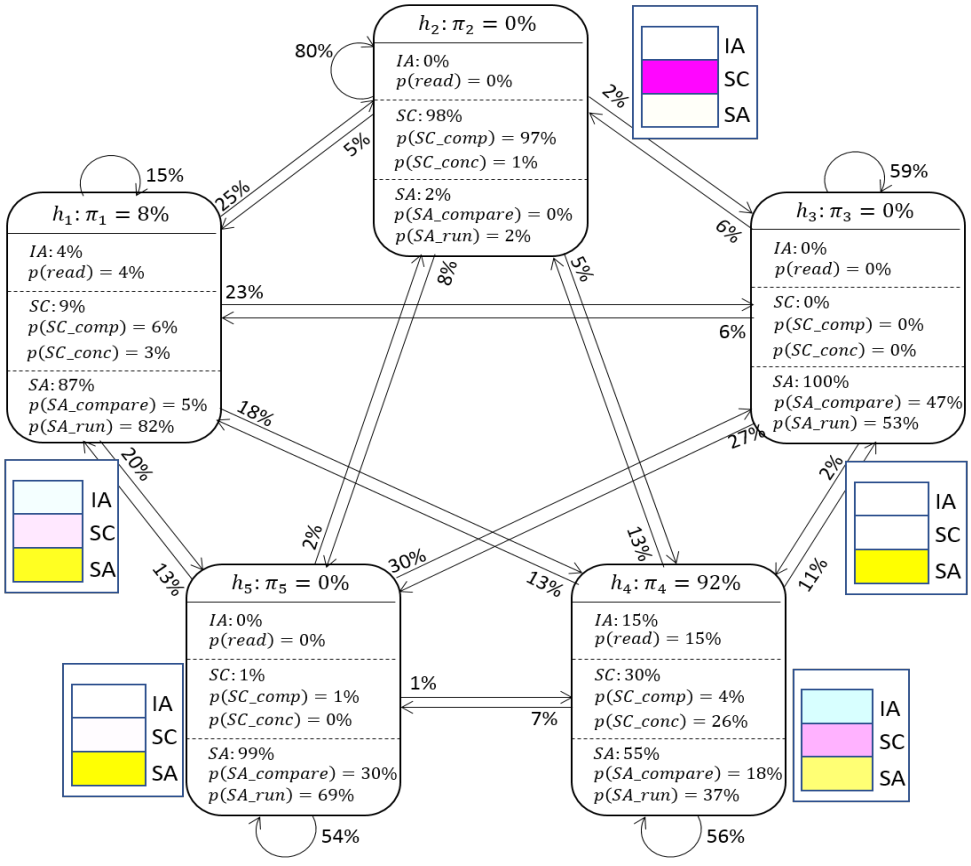


Figure 5.23: The HMM of Cluster 1 (23 students) derived for the **rollercoaster** unit

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4	π_5
0.97	0	0	0.02	0.01

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4	h_5
h_1	0.86	0.08	0.01	0.02	0.02
h_2	0.01	0.43	0.12	0.09	0.35
h_3	0	0.07	0.90	0.01	0.02
h_4	0.18	0.23	0.21	0.09	0.28
h_5	0.04	0.50	0.06	0.10	0.30

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0.44	0.02	0	0.13	0.02
	$p(search)$	0.06	0	0	0	0
SC	$p(SC\_comp)$	0.04	0.92	0.02	0.72	0.96
	$p(SC\_conc)$	0.43	0	0	0.02	0.01
SA	$p(SA\_compare)$	0	0	0.32	0.01	0
	$p(SA\_run)$	0.02	0.06	0.66	0.12	0.01

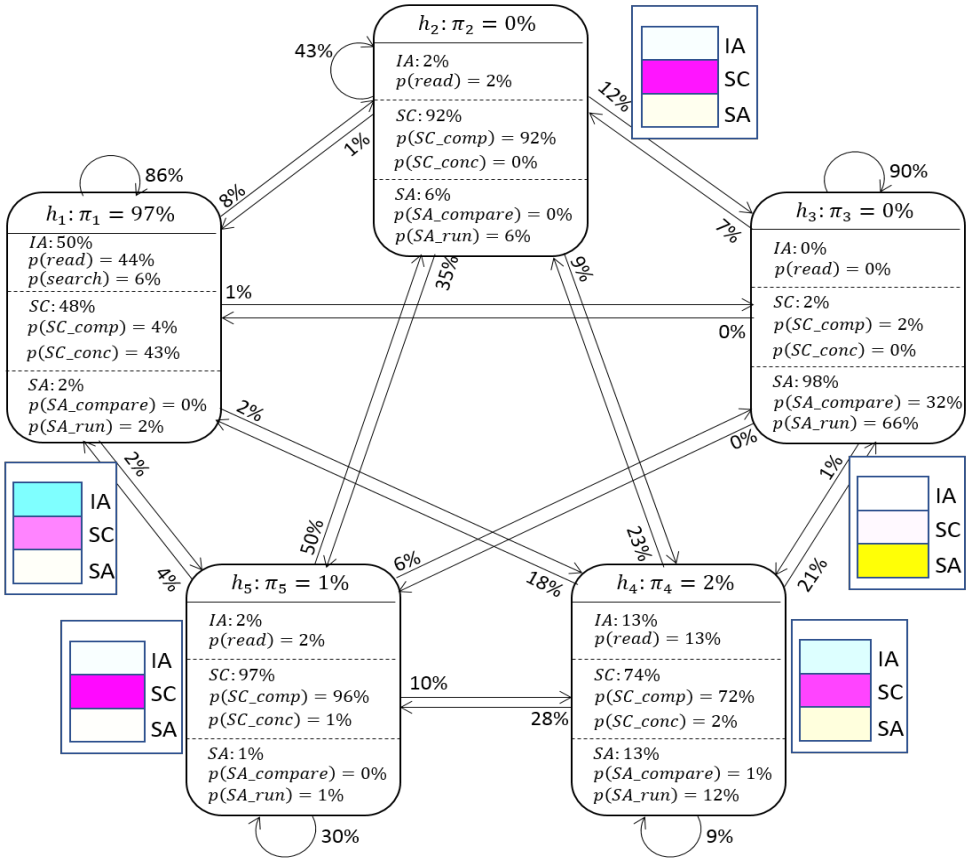


Figure 5.24: The HMM of Cluster 2 (4 students) derived for the **rollercoaster** unit

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4	π_5
0.07	0	0.02	0.90	0

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4	h_5
h_1	0.35	0.33	0.25	0.02	0.05
h_2	0.24	0.25	0.41	0.01	0.09
h_3	0.28	0.47	0.12	0	0.12
h_4	0	0.01	0.03	0.86	0.09
h_5	0.02	0.04	0.05	0.01	0.88

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0	0.01	0	0.30	0.01
	$p(search)$	0	0	0	0.02	0
SC	$p(SC\_comp)$	0.01	0.05	0.03	0.01	0.94
	$p(SC\_conc)$	0	0	0	0.66	0
SA	$p(SA\_compare)$	0.31	0.43	0.09	0	0
	$p(SA\_run)$	0.68	0.51	0.88	0.03	0.05

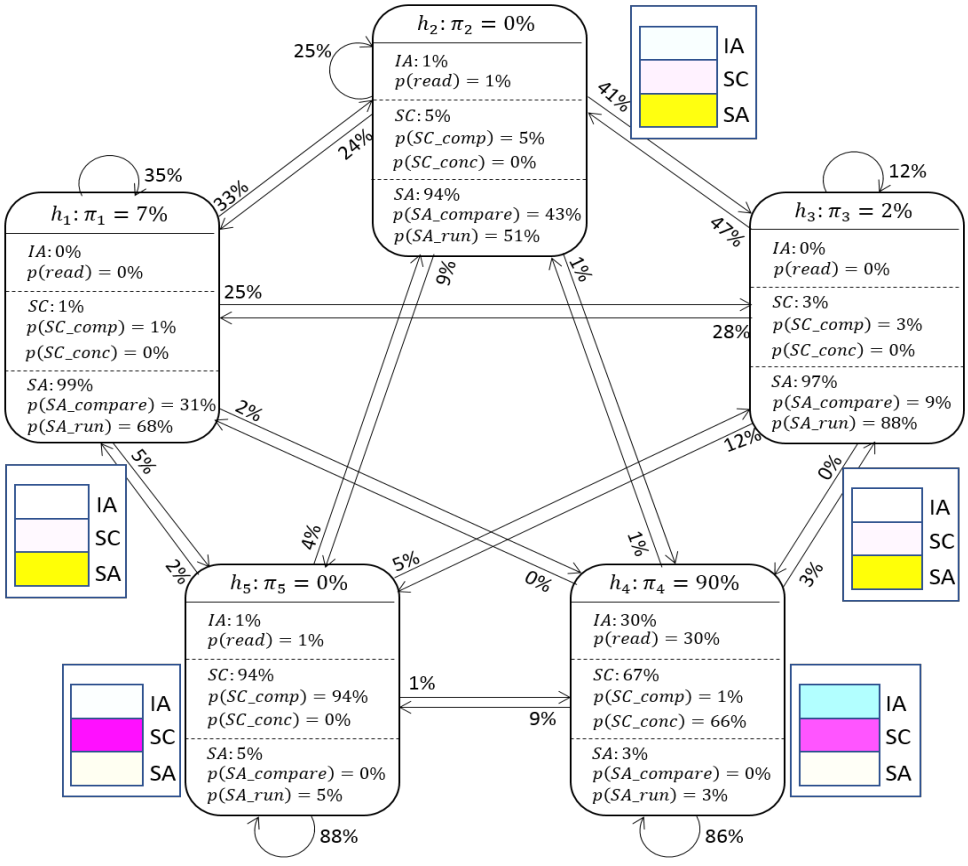


Figure 5.25: The HMM of Cluster 3 (38 students) derived for the **rollercoaster** unit

are supported by IA actions. We use this to measure how effective the **IA → SC transitions** are in help building the model correctly.

- **SA → SC transitions**, which is the average percentage of SA actions that support later SC actions among all the SA actions performed by a student. This is the strategy discussed in [68], that uses information from the simulation results of solution assessment actions to support model building actions.

- **SA → SC effectiveness**, which is the average percentage of effective SC actions that are supported by SA actions. We use this to measure how effective the **SA → SC transitions** are in help building the model correctly.

- **SC → IA transitions**, which is the average percentage of SC actions that support later IA actions among all the SC actions performed by a student. This relates the strategy of seeking information relevant to the part of the model currently being constructed by the student, which can be specified as an SC action (e.g., conceptual edits or computational model edits) temporally followed by a coherent IA action (e.g., read science resource) [68].

- **SC → SA transitions**, which is the average percentage of SC actions that support later SA actions among all the SC actions performed by a student. This relates to a strategy of seeking information relevant to the current model built by a student by taking solution assessment actions.

- **Conceptual Model Edit effort**, which is the average percentage of conceptual model edits (**SC_conc**) among all the actions performed by a student. In CTSiM, the conceptual model edit actions are the SC actions for constructing an abstract conceptual model of the science topic using an agent-based framework [1].

- **Computational Model Edit effort**, which is the average percentage of computational model edits (**SC_comp**) among all the actions performed by a student. In

CTSiM, the computational model edit actions refer to SC actions for building computational models of agent behaviors using a block-structured visual language [1].

- **Model Testing effort**, which is the average percentage of model test actions (**SA_run**) among all the actions performed by a student. In CTSiM, the model test actions are the SA actions for verifying the correctness of students' domain model by running simulations [1].

- **Model Comparison effort**, which is the average percentage of model comparison actions (**SA_compare**) among all the actions performed by a student. In CTSiM, the model test actions are the SA actions for verifying the correctness of students' domain model by comparing them with an expert model that run synchronously with theirs [1].

- **Compare Model in Parts**, which is the average percentage of model comparison actions (**SA_compare**) that explicitly choose a subset of the agents behavior(s) to all the **SA_compare** actions performed by a student. It has been shown that this **Compare Model in Parts** action is a useful model verification and debugging strategy, especially in complex units with multiple agents and multiple agent behaviors [68].

- **Coherence of Model edits**, which is the average number of transitions described by a conceptual model edit action (**SC_conc**) followed by a coherent computational model edit action (**SC_comp**). This is also a specific strategy measure for CTSiM, that "involves maintaining the correspondence between the conceptual and computational models for each agent/environment behavior" [68].

- **Number of ModelEdit chunks**, which is the average number of model-build chunks that are described as consecutive conceptual model edit actions (**SC_conc**) or coherent computational model edit actions (**SC_comp**).

- **Size of ModelEdit chunks**, which is the average size of model-build chunks that

are described as consecutive conceptual model edit actions (**SC_conc**) or coherent computational model edit actions (**SC_comp**). The measures of **Number of Model-Edit chunks** and **Size of Model-Edit chunks** are useful for comparing and analyzing students' behavior in different groups [68]. Generally speaking, a student who has more but smaller model-build chunks, has applied more frequently the strategy to divide the model building task into smaller sub-tasks, and worked on different parts of the model separately.

- **Number of Actions**, which is the average number of actions taken by the students within a cluster.

- $\overline{S_g}$, which is average pre- and post-test score gain. It measures on average, how much students learned as a result of the model building intervention, which is the sum of domain gain and CT gain (the **higher**, the better).

- $\overline{S_m}$, which is the average model distance from the student-built models to the canonical expert model including conceptual and computational parts. This measure shows how good the students perform in their model building tasks (the **lower**, the better).

Table 5.7 shows the results of the measures defined above. We also perform the pairwise MannWhitney U-Test for each pair of the samples from the three clusters and show the results (p-values) in Table 5.8.

As we can see from Tables 5.7 and 5.8:

- Students in cluster 2 had the best learning gain ($\overline{S_g} = 9.5$) as well as the model building performance ($\overline{S_m} = 22.5$). According to the $\overline{S_g}$ row in Table 5.8, their learning gain measure ($\overline{S_g} = 9.5$) was significantly better than students in the other three clusters (p-value < 0.05), while their model building performance ($\overline{S_m} = 22.5$) was slightly better than the other three clusters.

Table 5.7: Comparison of the four Clusters for the **rollercoaster** unit in CTSiM

	Cluster 0 (n = 33)	Cluster 1 (n = 23)	Cluster 2 (n = 4)	Cluster 3 (n = 38)
IA effort %	1.4 (2.6)	1.7 (1.6)	7.6 (1.7)	3.7 (2.8)
Search Rate %	2.3 (2.9)	1.9 (1.6)	8.2 (2.3)	1.4 (0.7)
IA→SC transitions%	14.4 (12.2)	12.7 (9.6)	14.5 (4.3)	15.3 (11.6)
IA → SC effectiveness %	61.7 (23.4)	59.6 (22.0)	64.5 (13.1)	65.6 (20.9)
SA→SC transitions%	7.6 (4.6)	10.0 (8.9)	11.0 (2.6)	12.9 (7.7)
SA → SC effectiveness %	63.2 (23.5)	66.9 (21.6)	65.9 (15.3)	69.7 (21.2)
SC→IA transitions%	0.6 (0.5)	0.5 (0.4)	2.5 (1.1)	0.7 (0.2)
SC→SA transitions%	18.6 (8.6)	19.3 (13.5)	11.4 (7.2)	14.6 (12.8)
Conceptual Model Edit %	4.3 (3.7)	4.3 (3.2)	9.3 (2.6)	8.7 (3.8)
Computational Model Edit %	26.5 (21.0)	30.3 (15.6)	36.2 (16.2)	40.1 (25.6)
Model Testing effort %	34.1 (22.8)	42.5 (25.1)	27.6 (8.5)	34.4 (19.8)
Model Comparison effort %	33.4 (25.6)	20.9 (11.5)	12.7 (4.2)	13.2 (5.6)
Compare Model in Parts %	34.2 (18.6)	24.8 (16.3)	29.0 (11.0)	23.6 (17.2)
Coherence of Model Edits	1.1 (1.5)	1.3 (0.9)	4.2 (0.8)	2.9 (1.1)
Number of ModelEdit chunks	11.7 (4.8)	10.5 (5.1)	15.2 (3.1)	13.8 (5.3)
Size of ModelEdit chunks	15.2 (10.6)	16.5 (13.2)	5.6 (2.9)	10.9 (7.9)
Number of Actions	517 (176)	474 (144)	180 (63)	298 (119)
$\overline{S_g}$	4.3 (5.1)	4.5 (4.8)	9.5 (5.1)	4.4 (5.2)
$\overline{S_m}$	22.6 (11.6)	24.8 (14.7)	22.5 (10.6)	22.6 (11.3)

- Students in cluster 0 ($\overline{S_g}$ = 4.3), cluster 1 ($\overline{S_g}$ = 4.5), and cluster 3 ($\overline{S_g}$ = 4.4) had similar learning gain measures, where differences were not significant (p-values $>$ 0.5). But according to Tables 5.7 and 5.8, students in Cluster 1 performed worse in building the domain model compared to students in Cluster 0 ($\overline{S_m}$ = 24.8 versus $\overline{S_m}$ = 22.6, $p < 0.05$) and cluster 3 ($\overline{S_m}$ = 24.8 versus $\overline{S_m}$ = 22.6, $p < 0.05$).

- Despite having similar learning gains and model building performance, students in cluster 3 took less number of actions compared to students in cluster 0 (298 versus 517, $p < 0.00001$), showing that students in cluster 3 are more efficient.

Overall, we rank the students' performance from different clusters in descending order as cluster **2** $>$ cluster **3** $>$ cluster **0** $>$ cluster **1**. We analyze and summarize the differences between the four clusters as following:

Table 5.8: Pairwise MannWhitney U-Test result (p-value) for each pair of the samples from the three clusters. The **bolded** p-values are less than 0.05, showing the corresponding differences between the two clusters are significant. For example, the p-value of the **IA effort** measure between cluster 0 and 1 is 0.001, which indicates the difference of the **IA effort** measure is significant between cluster 0 and 1

	p-value Cluster 0 and 1	p-value Cluster 0 and 2	p-value Cluster 0 and 3	p-value Cluster 1 and 2	p-value Cluster 1 and 3	p-value Cluster 2 and 3
IA effort	0.136	**0.001**	**0.009**	**0.003**	**0.008**	**0.029**
Search Rate	0.094	**0.002**	**0.001**	**0.002**	0.09	**0.001**
IA→SC transitions	0.206	0.083	0.2	**0.02**	**0.021**	0.257
IA→SC effectiveness	0.548	0.351	0.58	0.525	0.315	0.494
SA→SC transitions	**0.04**	**0.028**	**0.008**	0.421	0.125	0.376
SA→SC effectiveness	0.489	0.446	0.512	0.628	0.359	0.525
SC→IA transitions	0.314	**0.008**	0.072	**0.014**	0.291	**0.029**
SC→SA transitions	0.5	**0.039**	0.106	**0.02**	**0.012**	**0.04**
Conceptual Model Edit	0.438	**0.04**	**0.004**	**0.014**	**0.002**	0.52
Computational Model Edit	0.115	**0.014**	**0.001**	**0.006**	**0.009**	0.168
Model Test Effort	**0.043**	**0.011**	0.256	**0.008**	**0.031**	**0.009**
Model Comparison Effort	**0.001**	**0.005**	<**0.00001**	**0.006**	**0.002**	0.599
Compare Model in Parts	**0.0006**	0.142	**0.02**	0.275	0.721	0.247
Coherence of Model Edits	0.358	**0.002**	**0.004**	**0.0008**	**0.002**	**0.036**
Number of ModelEdit chunks	0.439	**0.002**	0.211	**0.002**	**0.041**	**0.01**
Size of ModelEdit chunks	0.609	**0.0003**	**0.016**	**0.00001**	**0.001**	**0.011**
Number of Actions	0.29	**0.0003**	<**0.00001**	**0.001**	**0.002**	**0.016**
$\overline{S_g}$	0.566	**0.002**	0.782	**0.003**	0.522	**0.003**
$\overline{S_m}$	**0.034**	0.4	0.944	0.06	**0.032**	0.087

Measures of IA effort and Search rate. Students in cluster 2 had the highest **IA effort** (7.6%) and **search rate** (8.2%) which were significantly higher than the other three clusters ($p > 0.05$, as shown in row **IA effort** and **Search rate** in Table 5.8). IA and search actions can help students acquire more information and be more adept in looking for targeted information. On the other hand, students in cluster 0 and 1 had similar measures for **IA effort** (1.4% versus 1.7%, $p = 0.136$) and **search rate** (2.3% versus 1.9%, $p = 0.094$), while students in cluster 3 did better than cluster 0 (3.7 versus 1.4, $p = 0.009$) and cluster 1 (3.7 versus 1.7, $p = 0.008$) in the **IA effort** measure. These measures generated a good match with the performance ranking of the four clusters (i.e., cluster **2** > cluster **3** > cluster **0** > cluster **1**).

Measures of IA→SC and SA→SC effectiveness. For the **IA→SC effectiveness** and **SA→SC effectiveness** in Tables 5.7 and 5.8, both of these two measures had close values with differences that were not significant (all p-values > 0.05), which means that if taking coherently supported SC actions, students from different groups would have similar percentage of building the model correctly. Therefore, it is more depending on applying better strategy and taking coherently supported actions to make effective solution construction actions, that result in the differences of learning performance across the four clusters.

Measures of IA→SC and SC→IA transitions. For the **IA→SC transitions** in Tables 5.7 and 5.8, students in cluster 0 (14.4%), 2 (14.5%), and 3 (15.3%) applied more of the model building strategy (**SC→IA transition**) than students in cluster 1 (12.7%), where they used the acquired information from the science resources to help building the domain model. On the other hand, the **SC→IA** transitions, which implies the strategy of seeking information relevant to the part of the model currently being constructed by the student [68], was done significantly more by students in Cluster 2 (2.5%, p-values < 0.05). These coherent transitions allow students to go back and forth between solution construction and information acquisition, and can help them understand the different parts to be built in the domain model. In general, these results also matched the performance ranking

of the four clusters (i.e., cluster **2** > cluster **3** > cluster **0** > cluster **1**).

Measures of SA→SC and SC→SA transitions. Compared to the coherent transitions between IA and SC actions, these measures show how frequently students run simulations to verify their models and use information gained from simulation results to edit the models. However, taking this strategy too often implies the use of trial-and-error approaches. As discussed in the experiment with Betty's Brain in Section 5.1.4, the trial-and-error approach can end up with bad model building performance if the modeling task is complex. In CTSiM, the modeling tasks are also complex because they have to model behaviors of individual agents and interactions between agents [68, 1]. According to Tables 5.7 and 5.8, students in cluster 0 and 1 took more **SC→SA** transitions (18.6% and 19.3%, p = 0.5) while the measure of their **SC→IA** transitions were extremely low (0.6% and 0.5%, p = 0.314), which showed the higher use of trial-and-error approaches among students in cluster 0 and 1. This also confirmed that the performance ranking of cluster 2 and 3 being better than cluster 0 and 1.

Measures of solution construction and solution assessment efforts. Generally speaking, these results show how much effort (measured by the percentage of the SC and SA actions among all actions) the students have put in solution construction and solution assessment. According to the results, we see that students in cluster 2 and 3 focused more on solution construction (**Conceptual** and **Computational Model Edits**), while students in cluster 0 and 1 took much more solution assessment actions to verify their models (**Model Test** and **Comparison**). Overall, all students took too many SC and SA actions in comparison to IA actions. However, students in cluster 2 and 3 did slightly better in reading their science resources. So, these measures also matched the performance ranking of the four clusters (i.e., cluster **2** > cluster **3** > cluster **0** > cluster **1**).

The measure of Compare Model in Parts. The **Compare Model in Parts** strategy is a very useful model verification and debugging strategy, especially in complex units with multiple agents and multiple agent behaviors [68]. It allows students to verify smaller parts

of the model, so that they can get better understanding of what parts went wrong in their models. Surprisingly, students in cluster 0 (34.2%) applied this strategy the most, followed by clusters 2, 1, and 3. This was not a perfect match of the performance ranking (i.e., cluster **2** > cluster **3** > cluster **0** > cluster **1**). However, given the fact that students in cluster 0 took more **Model Comparison** actions (33.4%) than students in cluster 1 (20.9%, $p = 0.001$), cluster 2 (12.7%, $p = 0.005$), and cluster 3 (13.2%, $p < 0.00001$), students in cluster 1 got more chance to use and understand the usefulness of the **Compare Model in Parts** strategy.

The measure of Coherence of Model Edits. This measure shows how good the students maintain the correspondence between the conceptual and computational models for each agent/environment behavior. According to the results of **Coherence of Model Edits** measure shown in Tables 5.7 and 5.8, we can rank this measure for the four clusters as: cluster **2** (4.2) > cluster **3** (2.9) > cluster **0** (1.1) ≈ cluster **1** (1.3). The p-values between each pair of clusters were significant except between cluster 0 and 1 ($p = 0.358$). So, the results of this measure was a good match of the performance ranking (i.e., cluster **2** > cluster **3** > cluster **0** > cluster **1**).

Measures of Number of ModelEdit chunks and Size of ModelEdit chunks. As discussed earlier, the use of more and smaller ModelEdit chunks indicate a better application of divide-and-conquer strategy, where students apply the strategy to divide the model building task into smaller sub-tasks, and worked on different parts of the model separately. So according the corresponding rows in Tables 5.7 and 5.8, we rank this measure for the four clusters as: cluster **2** (number = 15.2, size = 5.6) > cluster **3** (number = 13.8, size = 10.9) > cluster **0** (number = 11.7, size = 15.2) ≈ cluster **1** (number = 10.6, size = 16.5), where the differences were not significant between cluster 0 and 1 for both **Number of ModelEdit chunks** measure ($p = 0.439$) and **Size of ModelEdit chunks** measure ($p = 0.609$). This is also a good match with the performance ranking (i.e., cluster **2** > cluster **3** > cluster **0** > cluster **1**).

Overall, the measures in Table 5.7 are good matches of the performance ranking for the four clusters (i.e., cluster **2** > cluster **3** > cluster **0** > cluster **1**). Among all the measure differences, most of them are significant between pairs of clusters except for cluster 0 and 1. However, there are some measures with significant differences between cluster 0 and 1 such as measures of **SA→SC transition** ($p = 0.04$), **Model Test Effort** ($p = 0.043$) , **Model Comparison Effort** ($p = 0.001$), and **Compare Model in Parts** ($p = 0.0006$). These differences indicate that students in clusters 0 and 1 had different behaviors but similar learning gains $\overline{S_g}$ (4.3 versus 4.5, $p = 0.566$), and significant different model building performance $\overline{S_m}$ (22.6 versus 24.8, $p = 0.034$).

Despite having the best learning gains and model building performance, students in cluster 2 still showed deficiencies in learning with CTSiM. For example, their effort in information acquisition (7.6%) and the percentages of IA→SC (14.5%) and SC→IA (2.5%) transitions were low; their "Compare Model in Part" percentage (29.0%) was not the best. These results showed that the students in cluster 2 still had a lot of room to improve, not to mention students from other groups. These improvements can be achieved either (1) by going through more studies in CTSiM and (2) by getting help from appropriate scaffolding. In the next section, we first take a look at how students' behaviors evolve over time from their work on a later unit (i.e., the macroscopic fish tank unit) in CTSiM.

5.2.1.2 HMM Clustering Results for the "Macroscopic" Unit

Similar to what we did for previous experiments, we derive the plot of Partition Mutual Information (PMI, illustrated in Section 4.2.3) values with respect to different numbers clusters, which is shown in Figure 5.26. The PMI value started to level off when it had 4 clusters. So we chose 4 as the number of clusters to prevent from getting over-complex HMMs.

Next, we run the HMM clustering algorithm to generate four clusters of action sequences that maximizes the PMI value as described in Section 4.2.3. And for each cluster,

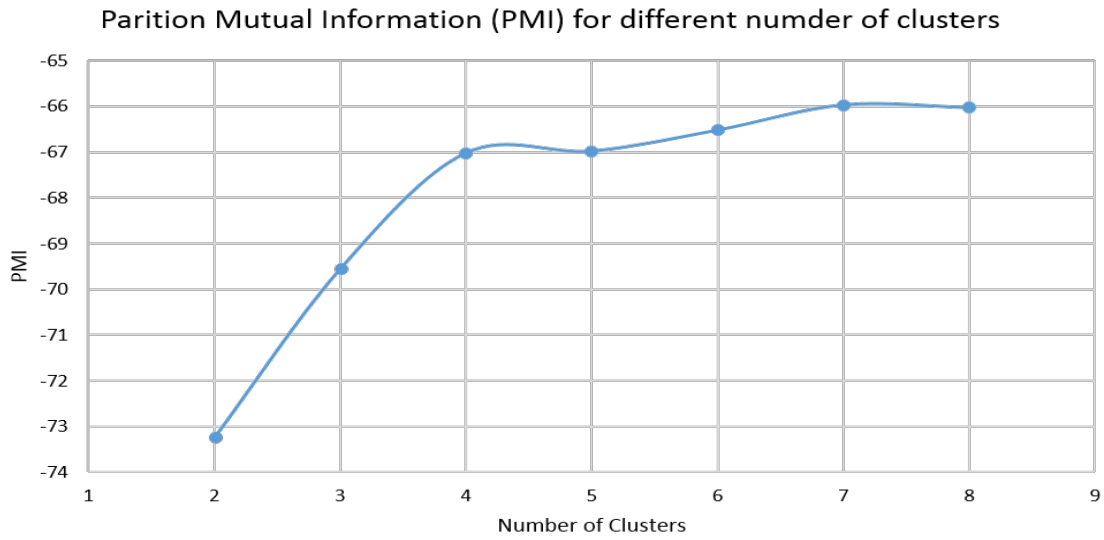


Figure 5.26: Partition Mutual Information values calculated for Data from the **macroscopic** fish tank unit in CTSiM

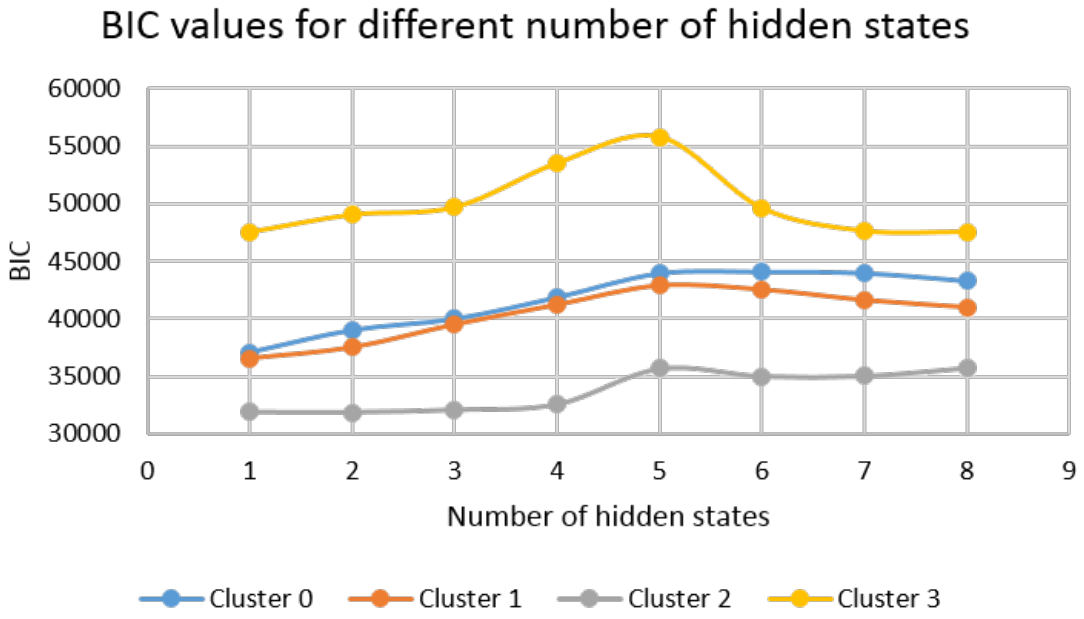


Figure 5.27: Bayesian Information Criterion (BIC) values calculated for the four clusters derived from HMM clustering on data collected from the **macroscopic** fish tank unit in CTSiM. The BIC values were re-scaled to fit the window.

we compute the Bayesian Information Criterion (BIC) to determine the best number of hidden states when learning the HMMs. Figures 5.27 shows the plots of the BIC values

computed for the four derived clusters. As we can see from the figure, the BIC values for all four clusters started to level off with 5 hidden states. Therefore, we chose 5 as the number of hidden states to generate HMMs for the four clusters. Using the determined number of hidden states, we generate the HMMs and present them in in Figure 5.28 (cluster 0), Figure 5.29 (cluster 1), Figure 5.30 (cluster 2), and Figure 5.31 (cluster 3), respectively.

In order to compare HMMs between the **macroscopic** fish tank unit and the **rollercoaster** unit, we use the measures presented in Section 5.2.1.1. The results are shown in Table 5.9. We also perform the pairwise MannWhitney U-Test for each pair of the samples from the four clusters and show the results (p-values) in Table 5.10. The **bolded** p-values are less than 0.05, showing the corresponding differences between the two clusters are significant. For example, the p-value of the **IA effort** measure between cluster 0 and 1 is 0.026, which indicates the difference of the **IA effort** measure is significant between cluster 0 and 1.

As we can see from the results in Tables 5.9 and 5.10:

- The performance measures for the four clusters were: cluster 0 ($\overline{S_g}$ = 13.4, $\overline{S_m}$ = 103.3), cluster 1 ($\overline{S_g}$ = 12.7, $\overline{S_m}$ = 106.9), Cluster 2 ($\overline{S_g}$ = 15.2, $\overline{S_m}$ = 87.0), cluster 3 ($\overline{S_g}$ = 13.5, $\overline{S_m}$ = 98.0).

- By comparing the differences of learning gains between clusters, we can see that students of cluster 0, 1, and 3 improved significantly compared to the **rollercoaster** unit (p-values < 0.05). Students in cluster 2 had the best learning gain ($\overline{S_g}$ = 15.2) followed by cluster 3 ($\overline{S_g}$ = 13.5), cluster 1 ($\overline{S_g}$ = 13.4), and cluster 0 ($\overline{S_g}$ = 12.7). The differences were not significant (p-values > 0.05) except between clusters 0 and 2 (13.4 versus 15.2, $p = 0.031$), as well as between clusters 1 and 2 (12.7 versus 15.2, $p = 0.011$).

- On the other hand, students of cluster 2 who had the highest learning gain, also improved their domain models and achieved the best average model score ($\overline{S_m} =$

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4	π_5
0.01	0.70	0.27	0.02	0

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4	h_5
h_1	0.16	0.17	0.07	0.31	0.28
h_2	0.04	0.90	0.02	0.03	0.01
h_3	0.13	0.09	0.57	0.08	0.13
h_4	0.09	0.03	0.02	0.45	0.41
h_5	0.03	0	0.01	0.29	0.66

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0	0	0.35	0	0
	$p(search)$	0	0	0.03	0	0
SC	$p(SC\_comp)$	0.15	0.78	0.03	0.01	0
	$p(SC\_conc)$	0.01	0.20	0.01	0	0
SA	$p(SA\_compare)$	0.46	0	0.13	0.59	0.80
	$p(SA\_run)$	0.38	0.02	0.46	0.40	0.20

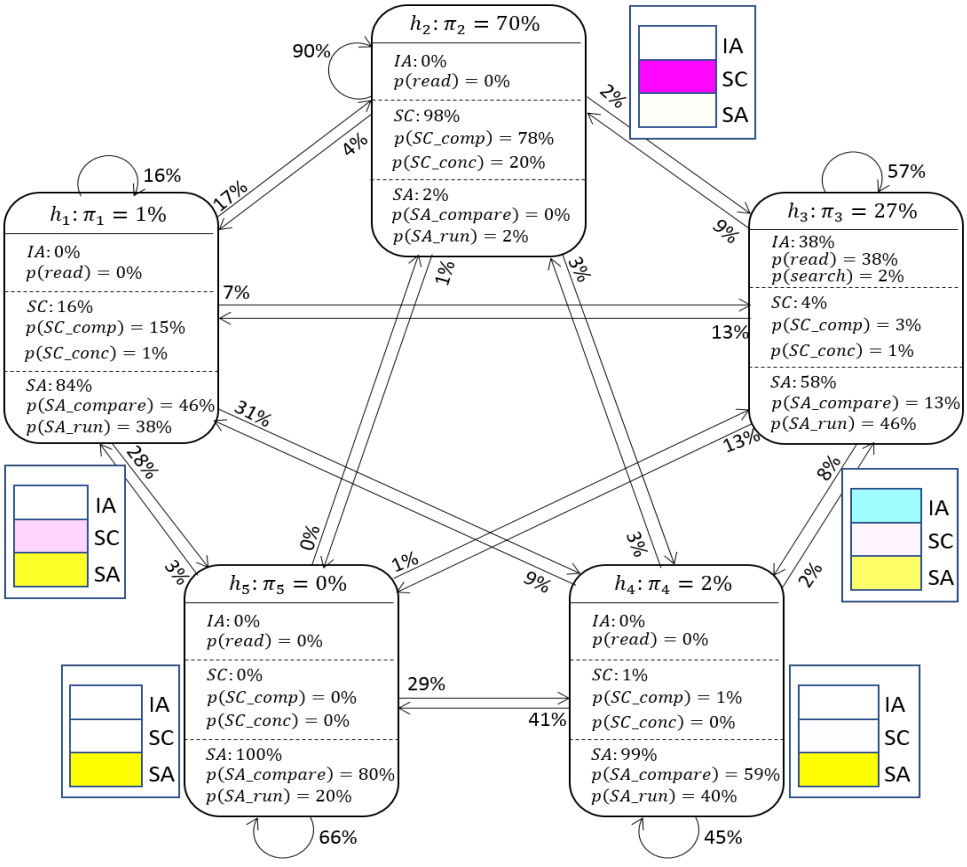


Figure 5.28: The HMM of Cluster 0 (19 students) derived for the **macroscopic** unit

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4	π_5
0.02	0	0.08	0	0.89

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4	h_5
h_1	0.21	0.04	0.22	0.52	0.01
h_2	0.05	0.87	0.03	0.03	0.02
h_3	0.11	0.08	0.09	0.70	0.02
h_4	0.25	0.02	0.33	0.40	0
h_5	0.01	0.07	0.01	0.01	0.89

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0	0	0	0	0.38
	$p(search)$	0	0	0	0	0.04
SC	$p(SC\_comp)$	0.01	0.93	0	0	0.01
	$p(SC\_conc)$	0	0.01	0	0	0.54
SA	$p(SA\_compare)$	0.59	0	0.50	0.69	0
	$p(SA\_run)$	0.39	0.07	0.50	0.31	0.04

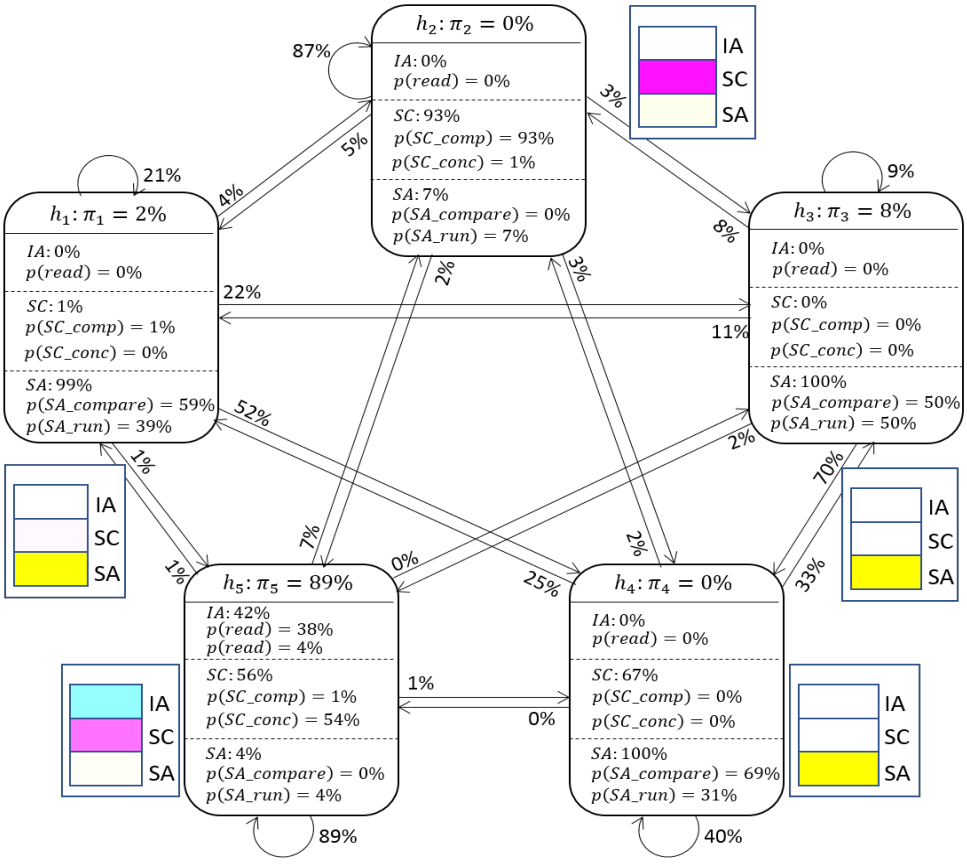


Figure 5.29: The HMM of Cluster 1 (19 students) derived for the **macroscopic** unit

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4	π_5
0	0.89	0.05	0.03	0.02

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4	h_5
h_1	0.86	0.02	0.02	0.01	0.09
h_2	0.05	0.91	0	0	0.03
h_3	0.02	0	0.46	0.46	0.07
h_4	0.01	0	0.69	0.23	0.06
h_5	0.33	0.08	0.11	0.35	0.12

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0	0.42	0	0	0.03
	$p(search)$	0	0.05	0	0	0
SC	$p(SC\_comp)$	0.98	0.02	0	0.72	0.32
	$p(SC\_conc)$	0.01	0.50	0	0.28	0.03
SA	$p(SA\_compare)$	0	0	0.60	0	0.23
	$p(SA\_run)$	0.01	0.01	0.40	0	0.39

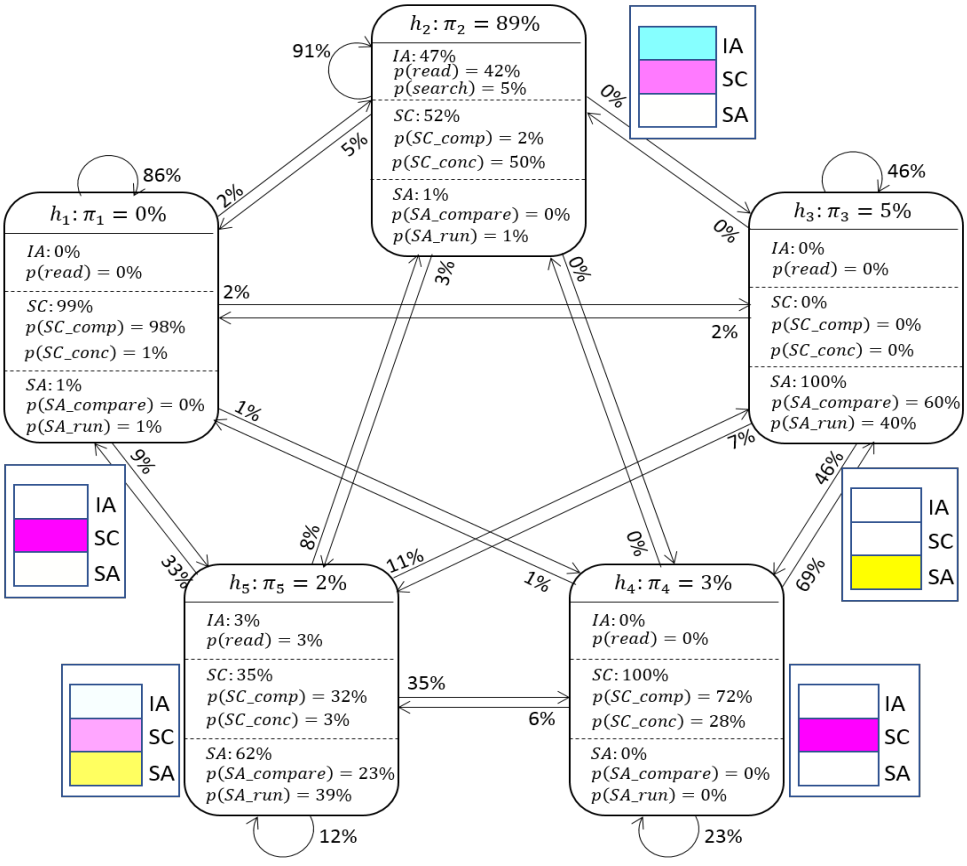


Figure 5.30: The HMM of Cluster 2 (20 students) derived for the **macroscopic** unit

π:

π_1	π_2	π_3	π_4	π_5
0	0	0.56	0.40	0.04

A:

	h_1	h_2	h_3	h_4	h_5
h_1	0.90	0.06	0	0.02	0.02
h_2	0.04	0.83	0	0	0.13
h_3	0.04	0.01	0.86	0.06	0.02
h_4	0.17	0.01	0.13	0.67	0.02
h_5	0.10	0.70	0.04	0.03	0.13

B:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0	0	0	0.52	0
	$p(search)$	0	0	0	0.03	0
SC	$p(SC\_comp)$	0.96	0.01	0	0.08	0.02
	$p(SC\_conc)$	0	0	0.95	0.33	0.01
SA	$p(SA\_compare)$	0	0.53	0.01	0	0.29
	$p(SA\_run)$	0.04	0.46	0.04	0.04	0.68

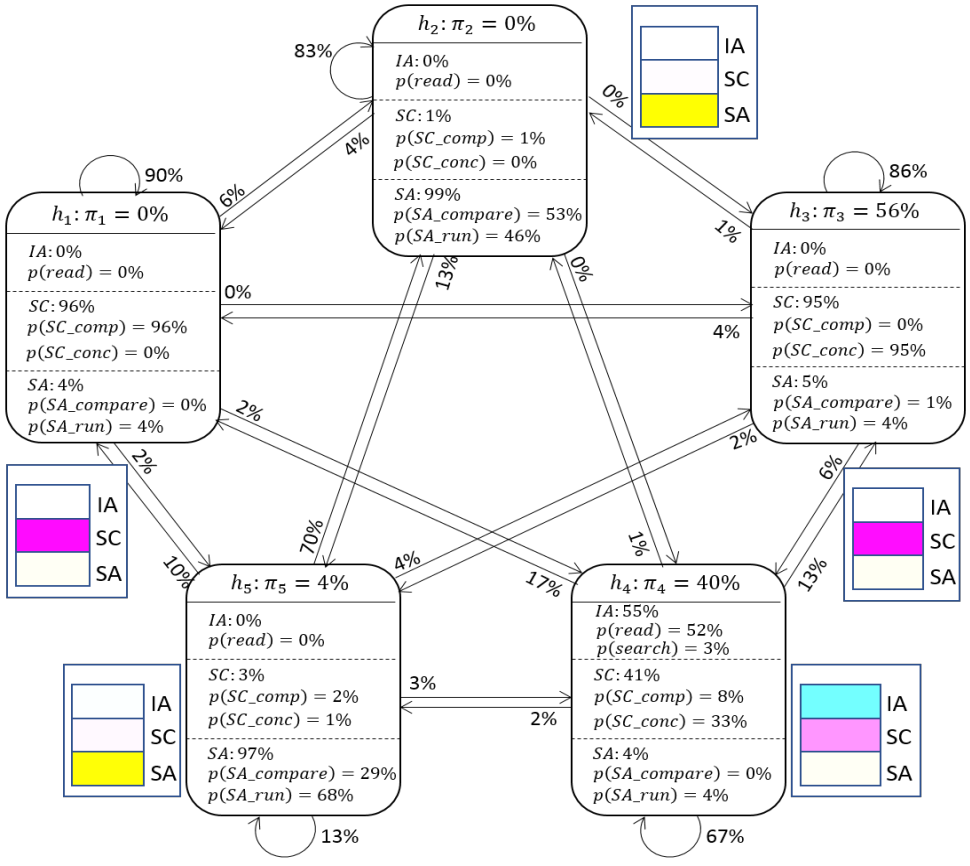


Figure 5.31: The HMM of Cluster 3 (40 students) derived for the **macroscopic** unit

Table 5.9: Comparison of the four Clusters for the **macroscopic** fish tank unit in CTSiM. Results are presented as mean (standard deviation).

	Cluster 0 (n = 19)	Cluster 1 (n = 19)	Cluster 2 (n = 20)	Cluster 3 (n = 40)
Overall IA effort %	1.6 (0.7)	2.3 (1.6)	6.7 (2.1)	4.8 (1.8)
Search Rate %	5.3 (3.6)	10.6 (4.1)	13.2 (4.5)	5.2 (2.9)
IA→SC transitions%	15.7 (11.3)	14.8 (9.9)	30.1 (15.4)	25.1 (13.6)
IA → SC effectiveness %	64.2 (19.8)	62.9 (21.3)	68.7 (24.3)	67.9 (23.6)
SA→SC transitions%	4.4 (2.8)	6.1 (3.6)	5.6 (3.2)	9.0 (4.7)
SA → SC effectiveness %	71.5 (15.6)	70.6 (16.3)	73.2 (16.8)	74.8 (18.2)
SC→IA transitions%	0.8 (1.3)	1.1 (1.1)	1.9 (1.5)	0.7 (1.6)
SC→SA transitions%	11.7 (6.6)	12.6 (7.2)	9.3 (5.8)	9.8 (5.5)
Conceptual Model Edit %	5.8 (1.4)	6.5 (2.1)	9.3 (2.8)	11.7 (2.9)
Computational Model Edit %	23.0 (12.5)	25.2 (13.6)	30.2 (17.2)	36.8 (21.5)
Model Testing effort %	21.7 (19.5)	26.3 (21.0)	20.7 (20.6)	25.5 (22.6)
Model Compare effort %	47.9 (26.3)	37.2 (22.9)	33.1 (20.8)	23.6 (19.3)
Compare Model in Parts %	65.8 (13.6)	65.2 (10.9)	65.6 (12.9)	60.9 (10.4)
Coherence of Model Edits	3.6 (2.7)	3.1 (3.2)	6.6 (4.8)	4.8 (4.2)
Number of ModelEdit chunks	32.7 (6.9)	30.8 (6.1)	45.1 (10.3)	39.5 (8.8)
Size of ModelEdit chunks	9.6 (6.2)	13.1 (6.7)	7.6 (4.4)	7.3 (5.6)
Number of Actions	995 (233)	1128 (261)	795 (183)	600 (152)
$\overline{S_g}$	13.4 (8.9)	12.7 (8.1)	15.2 (9.1)	13.5 (7.6)
$\overline{S_m}$	103.3 (37.6)	106.9 (49.6)	87.0 (27.9)	98.0 (33.2)

87.0), which was significantly better than cluster 0 ($\overline{S_m} = 103.3$, $p = 0.003$), cluster 1 ($\overline{S_m} = 106.9$, $p = 0.008$), and cluster 3 ($\overline{S_m} = 98.0$, $p = 0.029$).

- Students in clusters 2 and 3 were more effective than those in clusters 0 and 1 because they took less number of actions, but achieved better performance. Moreover, students in cluster 0 were more efficient than students in cluster 1 in terms of overall number of actions taken (995 versus 1128, $p = 0.026$).

Overall, based on the performance measures ($\overline{S_g}$ and $\overline{S_m}$) as well as their total effort (**Number of Actions**), we can rank the four clusters in descending order as Cluster 2 > cluster 3 > cluster 0 > cluster 1, which corresponds to the ranking in the **rollercoaster** unit. Next we analyze all the other measures that are used in the **macroscopic** fish tank unit as following:

Table 5.10: Pairwise MannWhitney U-Test result (p-value) for each pair of the samples from the three clusters. The **bolded** p-values are less than 0.05, showing the corresponding differences between the two clusters is significant.

	p-value Cluster 0 and 1	p-value Cluster 0 and 2	p-value Cluster 0 and 3	p-value Cluster 1 and 2	p-value Cluster 1 and 3	p-value Cluster 2 and 3
IA effort	**0.026**	**0.00008**	**0.00001**	**0.002**	**0.002**	0.058
Search Rate	**0.0005**	**0.00007**	0.355	0.082	**0.00001**	**<0.00001**
IA→SC transitions	0.736	**0.0004**	**0.0008**	**0.0007**	**0.002**	**0.023**
IA→SC effectiveness	0.53	0.536	0.603	0.555	0.33	0.246
SA→SC transitions	**0.002**	0.082	**0.002**	0.091	**0.017**	**0.033**
SA→SC effectiveness	0.357	0.6	0.364	0.465	**0.027**	0.259
SC→IA transitions	0.118	**0.0003**	0.635	**0.0001**	**0.05**	**<0.00001**
SC→SA transitions	0.349	0.177	0.163	**0.049**	**0.022**	0.751
Conceptual Model Edit	0.082	**0.018**	**0.00001**	**0.025**	**0.0003**	**0.019**
Computational Model Edit	0.14	**0.005**	**0.00008**	**0.043**	**0.0004**	0.24
Model Test Effort	**0.022**	0.122	0.127	**0.039**	0.277	**0.021**
Model Comparison Effort	**0.034**	**0.025**	**0.00004**	0.088	**0.026**	**0.008**
Compare Model in Parts	0.872	0.715	0.153	0.736	0.206	0.311
Coherence of Model Edits	0.311	**0.0001**	**0.003**	**0.0002**	**0.006**	**0.016**
Number of ModelEdit chunks	0.215	**0.012**	**0.036**	**0.001**	**0.043**	0.252
Size of ModelEdit chunks	**0.009**	**0.031**	**0.034**	**0.001**	**0.0004**	0.66
Number of Actions	**0.026**	0.129	**0.003**	**0.008**	**0.00005**	**0.022**
$\overline{S_g}$	0.895	**0.031**	0.322	**0.011**	0.408	0.059
$\overline{S_m}$	0.715	**0.003**	**0.029**	**0.008**	0.08	**0.034**

Measures of IA effort and Search rate. Students in cluster 2 had the highest **IA effort** (6.7%) and **search rate** (13.2%), which were significantly higher than the other three clusters (p-values < 0.05, as shown in row **IA effort** and **Search rate** in Table 5.8). The **IA effort**, which measures the amount of time students spent on acquiring information from science resource was ranked as: Cluster 2 (6.7%) > cluster 3 (4.8%) > cluster 1 (2.3%) > cluster 0 (1.6%), where for each adjacently ranked clusters, the differences are significant (p-values < 0.05) except between clusters 2 and 3 ($p = 0.058$). Generally speaking, this was a good match with the performance measure, given that students in clusters 0 and 1 had similar performance measures on $\overline{S_g}$ ($p = 0.895$) and $\overline{S_m}$ ($p = 0.715$).

On the other hand, the **search rate** measure, which shows how frequently students actively look for information from the science resource, was ranked as: Cluster 2 (13.2%) > cluster 1 (10.6%) > cluster 0 (5.3%) > cluster 3 (5.2%). The differences were not significant between clusters 2 and 1 ($p = 0.082$), and between clusters 0 and 3 ($p = 0.355$). This did not match perfectly with the performance ranking. However, the **search rate** is the percentage of taking "search" actions among all IA actions. So, with the higher **IA effort** (4.8%), students in cluster 3 still took more numbers of "search" actions than students in clusters 1 and 2, and therefore, still match the performance ranking.

Measures of IA→SC and SA→SC effectiveness. According to row **IA→SC effectiveness** and **SA→SC effectiveness** in Tables 5.9 and 5.10, both of these measures had similar values with differences that were not significant (p-values > 0.05) except the **SA→SC** effectiveness measure between clusters 1 and 3 ($p = 0.027$). This again means that when taking coherently supported SC actions either by information acquisition or solution assessment, students from different groups had similar percentage of correct model building actions. Therefore, it depends more on applying better strategy and taking coherently supported actions, to achieve better learning performance.

Measures of IA→SC and SC→IA transitions. The measure of **IA→SC** transitions, which relates the strategy of using information acquired from science resource to support

model building [68], can be ranked as: Cluster 2 (30.1%) > cluster 3 (25.1%) > cluster 0 (14.7%) ≈ cluster 1 (13.8%), where the differences between clusters 0 and 1 were not significant ($p = 0.736$). This was a good match with the performance ranking. On the other hand, the measure of **SC→IA** transitions, which implies the strategy of seeking information relevant to the part of the model currently being constructed by the student [68], was ranked as: Cluster 2 (1.9%) > cluster 1 (1.1%) > cluster 0 (0.8%) ≈ cluster 3 (0.7%), which was not a good match with the performance ranking. On the other hand, because of the small values (less than 2%), the weights for judging the performance ranking using this measure is also low.

The measure of Compare Model in Parts. Surprisingly, the measure of **Compare Model in Parts**, which is a very useful strategy for model verification and debugging, especially in complex units with multiple agents and multiple agent behaviors [68], had high values for all four clusters (65.8%, 65.2%, 65.6%, and 60.9%, respectively and p-values > 0.05). This is a measure that students from all four clusters did well in applying the strategy, and it did not violate the performance ranking. Higher use of this **Compare Model in Parts** strategy helped students from all four clusters show proficiency in detecting mistakes from their domain models.

The measure of Coherence of Model Edits. This measure shows the correspondence between the students' conceptual and computational models. According to the results in Tables 5.7 and 5.8, we can rank this measure for the four clusters as: cluster **2** (4.2) > cluster **3** (2.9) > cluster **0** (1.1) ≈ cluster **1** (1.3). The p-values between each pair of clusters were significant except between clusters 0 and 1 ($p = 0.358$). So, the results of this measure was a good match of the performance ranking (i.e., cluster **2** > cluster **3** > cluster **0** > cluster **1**).

Measures of Number of ModelEdit chunks and Size of ModelEdit chunks. As discussed earlier, the use of more and smaller ModelEdit chunks indicate a better application of divide-and-conquer strategy, where students apply the strategy to divide the model build-

ing task into smaller sub-tasks, and worked on different parts of the model separately. So according the corresponding rows in Tables 5.9 and 5.10, we ranked this measure for the four clusters as: cluster **2** (number = 45.1, size = 7.6) $\approx$ cluster **3** (number = 39.5, size = 7.3) $>$ cluster **0** (number = 32.7, size = 9.6) $>$ cluster **1** (number = 30.8, size = 13.1), where the differences were not significant between cluster 2 and 3, for both **Number of ModelEdit chunks** measure (p = 0.252) and **Size of ModelEdit chunks** measure (p = 0.66). This was also a good match with the performance ranking (i.e., cluster **2** $>$ cluster **3** $>$ cluster **0** $>$ cluster **1**).

Overall, the measures for the **macroscopic** fish tank unit shown in Table 5.9, are good matches of the performance ranking for the four clusters (i.e., cluster **2** $>$ cluster **3** $>$ cluster **0** $>$ cluster **1**). Among all the measure differences, most of them were significantly between each pair of clusters except for clusters 0 and 1. On the other hand, only some measures had significant differences between cluster 0 and 1 such as measures of **IA effort** ($p = 0.026$), **Search Rate** ($p = 0.0005$) , **SA$\rightarrow$SC transition** ($p = 0.002$), **Model test effort** (p = 0.022), **Model comparison effort** (p = 0.34), and **Size of ModelEdit chunks** (p = 0.009). As discussed earlier, these differences show the students in clusters 0 and 1 had different behaviors, but similar learning gains $\overline{S_g}$ (13.4 versus 12.7, p = 0.895) and model building performance $\overline{S_m}$ (103.3 versus 106.9, p = 0.715).

5.2.1.3 Comparison of the HMM Clustering results between the "Rollercoaster" and "Macroscopic" units

Since the study for both **rollercoaster** and **macroscopic** fish tank units consists of the same students and both data sets produced the same number of clusters, we checked if students transitioned from lower ranking clusters (in the **rollercoaster** unit) to higher ranking clusters (in the **macroscopic** fish tank unit), and vice versa.

Table 5.11 shows the number of students who transitioned across the four clusters. The number in the entry (i, j) means the number of students transitioned from cluster i (in the

Table 5.11: Cluster transitions from **rollercoaster** unit to **macroscopic** fish tank unit

macroscopic / **rollercoaster**	Cluster 0 (n = 19)	Cluster 1 (n = 19)	Cluster 2 (n = 20)	Cluster 0 (n = 40)
Cluster 0 (n = 33)	14 -	6 ↓	9 ↑	4 ↑
Cluster 1 (n = 23)	3 ↑	7 -	2 ↑	11 ↑
Cluster 2 (n = 4)	0	0	3 -	1 ↓
Cluster 3 (n = 38)	2 ↓	6 ↓	6 ↑	24 -

rollercoaster unit) to cluster *j* (in the **macroscopic** fish tank unit). For example, 9 students transitioned from cluster 0 (in the **rollercoaster** unit) to cluster 2 (in the **macroscopic** fish tank unit). For both units, the performance ranking of the students is the same (i.e., cluster **2** > cluster **3** > cluster **0** > cluster **1**). We use the "↑" / "↓" arrows to indicate the transitions to higher / lower ranking clusters, while "-" means no transitions were made. Overall, 35 (35.7%) students transitioned to better clusters, 48 (50.0%) students stayed in the same cluster, and 15 (15.3%) of them went into lower-ranking clusters. However, this doesn't necessarily mean that the 15 students showed worse behaviors in the **macroscopic** fish tank unit, but they did not improve as much as other students.

We have compared the measures horizontally between clusters for both the **rollercoaster** and the **macroscopic** fish tank units. We are also interested to see how students' learning behaviors evolved over time. So we compared the measures vertically between the "Rollercoaster" unit (Table 5.7) and the "Macroscopic" unit(Table 5.9). As we can see from the tables:

- Although no significant improvements for the IA effort can be observed, students of all clusters took a lot more search actions (cluster 0: 2.3% $\rightarrow$ 5.3%, $p = 0.001$; cluster 1: 1.9% $\rightarrow$ 10.6%, $p < 0.00001$; cluster 2: 8.2% $\rightarrow$ 13.2% $p = 0.013$; cluster 3: 1.4% $\rightarrow$ 5.2%, $p = 0.0008$). This indicates that the students in the later unit, became more active in looking for the information about the specific knowledge they were unsure about or information that were needed for building the domain model.

- Students in all the clusters, especially for cluster 2 and 3, made more IA$\rightarrow$SC transi-

tions (cluster 0: 14.4% $\rightarrow$ 15.7%, p = 0.123; cluster 1: 12.7% $\rightarrow$ 14.8% p = 0.044; cluster 2: 14.5% $\rightarrow$ 30.1%, $p < 0.00001$; cluster 3: 15.3% $\rightarrow$ 25.1%, $p < 0.00001$). This indicates that students in the later unit (**macroscopic**) applied more of the strategy that uses the acquired information from science resources to support their model building actions [68].

- The percentage of SA$\rightarrow$SC transitions of all the clusters decreased significantly (cluster 0: 7.6% $\rightarrow$ 4.4%, p = 0.004; cluster 1: 10.0% $\rightarrow$ 6.1%, p = 0.005; cluster 2: 11.0% $\rightarrow$ 5.6% p = 0.001; cluster 3: 12.9% $\rightarrow$ 9.0%, p = 0.015). There are two potential reasons: (1) the students applied less trial-and-error in building the domain models; and (2) the chunk size of SA actions increased due to the increment of model complexity in **macroscopic** fish tank unit (there are more agents than in the **rollercoaster** unit). Whenever a student want to verify their model, he/she need to repeatedly run the simulation to compare different parts of the model with the expert model, so as to know what part(s) went wrong.

- Another important improvement for all the students was that they performed significantly more **Compare Model in Parts** strategy when doing **SA_compare** actions (cluster 0: 34.2% $\rightarrow$ 65.8%, $p < 0.00001$; cluster 1: 24.8% $\rightarrow$ 65.2%, $p < 0.00001$; cluster 2: 29.0% $\rightarrow$ 65.6%, $p < 0.00001$; cluster 3: 23.6% $\rightarrow$ 60.9%, $p < 0.00001$). This divide-and-conquer strategy applied to solution assessment effectively helped them detect the incorrect **parts** in their domain models. Some of the previous work in our group have shown that using more of the "Compare Model in Parts" action can help students perform better in building the domain model [70, 68, 54].

These improvements showed the actual learning progress that students achieved on different units over time. Despite having these improvements in the performance and the coherent measures, however, certain deficiencies still existed for students from all the clusters:

- **Students in the later unit still performed very few IA actions** (cluster 0: 1.4% $\rightarrow$ 1.6%, p = 0.245; cluster 1: 1.7% $\rightarrow$ 2.3% p = 0.038; cluster 2: 7.6% $\rightarrow$ 6.7%, p = 0.125; cluster 3: 3.7% $\rightarrow$ 4.8%, p = 0.04). The information acquisition actions helped them not only learn, but also build better domain models. This can be seen by the performance outcome of the students in cluster 2, who took the most IA actions and achieved better learning gain and model building performance. Figure 5.32 illustrates the effectiveness percentage of SC actions with respect to the percentage of taking IA actions in different learning stages (estimated by a scale of every 200 actions) for all the students. As we can see, there were improvements from the **roller-coaster** unit to the **macroscopic** fish tank unit. However, these improvements were insufficient to help them build completely correct domain models. Specifically, for the SC effectiveness percentage of under 0.5, it is possible that some of the students' performance on the domain model building became worse in later stages (may not be true for all the scenarios because one effective SC action, e.g., the RemoveBlock action, can correct multiple ineffective SC actions made before). Comparing this plot with the change of usage percentage for IA actions, it is reasonable to believe that the effectiveness percentage of SC actions is partially dependent on the use of IA actions.

- The percentage of IA$\rightarrow$SC transitions were still low for cluster 0 (14.4% $\rightarrow$ 15.7%, p = 0.123), cluster 1(12.7% $\rightarrow$ 14.8% p = 0.044), cluster 2 (14.5% $\rightarrow$ 30.1%, $p <$ 0.00001), and cluster 3 (15.3% $\rightarrow$ 25.1%, $p <$ 0.00001), even with significant improvements for most of them. It would be better if the students used more information acquired from science resources to support their subsequent solution construction actions.

- In addition, the percentage of **SC$\rightarrow$IA** transitions (cluster 0: 0.6% $\rightarrow$ 0.8%, p = 0.031; cluster 1: 0.5% $\rightarrow$ 1.1%, p = 0.0008; cluster 2: 2.5% $\rightarrow$ 1.9%, p = 0.013;

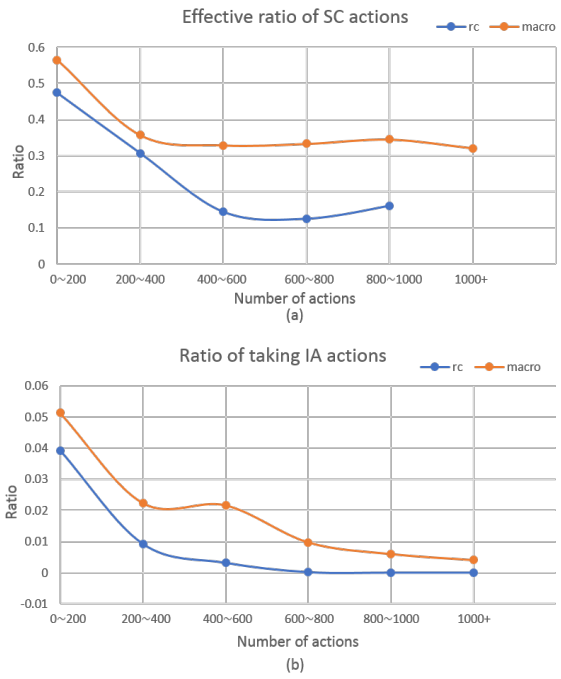


Figure 5.32: Effective percentage of SC actions (a) and percentage of taking IA actions (b) of all the students during their study in **rollercoaster** and the **macroscopic** fish tank (macro) units. X-axis is the number of actions taken during the study.

cluster 3: 0.7% $\rightarrow$ 0.7%, $p = 0.725$) were still low for most of the students. According to the use of **Compare Model in Parts** in **macroscopic** fish tank unit, we can see that the students have already figured it out that divide-and-conquer strategy can help them detect mistakes. Therefore, it is important for them to apply the same strategy more effectively for model building. This strategy should be applied to divide the modeling task into smaller sub-tasks. And they should use a lot more coherent actions, especially actions that are coherently supported by information acquisition actions, to build different parts of the domain model.

- **The overall effort for taking SA actions for all the students were too high.** This value is computed by the sum of **Model Testing effort** and **Model Compare effort** (cluster 0: 67.5% $\rightarrow$ 69.6%, $p = 0.508$; cluster 1: 63.4% $\rightarrow$ 63.5%, $p = 0.735$; cluster 2: 40.3% $\rightarrow$ 53.8%, $p = 0.0003$; cluster 3: 47.6% $\rightarrow$ 49.1%, $p = 0.108$).

These results show that the students spent too much time on acquiring information by checking the correctness of the domain models, rather than acquiring information from the science resource. Solely depending on information derived from model simulation without understanding of the knowledge, may result in frequent use of trial-and-error approaches, which had negative impact on model building tasks.

Given these deficiencies, we believe that there are better ways to help the students become good learners. By providing proper adaptive scaffolding, the students should be able to improve their learning and model building strategies that have not been achieved through their self-improvements over time (e.g., the improvements from **rollercoaster** unit to **macroscopic** fish tank unit). So next, we apply the reinforcement scaffolding approach to data from the two units and discuss the results as well as potential adaptive scaffolds generated based on the reinforced models.

5.2.2 Analysis of Reinforced Classification Model

First, we apply the same methods used for the Betty's Brain Experiments (Section 5.1.2), to derive the reinforced classification HMMs for the **rollercoaster** unit, which are shown in Figure 5.33 (cluster 0), Figure 5.34 (cluster 1), Figure 5.35 (cluster 2), and Figure 5.36 (cluster 3).

Table 5.12 shows the comparison of the measures excluding those related to learning performance, which we could not compute for the reinforced models. In the table, the **bolded** entries are those with significant change pre- and post-reinforcement learning (p-values < 0.05). For example, the change of **IA effort** measure for cluster 2 pre- and post-reinforcement learning was significant ($p = 0.003$).

As we can see from Table 5.12, the differences for most measures that pre- and post-reinforcement learning were not significant (p values > 0.05), with only a few exceptions (bolded entries in the table). The parameters in the reinforced classification HMMs were tuned by the reinforcement learning, which we believe can achieve better classification

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4	π_5
0.15	0.75	0.06	0.02	0.02

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4	h_5
h_1	0.09	0.16	0.38	0.20	0.27
h_2	0.03	0.89	0.01	0	0.08
h_3	0.08	0.01	0.62	0.19	0.11
h_4	0.13	0.06	0.39	0.30	0.11
h_5	0.31	0.07	0.41	0.18	0.03

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0	0.04	0	0	0.01
	$p(search)$	0	0.01	0	0	0
SC	$p(SC\_comp)$	0	0.73	0	0.11	0.10
	$p(SC\_conc)$	0	0.15	0	0.01	0.03
SA	$p(SA\_compare)$	0.12	0.02	0.61	0.41	0.71
	$p(SA\_run)$	0.88	0.05	0.38	0.47	0.15

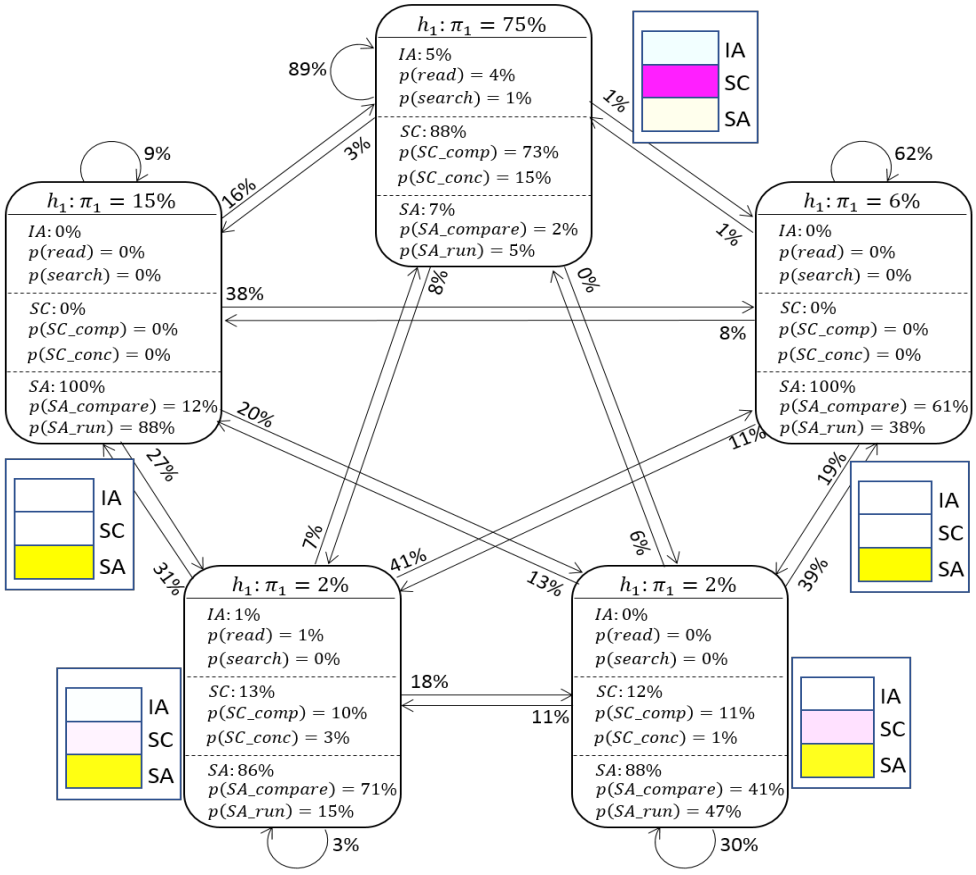


Figure 5.33: The reinforced classification HMM of Cluster 0 for the **rollercoaster** unit

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4	π_5
0.07	0.04	0	0.89	0

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4	h_5
h_1	0.25	0.22	0.18	0.21	0.14
h_2	0.11	0.75	0.05	0.04	0.05
h_3	0.09	0.12	0.49	0.05	0.25
h_4	0.09	0.19	0.13	0.39	0.20
h_5	0.21	0.03	0.41	0.05	0.20

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0.05	0	0	0.11	0
	$p(search)$	0.01	0	0	0.01	0
SC	$p(SC\_comp)$	0.10	0.95	0	0	0.02
	$p(SC\_conc)$	0.11	0	0	0.21	0.05
SA	$p(SA\_compare)$	0.17	0.05	0.40	0.12	0.27
	$p(SA\_run)$	0.54	0.01	0.59	0.55	0.66

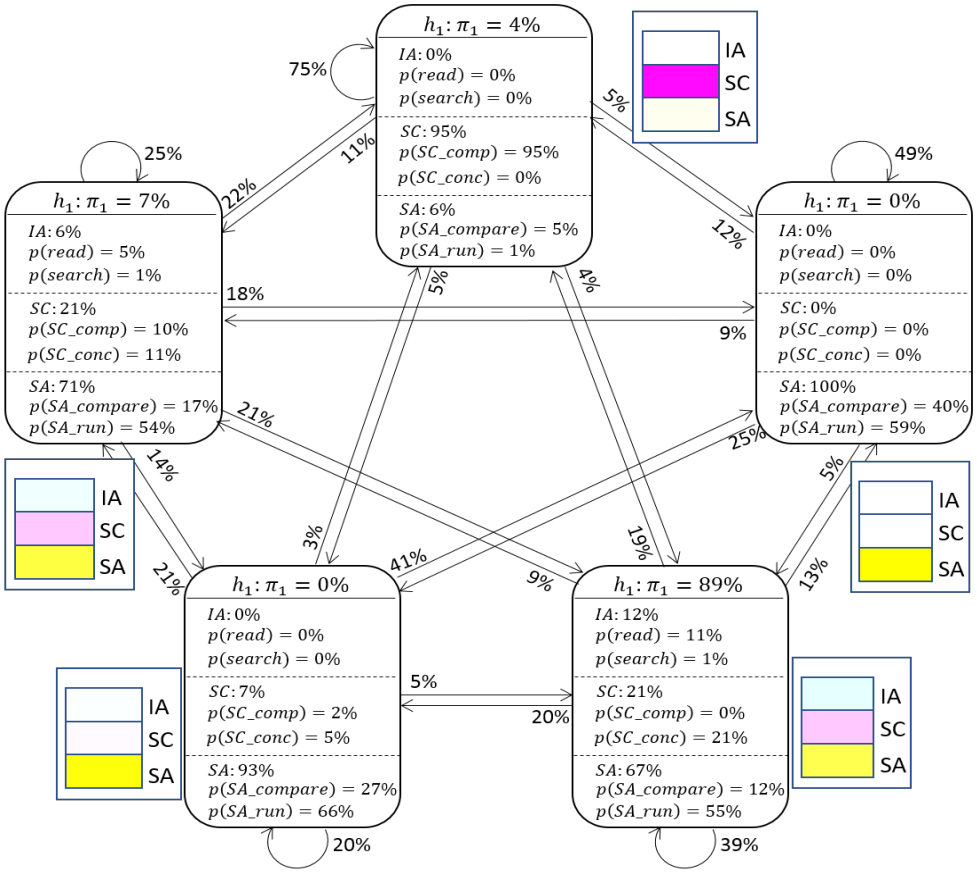


Figure 5.34: The reinforced classification HMM of Cluster 1 for the **rollercoaster** unit

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4	π_5
0.91	0.01	0.01	0.01	0.06

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4	h_5
h_1	0.79	0.11	0.03	0.02	0.05
h_2	0.03	0.52	0.10	0.11	0.23
h_3	0.05	0.11	0.82	0.03	0
h_4	0.22	0.17	0.21	0.13	0.27
h_5	0.01	0.62	0.03	0.06	0.28

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0.51	0	0	0.15	0
	$p(search)$	0.08	0	0	0.02	0
SC	$p(SC\_comp)$	0.09	0.81	0	0.71	0.72
	$p(SC\_conc)$	0.28	0.12	0	0.10	0.12
SA	$p(SA\_compare)$	0	0.03	0.29	0	0.04
	$p(SA\_run)$	0.04	0.04	0.70	0.02	0.12

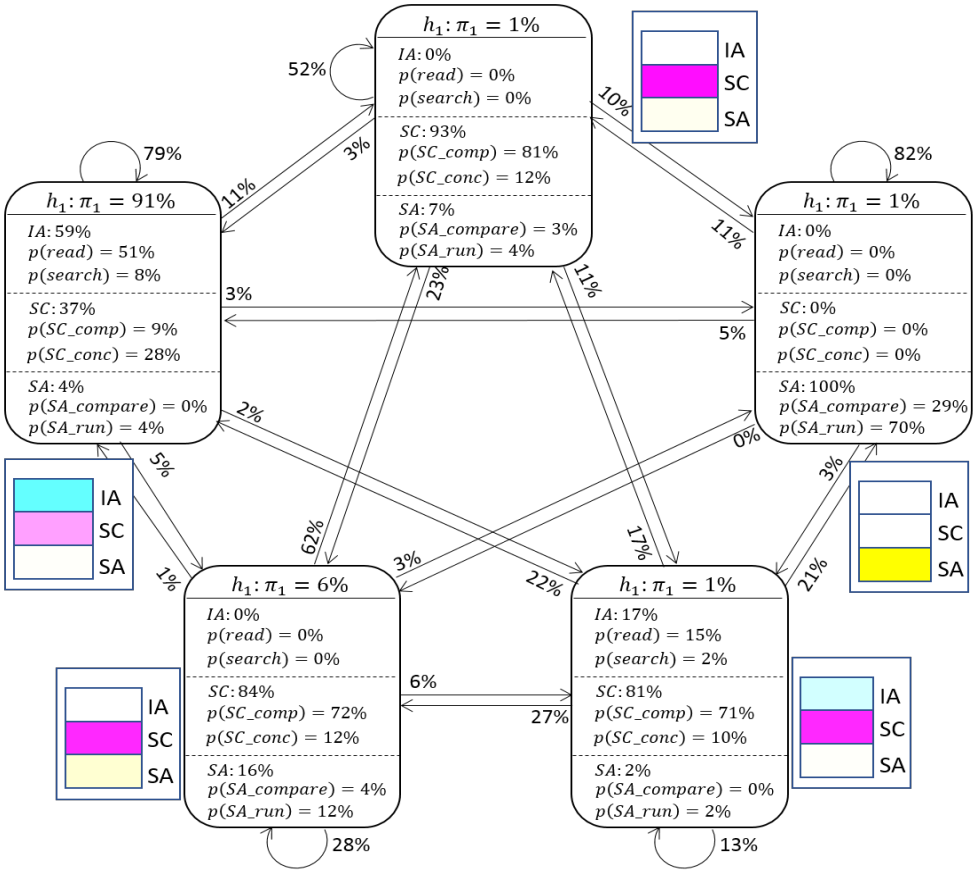


Figure 5.35: The reinforced classification HMM of Cluster 2 for the **rollercoaster** unit

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4	π_5
0.12	0.02	0.01	0.83	0.02

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4	h_5
h_1	0.27	0.32	0.27	0.05	0.09
h_2	0.21	0.35	0.35	0.03	0.06
h_3	0.38	0.51	0.01	0	0.11
h_4	0.05	0.07	0.09	0.58	0.21
h_5	0.01	0.15	0.09	0.06	0.69

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0	0	0	0.44	0.03
	$p(search)$	0	0	0	0.03	0.01
SC	$p(SC\_comp)$	0.06	0.09	0	0	0.88
	$p(SC\_conc)$	0.01	0.03	0	0.52	0.03
SA	$p(SA\_compare)$	0.33	0.39	0.15	0	0.02
	$p(SA\_run)$	0.60	0.50	0.85	0	0.03

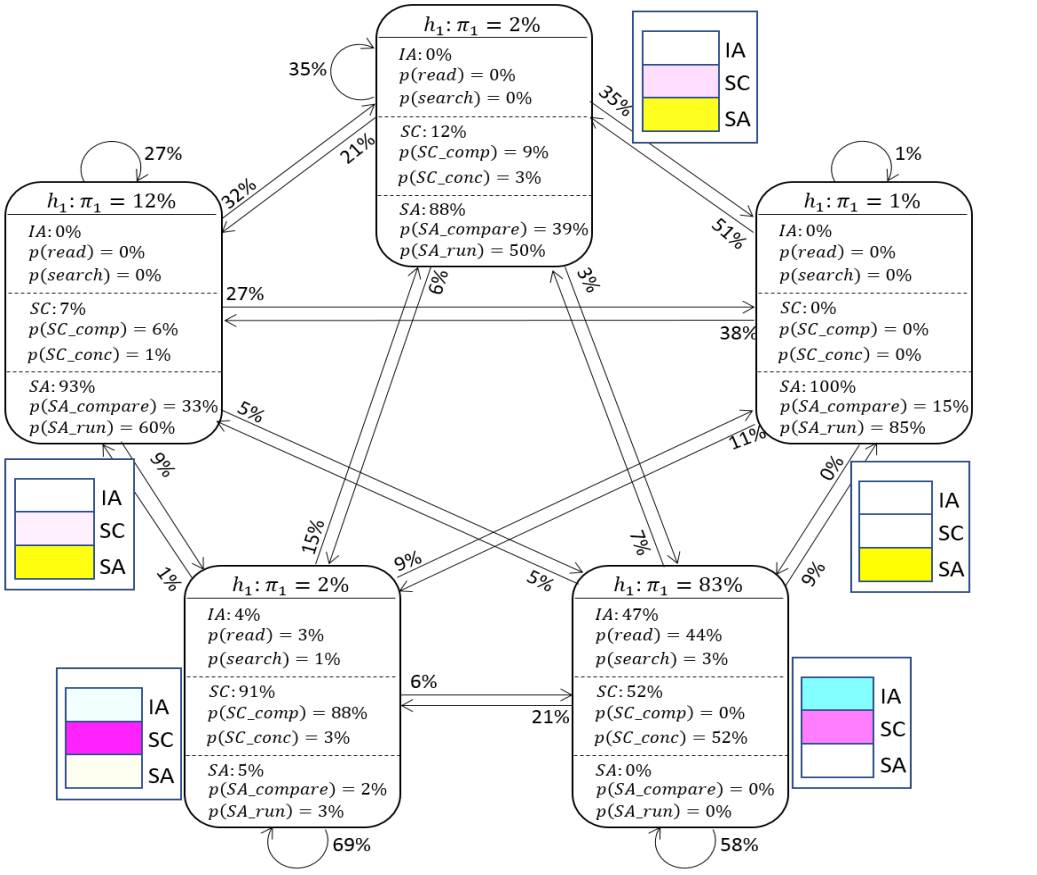


Figure 5.36: The reinforced classification HMM of Cluster 3 for the **rollercoaster** unit

Table 5.12: Comparison of the four Clusters post-reinforcement classification learning for the **rollercoaster** unit in CTSiM. The **bolded** entries are those with significant change pre- and post-reinforcement learning (p-values < 0.05).

	Cluster 0 (n = 33)	Cluster 1 (n = 23)	Cluster 2 (n = 4)	Cluster 3 (n = 38)
Overall IA effort %	1.2 (2.1)	1.5 (1.9)	**9.1 (3.6)**	4.1 (3.1)
Search Rate %	2.2 (1.8)	**2.6 (1.9)**	8.5 (3.4)	2.1 (1.3)
IA→SC transitions%	12.8 (11.7)	13.4 (8.5)	16.1 (9.5)	**19.1 (10.2)**
SA→SC transitions%	8.1 (3.9)	9.5 (5.5)	**12.2 (7.6)**	11.9 (6.1)
SC→IA transitions%	**0.1 (0.3)**	0.6 (0.9)	3.0 (1.2)	0.9 (0.4)
SC→SA transitions%	18.1 (6.7)	18.8 (6.0)	12.8 (5.8)	15.3 (7.3)
Conceptual Model Edit %	**5.4 (3.9)**	4.3 (3.2)	**11.5 (4.0)**	8.8 (3.0)
Computational Model Edit %	**31.2 (19.6)**	29.7 (13.5)	35.7 (15.6)	42.3 (22.9)
Model Testing effort %	34.6 (22.5)	39.9 (26.3)	25.4 (13.5)	32.4 (20.9)
Model Comparison effort %	36.7 (23.9)	22.2 (11.1)	13.7 (5.2)	**10.9 (6.6)**
Compare Model in Parts %	34.2 (16.4)	25.6 (11.8)	31.8 (13.9)	28.2 (22.4)
Coherence of Model Edits	0.9 (2.5)	**0.9 (1.6)**	3.9 (1.9)	3.1 (2.2)
Number of ModelEdit chunks	10.9 (6.0)	9.9 (4.7)	15.6 (5.7)	14.2 (6.2)
Size of ModelEdit chunks	14.8 (9.9)	17.2 (10.5)	5.5 (1.7)	11.3 (6.3)

accuracy. In order to verify the improvements in the classification accuracy. We ran the leave-one-out cross-validation (LOOCV) on the original data set, and the expanded data set used to learn the updated HMMs. We compared the results of LOOCVs performed on the two data sets to show the improvements in classification accuracy.

In every iteration of the LOOCV, we took one action sequence s_a from cluster C_i of the original student data out and performed the reinforcement learning using the rest of the data to generate four updated HMMs (i.e., HMMs for the four clusters discussed above). The action sequence s_a was assigned with classification label j if the **reinforced** HMM of cluster C_j had the highest log-likelihood value for s_a. If j is the same as the **original** cluster label (i.e., i) that s_a belongs to, we say the classification is accurate. The average classification accuracy of all students' action sequences was computed for the LOOCVs.

The average classification accuracy of LOOCV on the original data set was 0.62 (in the **rollercoaster** unit) which is even lower than the result of Betty's Brain data (0.68). Because of the complexity of the CTSiM system, there are more available actions that can

be taken than that in the Betty's Brain system. As we can see from Tables 5.6 and 3.1, there are 32 available actions for CTSiM versus 18 for Betty's Brain. The increased number of actions/observations results in the more complex HMMs. So when the sample size is small (i.e., 98 students), the HMMs may overfit the data and lead to low LOOCV accuracy.

On the other hand, after applying reinforcement learning to generate additional action sequences that grew the data set to four times of its original, the LOOCV accuracy of reinforced classification HMMs for the **rollercoaster** unit have been increased to 0.81 (**0.62** $\rightarrow$ **0.81**). This result showed significant improvement in classifying students into the correct groups, which is helpful in providing the scaffolding that adapts students with different learning behaviors. Next, we derive and analyze the reinforced scaffolding models for the **rollercoaster** and the **macroscopic** fish tank units, and use the results to derive example scaffolds.

5.2.3 Analysis of Reinforced Scaffolding Model

We ran the reinforcement learning algorithm as illustrated in Section 4.3.3 to **extend** the **existing** action sequences for the **rollercoaster** unit. The extended action sequences were then used to learn updated HMMs as the reinforced scaffolding model. The reinforced scaffolding models for the **rollercoaster** unit are shown in Figure 5.37 (cluster 0), Figure 5.38 (cluster 1), Figure 5.39 (cluster 2), and Figure 5.40 (cluster 3).

In order to compare the reinforced scaffolding HMMs with the original HMMs for the **rollercoaster** unit, we generated results of the same measures from the extended action sequences that were used to learn reinforced HMMs and compared them with the results from the original data shown in Table 5.7. The newly generated results of the measures for the **rollercoaster** unit are shown in Table 5.13. The learning performance measures ($\overline{S_g}$ and $\overline{S_m}$) were excluded here because there is no accurate way to compute the real learning performance out of a virtual agent informed by reinforcement learning.

Instead of horizontal comparison of the measures between different clusters, we are

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4	π_5
0.77	0.13	0.07	0.03	0

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4	h_5
h_1	0.23	0.57	0.06	0.12	0.02
h_2	0.16	0.45	0.32	0.03	0.03
h_3	0.15	0.23	0.33	0.18	0.11
h_4	0.15	0.12	0.26	0.34	0.13
h_5	0	0.01	0.05	0.05	0.89

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0.57	0.12	0.15	0.11	0.02
	$p(search)$	0.03	0.04	0.05	0.03	0
SC	$p(SC\_comp)$	0.12	0.44	0.15	0.22	0.05
	$p(SC\_conc)$	0.26	0.22	0.16	0.06	0
SA	$p(SA\_compare)$	0.01	0.11	0.24	0.36	0.51
	$p(SA\_run)$	0.01	0.07	0.25	0.22	0.42

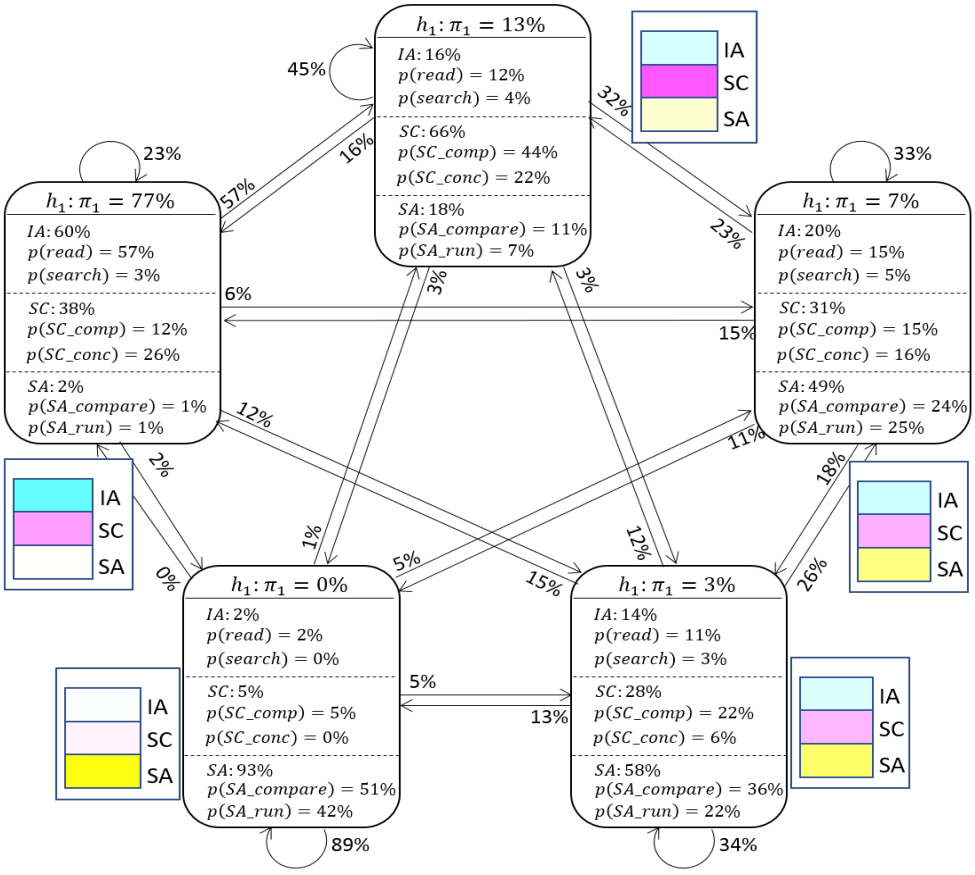


Figure 5.37: The reinforced scaffolding HMM of Cluster 0 for the **rollercoaster** unit

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4	π_5
0.81	0.11	0.06	0.02	0

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4	h_5
h_1	0.34	0.44	0.11	0.10	0.01
h_2	0.23	0.33	0.23	0.13	0.08
h_3	0.11	0.33	0.36	0.11	0.09
h_4	0.12	0.15	0.21	0.39	0.13
h_5	0	0.01	0.05	0.05	0.89

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0.51	0.17	0.29	0.05	0.01
	$p(search)$	0.02	0.03	0.04	0.05	0
SC	$p(SC\_comp)$	0.09	0.23	0.12	0.36	0.03
	$p(SC\_conc)$	0.27	0.36	0.29	0.03	0
SA	$p(SA\_compare)$	0.05	0.08	0.12	0.44	0.65
	$p(SA\_run)$	0.08	0.13	0.14	0.07	0.31

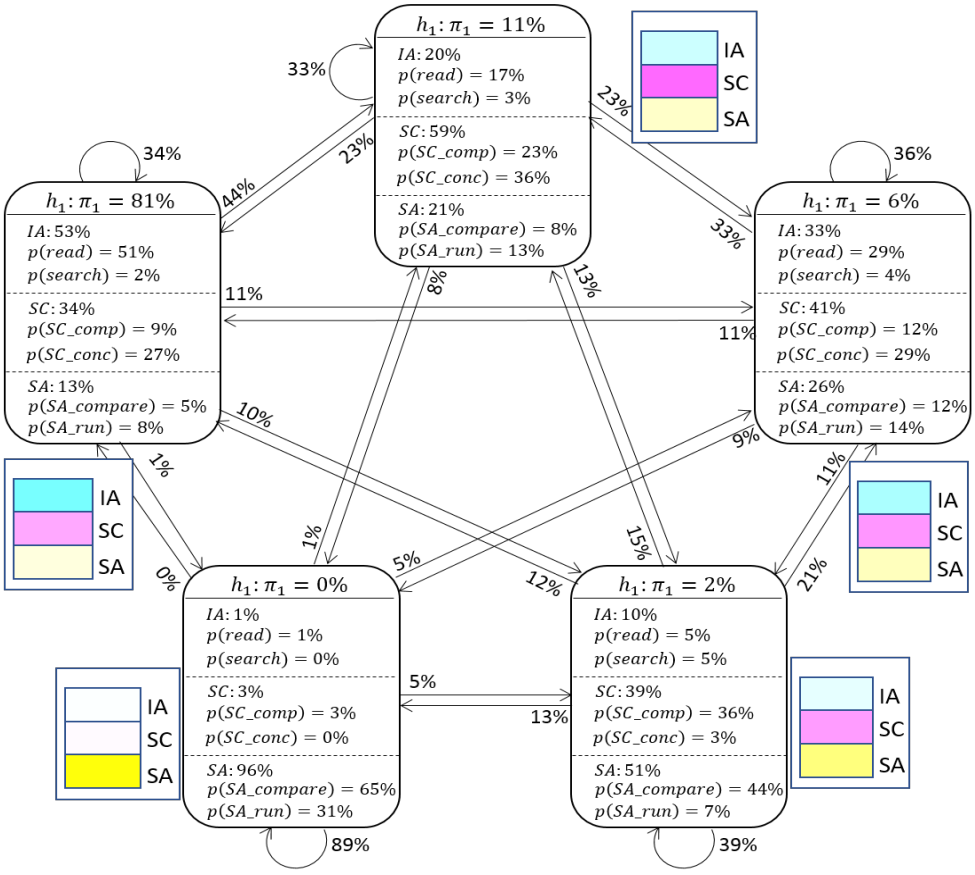


Figure 5.38: The reinforced scaffolding HMM of Cluster 1 for the **rollercoaster** unit

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4	π_5
0.67	0.18	0.10	0.04	0.01

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4	h_5
h_1	0.43	0.21	0.13	0.15	0.08
h_2	0.31	0.31	0.12	0.16	0.10
h_3	0.05	0.23	0.51	0.16	0.05
h_4	0.10	0.09	0.16	0.52	0.12
h_5	0.02	0.06	0.06	0.07	0.79

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0.66	0.21	0.24	0.18	0.02
	$p(search)$	0.06	0.15	0.07	0.06	0.03
SC	$p(SC\_comp)$	0.03	0.15	0.25	0.29	0.09
	$p(SC\_conc)$	0.11	0.21	0.17	0.13	0.01
SA	$p(SA\_compare)$	0.05	0.11	0.22	0.33	0.63
	$p(SA\_run)$	0.09	0.17	0.05	0.01	0.22

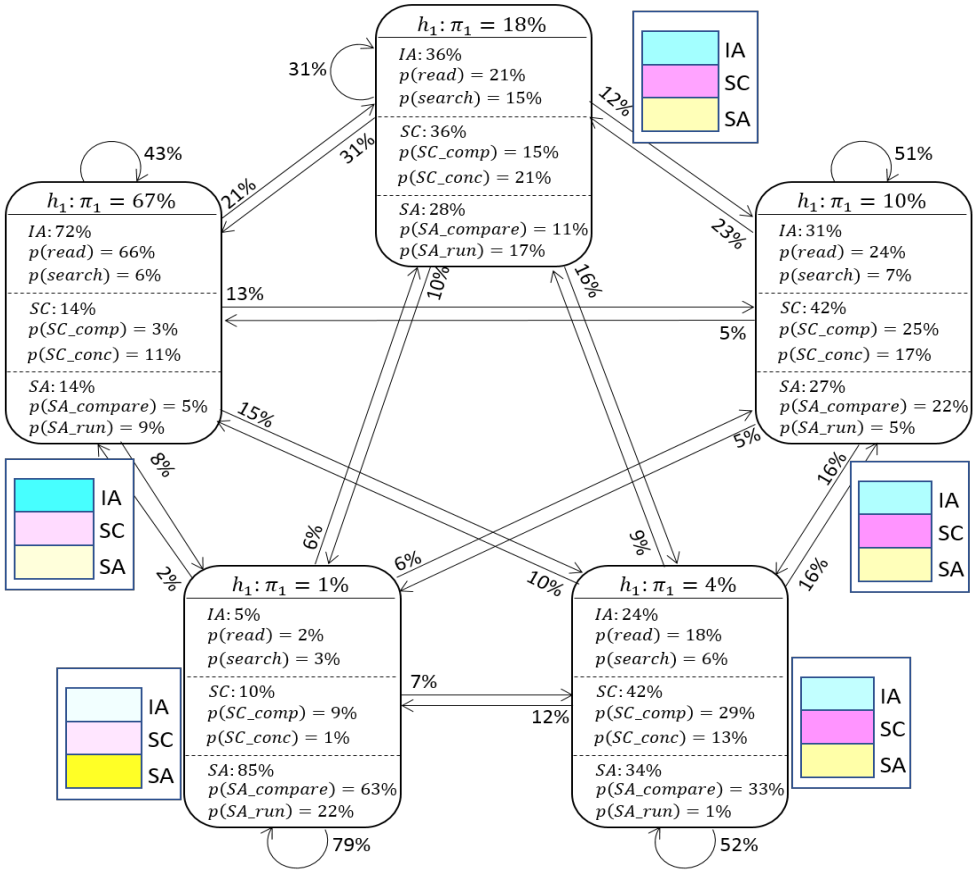


Figure 5.39: The reinforced scaffolding HMM of Cluster 2 for the **rollercoaster** unit

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4	π_5
0.63	0.16	0.13	0.07	0.01

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4	h_5
h_1	0.46	0.17	0.16	0.16	0.05
h_2	0.21	0.21	0.26	0.27	0.05
h_3	0.14	0.08	0.66	0.03	0.09
h_4	0.21	0.10	0.14	0.46	0.09
h_5	0.03	0.04	0.06	0.06	0.81

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0.41	0.18	0.11	0.05	0.01
	$p(search)$	0.02	0.02	0.03	0.01	0
SC	$p(SC\_comp)$	0.19	0.15	0.37	0.26	0.05
	$p(SC\_conc)$	0.25	0.28	0.14	0.05	0.02
SA	$p(SA\_compare)$	0.01	0.10	0.29	0.48	0.77
	$p(SA\_run)$	0.12	0.27	0.06	0.15	0.15

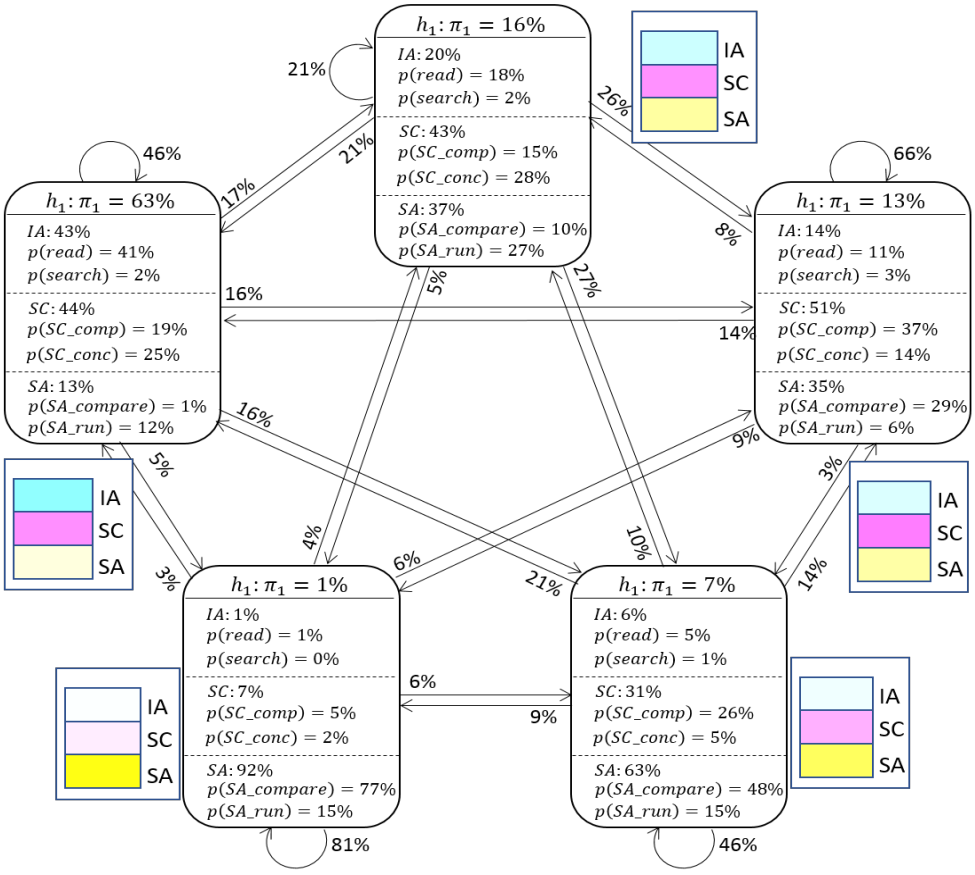


Figure 5.40: The reinforced scaffolding HMM of Cluster 3 for the **rollercoaster** unit

Table 5.13: Comparison of the four Clusters post-reinforcement scaffolding learning for the **rollercoaster** unit in CTSiM. Results are presented as mean (standard deviation).

	Cluster 0 (n = 33)	Cluster 1 (n = 23)	Cluster 2 (n = 4)	Cluster 3 (n = 38)
Overall IA effort %	15.2 (5.4)	10.1 (3.3)	17.0 (5.9)	11.8 (4.9)
Search Rate %	8.2 (3.8)	7.6 (2.9)	21.5 (10.1)	8.1 (4.0)
IA→SC transitions%	52.5 (16.6)	66.3 (21.8)	53.3 (17.1)	59.1 (18.7)
SA→SC transitions%	20.1 (13.5)	18.5 (9.3)	22.1 (12.6)	21.9 (10.0)
SC→IA transitions%	3.1 (2.1)	2.6 (1.6)	3.1 (1.6)	3.3 (2.5)
SC→SA transitions%	48.2 (16.5)	54.9 (21.9)	53.8 (15.9)	55.2 (22.1)
Conceptual Model Edit %	11.4 (7.2)	10.9 (7.7)	12.1 (8.0)	11.8 (6.5)
Computational Model Edit %	30.1 (21.0)	35.8 (22.5)	36.9 (18.9)	37.3 (23.3)
Model Testing effort %	14.8 (7.7)	18.8 (8.7)	11.5 (5.4)	16.5 (7.0)
Model Comparison effort %	16.5 (10.1)	21.5 (10.9)	18.3 (8.9)	19.7 (8.8)
Compare Model in Parts %	59.2 (15.6)	55.5 (12.0)	61.3 (12.5)	60.3 (15.7)
Coherence of Model Edits	4.1 (1.9)	4.3 (2.1)	6.3 (3.1)	4.4 (1.7)
Number of ModelEdit chunks	30.3 (13.6)	31.5 (15.9)	36.3 (10.3)	32.5 (16.3)
Size of ModelEdit chunks	4.8 (0.9)	4.5 (1.1)	4.3 (1.4)	4.9 (1.8)

more interested to see the differences generated by the reinforced scaffolding models when compared to the original models. For the **rollercoaster** unit, as we can see from Tables 5.13 and Table 5.7:

- The average percentage of taking IA actions for all the clusters were increased significantly (cluster 0: 1.4% $\rightarrow$ 15.2%, $p < 0.00001$; cluster 1: 1.7% $\rightarrow$ 10.1%, $p < 0.00001$; cluster 2: 7.6% $\rightarrow$ 17.0%, $p = 0.00002$; cluster 3: 3.7% $\rightarrow$ 11.8%, $p < 0.00001$). We did not see this change from the **rollercoaster** unit to the **macroscopic** fish tank unit. The more effort put in acquiring information from science resource increased the potential that would support later actions [7]. This could help students perform better at model building as well as gain a better understanding of the knowledge through the process of model building.

- Besides, there were more use of search actions, especially for cluster 2 (cluster 0: 2.3% $\rightarrow$ 8.2%, $p < 0.00001$; cluster 1: 1.9% $\rightarrow$ 7.6%, $p < 0.00001$; cluster 2: 8.2% $\rightarrow$ 21.5%, $p < 0.00001$; cluster 3: 1.4% $\rightarrow$ 8.1%, $p < 0.00001$). Because students in

cluster 2 had the highest use of search actions in the original data, the final measure of it in cluster 2 was also higher than in other clusters after reinforcement learning. Using more *search* actions when acquiring information can make students become more involved in the knowledge-seeking process.

- The average percentage of coherent transitions also increased for all the clusters. Therefore, more SC actions were supported by prior acquired information either from IA actions or from SA actions. Supported SC actions can effectively increase the chance of correctly building the domain model. However, as we have discussed before, the higher percentage of SA→SC transitions may indicate the use of trial-and-error approaches for building the domain model. In a previous study with CT-SiM [55], the chunk size of IA, SC, and SA actions was shown to be an important factor for success in model building. The use of smaller chunks implies that students decomposed their modeling tasks into smaller sub-tasks. When the chunk size is high, there will be high percentages of self-categorical transitions (e.g., IA→IA, SC→SC, and SA→SA). The reinforcement learning applied to this dataset rewarded the use of smaller chunks for better long-term consequences, and therefore reduce the percentage of self-categorical transitions. So the percentage of SA→SC transitions have been increased due to the decrements of SA chunk sizes.

- The average percentage of SA actions (i.e., the sum of **Model Test** and **Model Compare** efforts) decreased for all the clusters (cluster 0: 67.5% → 31.3%, $p < 0.00001$; cluster 1: 63.4% → 40.3%, $p = 0.003$; cluster 2: 40.3% → 29.8%, $p = 0.015$; cluster 3: 47.6% → 36.2%, $p = 0.022$). In this case, the effort put in taking SA actions were redistributed (reduced in general) to allow for more use of coherently related actions between information acquisition, solution construction, and solution assessment.

- The average percentage of **Compare Model in Parts** increased to the values that were close to those in the the original **macroscopic** fish tank unit (cluster 0: 34.2%

$\rightarrow$ 59.2%, p = 0.009; cluster 1: 24.8% $\rightarrow$ 55.5%, $p < 0.00001$; cluster 2: 29.0% $\rightarrow$ 61.3%, $p < 0.00001$; cluster 3: 23.6% $\rightarrow$ 60.3%, $p < 0.00001$). This shows that students did make great improvements in taking the divide-and-conquer strategy applied to solution assessment from the **rollercoaster** unit to the **macroscopic** fish tank unit.

- The average number of **Coherence of Model Edits** increased for every cluster (cluster 0: 1.1% $\rightarrow$ 4.1%, $p < 0.00001$; cluster 1: 1.3% $\rightarrow$ 4.3%, $p < 0.00001$; cluster 2: 4.2% $\rightarrow$ 6.3%, p = 0.015; cluster 3: 2.9% $\rightarrow$ 4.4%, p = 0.037). It has been shown in [55, 68] that maintaining the correspondence between the conceptual and computational models for each agent/environment behavior is very useful for improving students' model building performance. The reinforced scaffolding models have also captured these changes.

- The average number of ModelEdit chunks increased (cluster 0: 11.7% $\rightarrow$ 30.3%, $p < 0.00001$; cluster 1: 10.5% $\rightarrow$ 31.5%, $p < 0.00001$; cluster 2: 15.2% $\rightarrow$ 36.3%, $p < 0.00001$; cluster 3: 13.8% $\rightarrow$ 32.5%, $p < 0.00001$), while the average size of ModelEdit chunks decreased (cluster 0: 15.2% $\rightarrow$ 4.8%, $p < 0.00001$; cluster 1: 16.5% $\rightarrow$ 4.5%, $p < 0.00001$; cluster 2: 5.6% $\rightarrow$ 4.3%, p = 0.043; cluster 3: 10.9% $\rightarrow$ 4.9%, p = 0.00003) significantly after the reinforcement learning, showing that building smaller parts of the model separately can help improve the model building performance.

Overall, we have observed a lot of significant changes made by the reinforcement learning that can be considered as improvements in students' learning behaviors. Next, we run the reinforcement learning algorithm as illustrated in Section 4.3.3 to **extend** the **existing** action sequences for the **macroscopic** fish tank unit. The extended action sequences were used to learn updated HMMs as the reinforced scaffolding model. The reinforced scaffolding models for the **macroscopic** fish tank unit are shown in Figure 5.41 (cluster 0),

Figure 5.42 (cluster 1), Figure 5.43 (cluster 2), and Figure 5.44 (cluster 3).

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4	π_5
0.75	0.07	0.10	0.06	0.02

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4	h_5
h_1	0.25	0.33	0.24	0.15	0.03
h_2	0.25	0.37	0.11	0.16	0.11
h_3	0.09	0.16	0.43	0.23	0.09
h_4	0.17	0.15	0.24	0.29	0.15
h_5	0.04	0.05	0.07	0.03	0.81

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0.75	0.41	0.16	0.18	0.05
	$p(search)$	0.06	0.05	0.10	0.03	0.01
SC	$p(SC\_comp)$	0.01	0.30	0.41	0.18	0.11
	$p(SC\_conc)$	0.18	0.12	0.03	0.02	0
SA	$p(SA\_compare)$	0	0.11	0.14	0.40	0.62
	$p(SA\_run)$	0	0.01	0.16	0.19	0.21

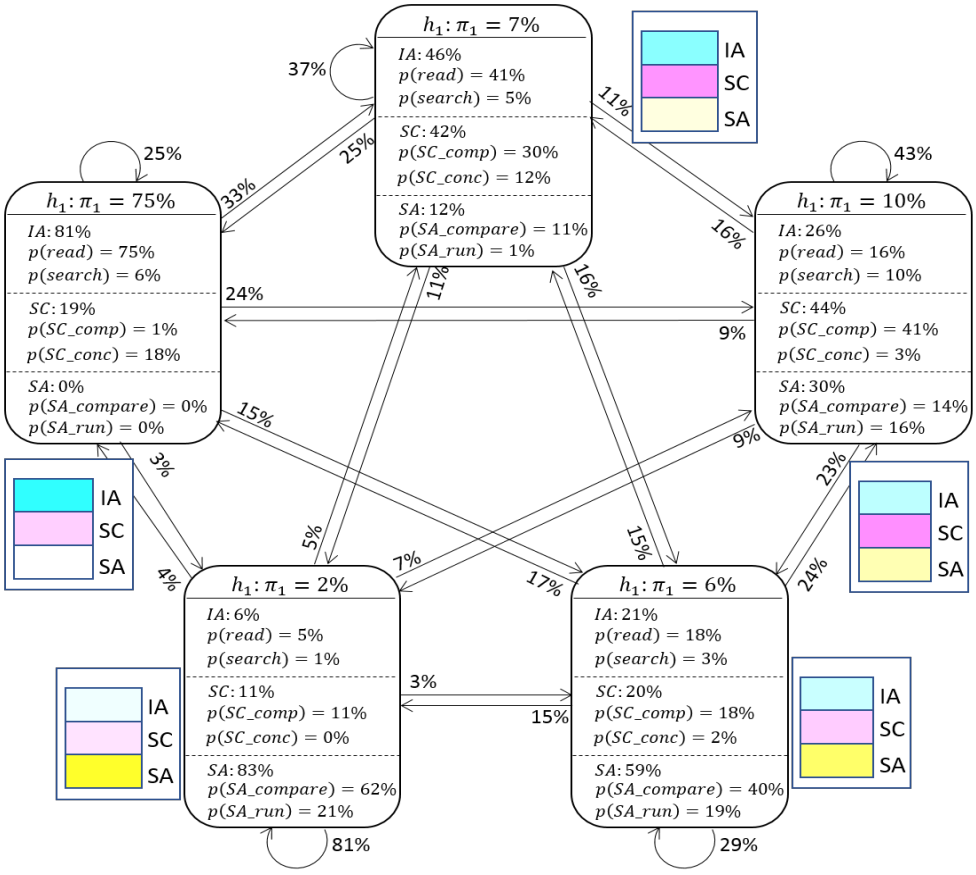


Figure 5.41: The reinforced scaffolding HMM of Cluster 0 for the **macroscopic** unit

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4	π_5
0.89	0.06	0.03	0.02	0

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4	h_5
h_1	0.44	0.16	0.16	0.20	0.04
h_2	0.11	0.62	0.13	0.08	0.06
h_3	0.16	0.13	0.41	0.19	0.11
h_4	0.09	0.09	0.16	0.52	0.14
h_5	0	0.02	0.01	0.06	0.91

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0.49	0.15	0.31	0.13	0.01
	$p(search)$	0.17	0.07	0.09	0.06	0
SC	$p(SC\_comp)$	0.03	0.27	0.22	0.29	0.03
	$p(SC\_conc)$	0.29	0.40	0.29	0	0
SA	$p(SA\_compare)$	0	0.03	0.02	0.23	0.44
	$p(SA\_run)$	0.01	0.08	0.06	0.29	0.48

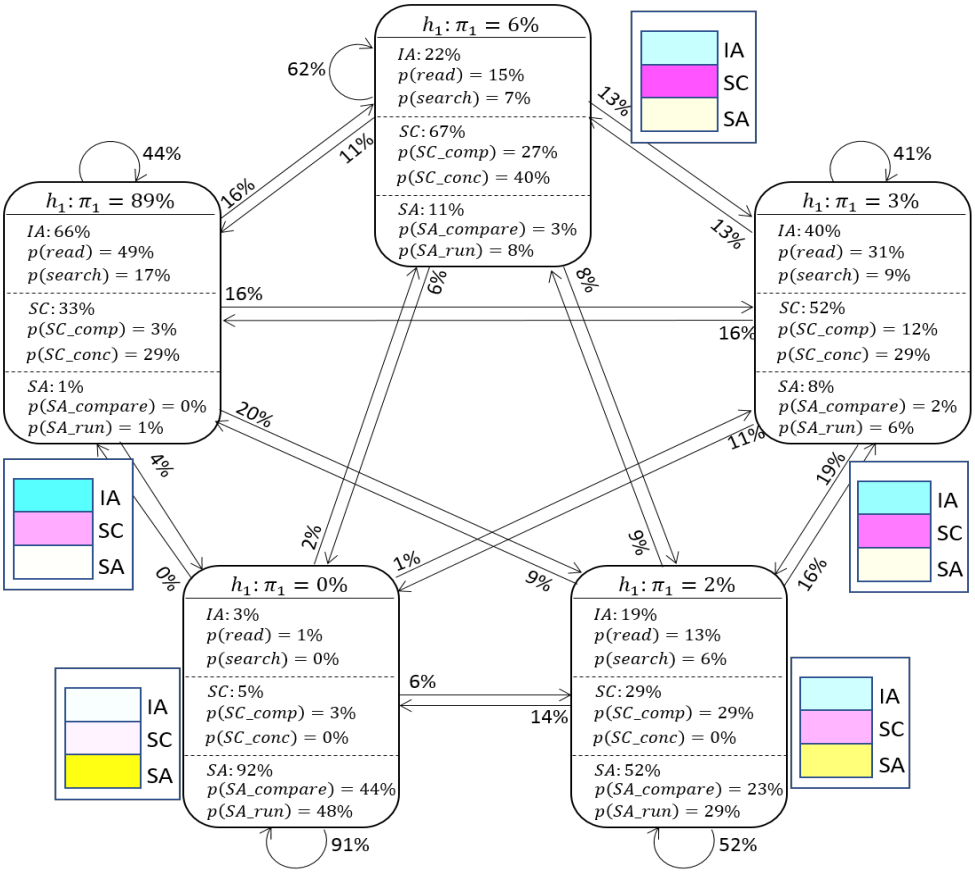


Figure 5.42: The reinforced scaffolding HMM of Cluster 1 for the **macroscopic** unit

In order to compare the reinforced scaffolding HMMs with the original HMMs for

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4	π_5
0.76	0.14	0.05	0.05	0

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4	h_5
h_1	0.31	0.29	0.28	0.09	0.03
h_2	0.19	0.39	0.11	0.23	0.08
h_3	0.21	0.15	0.44	0.09	0.11
h_4	0.05	0.11	0.11	0.54	0.19
h_5	0.02	0.04	0.06	0.03	0.85

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0.41	0.19	0.31	0.16	0.08
	$p(search)$	0.11	0.10	0.09	0.06	0.01
SC	$p(SC\_comp)$	0.13	0.21	0.29	0.15	0.03
	$p(SC\_conc)$	0.29	0.19	0.03	0.14	0.02
SA	$p(SA\_compare)$	0	0.21	0.19	0.39	0.67
	$p(SA\_run)$	0.06	0.10	0.09	0.10	0.19

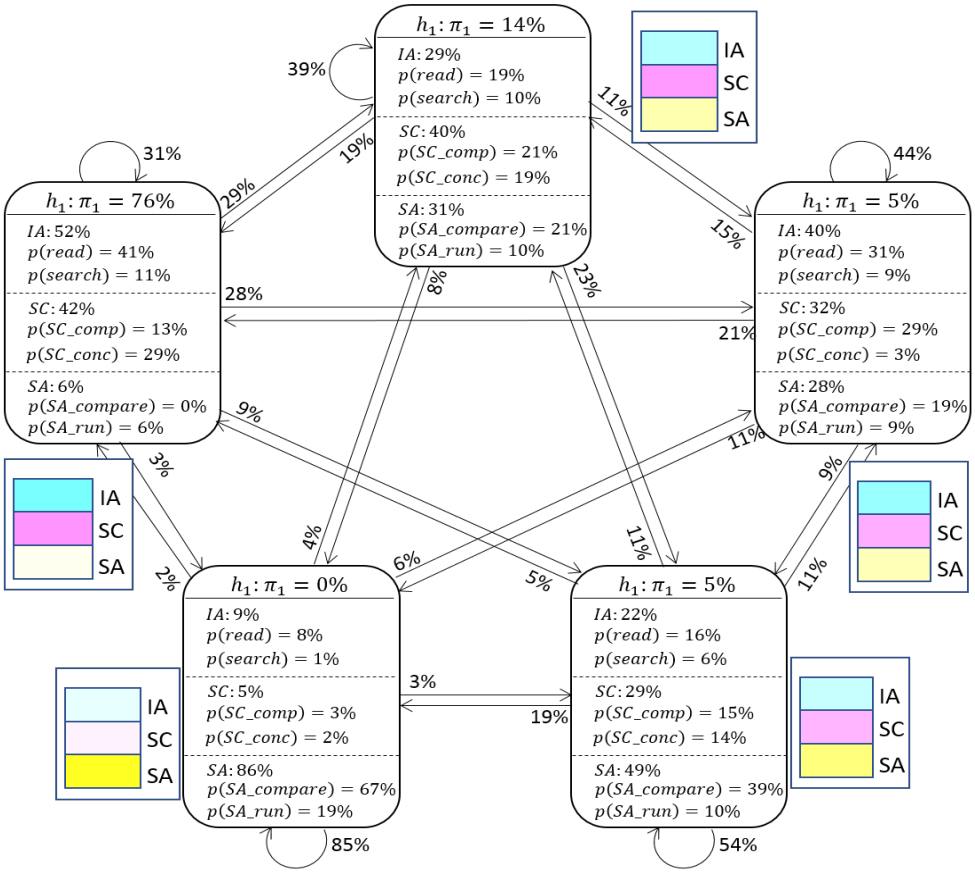


Figure 5.43: The reinforced scaffolding HMM of Cluster 2 for the **macroscopic** unit

$\boldsymbol{\pi}$:

π_1	π_2	π_3	π_4	π_5
0.80	0.10	0.06	0.02	0.02

$\boldsymbol{A}$:

	h_1	h_2	h_3	h_4	h_5
h_1	0.27	0.25	0.21	0.18	0.09
h_2	0.19	0.37	0.19	0.16	0.11
h_3	0.21	0.12	0.32	0.22	0.13
h_4	0.16	0.21	0.09	0.41	0.13
h_5	0.06	0.07	0.04	0.04	0.79

$\boldsymbol{B}$:

		h_1	h_2	h_3	h_4	h_5
IA	$p(read)$	0.56	0.29	0.22	0.17	0.08
	$p(search)$	0.07	0.05	0.07	0.02	0
SC	$p(SC\_comp)$	0.02	0.20	0.28	0.29	0.02
	$p(SC\_conc)$	0.25	0.32	0.03	0	0
SA	$p(SA\_compare)$	0	0.04	0.25	0.52	0.73
	$p(SA\_run)$	0.10	0.10	0.15	0.02	0.17

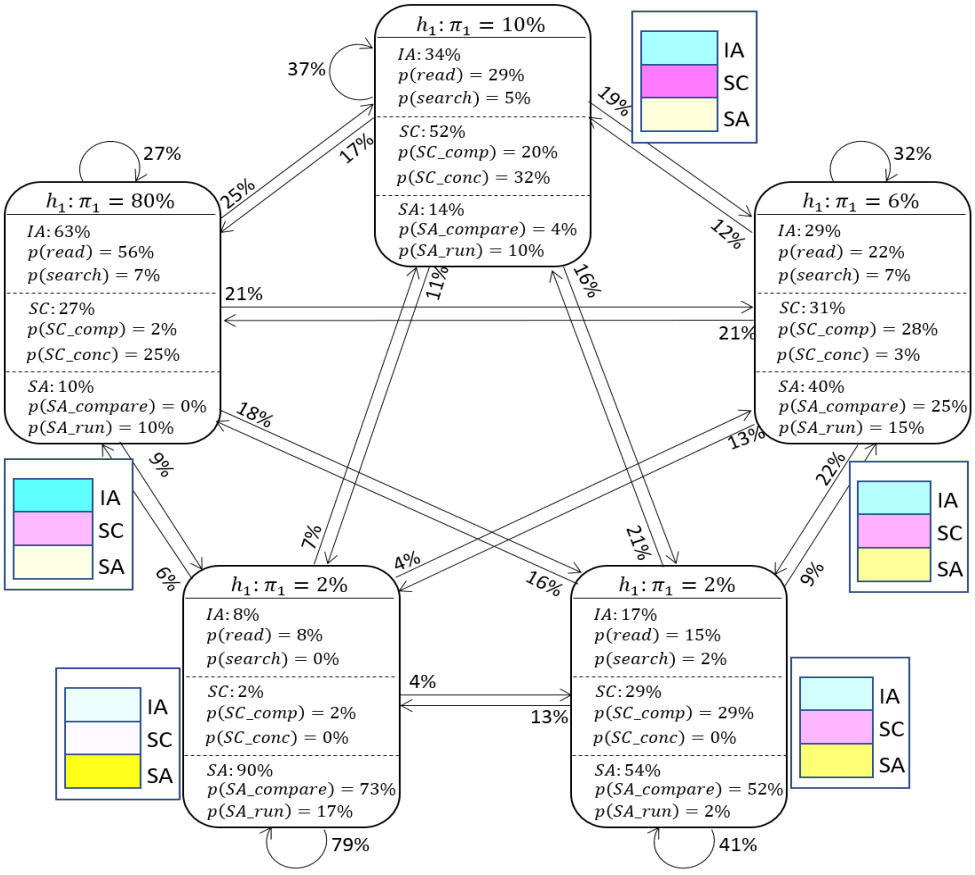


Figure 5.44: The reinforced scaffolding HMM of Cluster 3 for the **macroscopic** unit

Table 5.14: Comparison of the four Clusters post-reinforcement scaffolding learning for the **macroscopic** fish tank unit in CTSiM. Results are presented as mean (standard deviation).

	Cluster 0 (n = 19)	Cluster 1 (n = 19)	Cluster 2 (n = 20)	Cluster 3 (n = 40)
Overall IA effort %	31.1 (10.3)	28.6 (12.5)	25.2 (8.6)	26.8 (11.5)
Search Rate %	10.6 (5.1)	22.3 (8.7)	26.7 (12.5)	14.1 (6.6)
IA→SC transitions%	41.1 (17.6)	29.8 (9.5)	32.3 (8.3)	39.1 (12.3)
SA→SC transitions%	10.6 (4.7)	11.2 (5.1)	12.1 (5.1)	9.8 (4.8)
SC→IA transitions%	5.5 (3.3)	4.3 (2.9)	11.3 (7.3)	9.6 (6.1)
SC→SA transitions%	37.2 (12.0)	33.6 (12.6)	26.8 (8.5)	27.8 (9.4)
Conceptual Model Edit %	10.5 (3.5)	9.9 (2.9)	11.3 (4.2)	10.6 (2.6)
Computational Model Edit %	21.1 (10.1)	26.8 (12.5)	37.1 (16.0)	36.5 (18.9)
Model Testing effort %	12.5 (4.3)	18.6 (4.6)	8.3 (3.1)	9.5 (2.4)
Model Compare effort %	21.6 (13.5)	12.3 (8.8)	21.5 (11.4)	19.7 (13.0)
Compare Model in Parts %	81.8 (10.5)	78.2 (9.9)	80.5 (11.5)	77.5 (12.5)
Coherence of Model Edits	10.4 (5.3)	9.9 (5.5)	13.8 (6.7)	11.6 (5.9)
Number of ModelEdit chunks	59.7 (23.7)	60.2 (26.7)	63.1 (27.0)	64.5 (31.6)
Size of ModelEdit chunks	4.9 (1.5)	5.2 (1.9)	4.8 (1.1)	4.6 (2.2)

the **macroscopic** fish tank unit, we generated the results for the same measures from the extended action sequences that were used to learn reinforced HMMs and compared them with the results from original data shown in Table 5.9. The newly generated results of the measures for the **macroscopic** fish tank unit are shown in Table 5.14. The learning performance measures ($\overline{S_g}$ and $\overline{S_m}$) were excluded here because there is no accurate way to compute the real learning performance out of a virtual agent under reinforcement learning.

For the **macroscopic** fish tank unit, we compared the results from Table 5.14 and Table 5.9, and present the analyses as following:

- The average percentages of IA actions taken by students from all the clusters increased significantly (cluster 0: 1.6% $\rightarrow$ 31.1%, $p < 0.00001$; cluster 1: 2.3% $\rightarrow$ 28.6%, $p < 0.00001$; cluster 2: 6.7% $\rightarrow$ 25.2%, $p < 0.00001$; cluster 3: 4.8% $\rightarrow$ 26.8%, $p < 0.00001$). The amount of increment was higher than those in the **roller-coaster** unit, showing that more unused potential were performed to build better domain models in the **macroscopic** fish tank unit.

- Within the IA actions, there were more use of search actions for all the clusters (cluster 0: 5.3% $\rightarrow$ 10.6%, $p = 0.007$; cluster 1: 10.6% $\rightarrow$ 22.3%, $p = 0.003$; cluster 2: 13.2% $\rightarrow$ 26.7%, $p = 0.009$; cluster 3: 5.2% $\rightarrow$ 14.1%, $p < 0.00001$). These increments were also higher than the **rollercoaster** unit. One possible reason is that, the domain model to be built in the **macroscopic** fish tank unit is more complex than the **rollercoaster** unit, and the trial-and-error approach became even less reliable. In order to build correct domain models, students needed to search and acquire more information to support their solution construction actions.

- Similarly to our analyses for the **rollercoaster** unit, the percentage of taking coherent transitions increased for all the clusters. However, there were some differences between clusters. For example, cluster 2 and 3 had more of the SC$\rightarrow$IA transitions (cluster 0: 0.8% $\rightarrow$ 5.5%, $p < 0.00001$; cluster 1: 1.1% $\rightarrow$ 4.3%, $p = 0.0002$; cluster 2: 1.9% $\rightarrow$ 11.3%, $p < 0.00001$; cluster 3: 0.7% $\rightarrow$ 9.6%, $p < 0.00001$), while cluster 0 and 1 took more SC$\rightarrow$SA transitions (cluster 0: 11.7% $\rightarrow$ 37.2%, $p < 0.00001$; cluster 1: 12.6% $\rightarrow$ 33.6%, $p < 0.00001$; cluster 2: 9.3% $\rightarrow$ 26.8%, $p < 0.00001$; cluster 3: 9.8% $\rightarrow$ 27.8%, $p < 0.00001$). Depending on the different paths toward success, the learning strategy in cluster 2 and 3 encourages more on going back to read after building some parts of the domain model than cluster 0 and 1.

- The overall percentage of SA actions (i.e., the sum of **Model Test** and **Model Compare** efforts) also decreased for all the clusters (cluster 0: 69.6% $\rightarrow$ 34.1%, $p = 0.007$; cluster 1: 63.5% $\rightarrow$ 30.9%, $p = 0.0009$; cluster 2: 53.8% $\rightarrow$ 29.8%, $p = 0.009$; cluster 3: 49.1% $\rightarrow$ 29.2%, $p = 0.008$). And the percentage of **Compare Model in Parts** increased to even higher values (cluster 0: 65.8% $\rightarrow$ 81.8%, $p = 0.011$; cluster 1: 65.2% $\rightarrow$ 78.2%, $p = 0.023$; cluster 2: 65.6% $\rightarrow$ 80.5% $p = 0.031$; cluster 3: 60.9% $\rightarrow$ 77.5%, $p = 0.048$). Like in the **rollercoaster** unit, the effort put in taking SA actions were redistributed to allow the more use of coherently related actions between

information acquisition, solution construction, and solution assessment.

- The average number of **Coherence of Model Edits** increased for every cluster (cluster 0: 3.6% $\rightarrow$ 10.4%, $p < 0.00001$; cluster 1: 3.1% $\rightarrow$ 9.9%, $p < 0.00001$; cluster 2: 6.6% $\rightarrow$ 13.8%, $p < 0.00001$; cluster 3: 4.8% $\rightarrow$ 11.6%, $p = 0.00002$). The reason is the same as shown in [55, 68] that maintains the correspondence between the conceptual and computational models for each agent/environment behavior is very useful for improving students' model building performance.

- The average number of ModelEdit chunks increased (cluster 0: 32.7% $\rightarrow$ 59.7%, $p = 0.0009$; cluster 1: 30.8% $\rightarrow$ 60.2%, $p < 0.00001$; cluster 2: 45.1% $\rightarrow$ 63.1%, $p = 0.001$; cluster 3: 39.5% $\rightarrow$ 64.5%, $p = 0.0005$), while the average size of ModelEdit chunks decreased (cluster 0: 9.6% $\rightarrow$ 4.9%, $p < 0.00001$; cluster 1: 13.1% $\rightarrow$ 5.2%, $p < 0.00001$; cluster 2: 7.6% $\rightarrow$ 4.8%, $p = 0.019$; cluster 3: 7.3% $\rightarrow$ 4.6%, $p = 0.006$) significantly after the reinforcement learning, showing that building smaller parts of the model separately can also help improve the model building performance in the **macroscopic** fish tank unit.

Overall, we can see some significant changes in the learning behaviors after reinforcement learning according to Tables 5.13 and 5.14. For both units, the percentage of taking IA actions have increased significantly to encourage acquiring more information from the science resources so that the later SC action can be supported. The reinforced models also encouraged a lot more use of coherent transitions and divide-and-conquer strategies such as **Compare Model in Parts** actions to build the model in small chunks.

Some changes of these measures pre- and post-reinforcement learning in the **rollercoaster** unit have been reflected by the original evolution from the **rollercoaster** unit to the **macroscopic** fish tank unit, showing that the students can improve by themselves over time. But the reinforcement learning shows that students can improve at much faster rates when supported by adaptive scaffolding.

5.2.4 **Example scaffolds**

Using the same methodology illustrated in Section 5.1.4, we can identify students that are under-performing and provide adaptive scaffolds by comparing measures of their action sequences against the corresponding measures derived from the reinforced scaffolding models. We list some examples of the scaffolds that can be derived as follows:

- If a student who is learning in the **rollercoaster** unit, has taken a significant amount of SA actions whereas his/her effectiveness percentage of model building actions remain in low level. The students can be informed to use **Compare Model in Parts** function if he/she hasn't used it a lot. Otherwise, scaffolds can be given to encourage decomposition of the modeling task into smaller sub-tasks. Since **Compare Model in Parts** plays an important role in the divide-and-conquer strategy, the percentage measure of it should always be monitored when students are learning in CTSiM.

- According to the reinforced scaffolding models for **rollercoaster** and **macroscopic** fish tank units, scaffolds can be provided differently. For example, a student who is classified into cluster 0 can be encouraged to go back to read more frequently when he/she is learning in the **macroscopic** fish tank unit (IA effort in the reinforced model is 31.1%) than that in the **rollercoaster** unit (IA effort in the reinforced model is 15.2%). We can use specific action patterns to show the adaptive model building strategies that can be suggested to students in different units. For example, there is a frequent pattern "IA$\rightarrow$ SC_conc $\rightarrow$ SC_comp $\rightarrow$ SA" for the **rollercoaster** unit, which can be applied to build the domain model. This pattern refers to a strategy that uses information acquired by reading to support subsequent conceptual and computational ModelEdit actions that are coherently related. The solution assessment actions are taken at the end to verify the ModelEdit actions. But when it comes to the **macroscopic** fish tank unit, the more frequent pattern is changed to "IA $\rightarrow$ SC_conc $\rightarrow$ SC_comp $\rightarrow$ IA $\rightarrow$ SA $\rightarrow$ IA", where more IA actions are inserted into the same

strategy to help them stick to the parts to be built and reduce the chance of incorrect SC actions by acquiring more information from the science resources. In complex modeling tasks, it is hard to detect and correct mistakes compared to simpler models. So, it is important to apply the best strategy for building every part of the domain model.

- Within each unit, different scaffolds can be provided to a student if he/she is classified into different clusters. For example in the **macroscopic** fish tank unit, a student with low usage percentage of coherent transitions would receive suggestion on applying strategy of "IA $\rightarrow$ SC_conc $\rightarrow$ SC_comp $\rightarrow$ SA" if he/she is classified into cluster 2 or 3, whereas the suggestion would be to take more IA actions before taking solution assessment actions (i.e., "IA $\rightarrow$ SC_conc $\rightarrow$ SC_comp $\rightarrow$ IA $\rightarrow$ SA $\rightarrow$ IA") if he/she is classified into cluster 0 and 1. According to the reinforced scaffolding models, the percentage of going back to read after taking SA actions, i.e., the percentage of SC $\rightarrow$ IA transitions for cluster 0 (5.5%) and cluster 1 (4.3%), are significantly lower than that in cluster 2 (11.3%) and 3 (10.6%). The reinforced scaffolding models here can be interpreted as that students who are classified into cluster 2 and 3 have more successful model building actions by keeping high IA-involved coherent transitions.

- The scaffolds can be provided according to students' real-time performance. For example, if an under-performing student is building the model with SC actions having the average effectiveness percentage of above 0.5, the suggestion will be focusing on the higher level decomposition of the modeling tasks. But if the student is making mistakes repeatedly, a more detailed strategy with more coherent transitions between IA, SC, and SA actions can be suggested to make sure he/she understands what went wrong and why.

5.3 Summary

In this chapter, we have presented the experiment results using data collected from 98 6^{th} grade students who worked in the Betty's Brain system (see Section 3.1). We then introduced a second OELE, i.e., CTSiM, and presented experimental results using data collected from students who worked for two units in CTSiM (i.e., the **rollercoaster** and the **macroscopic** fish tank units).

Using each data set, we generated both the reinforced classification HMMs and the reinforced scaffolding HMMs. For the reinforced classification model, we run Leave-one-out cross validation (LOOCV) on the original data set as well as the data set used to train the reinforced classification models. The comparison of LOOCV accuracy for datasets pre- and post-reinforcement learning have shown that the reinforced classification models can perform better in classifying students into correct clusters that were derived by HMM clustering.

Then, we compared the reinforced scaffolding HMMs with the HMMs generated from the original data set. We used coherent measures adopted from previous work to illustrate the change of the learning behaviors pre- and post-reinforcement learning. These results have shown the strength of the reinforcement learning that the reinforced-scaffolding model can capture the evolution of learning behaviors that can lead to better learning outcomes, and thus provide the basis for adaptive scaffolding.

For the Betty's Brain system, the reinforcement learning mostly discovers some common action patterns that can help students' improve their learning across different clusters. On the other hand, in the CTSiM OELE, the reinforcement learning is more likely to find different action patterns that may help students from different cluster to make progress on building the domain models.

We have also illustrated the process of monitoring and providing scaffolds to students who might be under-performing. Some example scaffolds for different OELEs (i.e., Betty's Brain versus CTSiM) are provided to show that the reinforced classification and the rein-

forced scaffolding model can be used together to form the basis for generating adaptive scaffolds.

Chapter 6

Conclusions and Future Work

In this work, we have presented a data-driven learner modeling approach to model students' learning behaviors in Open-Ended Learning Environments (OELEs). The approach combines the use multiple machine learning techniques, such as Hidden Markov Model (HMMs), unsupervised learning, reinforcement learning, and Monte Carlo tree search (MCTS), within a reinforcement learning framework, to generate complete and accurate learner models from students' data collected from OELEs. This learner modeling approach generates reinforced learner models as the basis for providing adaptive scaffolds, which can help students building better domain models and become better learners in OELEs. We summarize the major accomplishments of this work in Sections 6.1 and 6.2, and present potential future work in Section 6.3.

6.1 Accomplishment I - The Learner Modeling Approach

Started by learning HMMs from the original data set as the initial learner model, we applied reinforcement learning and MCTS to generate/extend action sequences and combined them with the original students' action sequences to learn updated HMMs, which we believe are more complete and accurate. When generating HMMs for the original data set, we also applied an HMM clustering algorithm to derive clusters of students with similar behaviors, where the reinforcement learning was performed on each of the clusters.

Since we wanted the learner modeling approach to be generally applicable to different OELEs, we derived the action-view representation which logged students' actions as well as the corresponding context associated with the actions when they worked in OELEs. The action-view representation can be applied to different learning environments, and therefore, provided a solution to the data heterogeneity problem. We also used this representation to

derive coherence relations between actions, which were later applied to help to perform simulations in MCTS.

One of the challenges for learner modeling on data collected from classroom studies using OELEs, is that we may suffer from the data impoverishment problem (e.g., a Betty's Brain study with a class of 98 students), which can cause inaccurate models to be built. This motivated us to develop the reinforcement learning algorithm combined with MCTS to grow the data volume by generating/extending students' action sequences. Basically, the reinforcement learning was applied to simulate a virtual student interacting with the OELE, where those interactions were logged as actions for generating/extending action sequences.

Within the reinforcement learning framework, we applied the MCTS as the interpreter to determine which actions shall be taken by the virtual student to interact with the system. The MCTS can produce rewards for all of the available actions by taking into consideration long-term consequences according to specific modeling purposes. In this work, we configured the selection policy in MCTS differently to derive (1) the reinforced classification models by generating additional action sequences, to increase the accuracy of classifying students into correct groups; and (2) the reinforced scaffolding models by extending existing action sequences, to capture behavioral patterns that can lead to better performance in OELEs. Overall, we have successfully implemented the proposed learner modeling approach.

6.2 **Accomplishment II - Verification of the Learner Modeling Approach**

In order to verify our learner modeling approach, we performed experiments on two different OELEs, i.e., Betty's Brain and CTSiM, which we've presented in detail about their interfaces as well as available actions that students can take. For the experiments, we generated the HMMs from the original data set and used them to derive the reinforced models for each of the two experiments. In addition, we adopted various coherent measures from earlier work, to analyze and compare students' learning behaviors and performance.

For the original models, we generated results of the measures and performed pairwise MannWhitney U-Test for each pair of clusters. We used these results to analyze and discuss students' learning behaviors with respect to their corresponding learning performance. Then, for the reinforced models:

- We performed leave-one-out cross validation (LOOCV) on the original data set and the reinforced data set to show the improvements in classification accuracy delivered by the **reinforced classification** model. For both experiments, we have observed significant accuracy improvements pre- and post-reinforcement learning.

- We compared the **reinforced scaffolding** models with the original HMMs and empirically analyzed and interpreted the changes of the measures to discover the behavioral evolution that can lead to better performance. The experiments showed interesting results (e.g., the redistribution of effort put in information acquisition, solution construction, and solution assessment, the ratio change of using coherent actions, etc.), which are then used as the basis for providing example scaffolds.

For providing adaptive scaffolds, we used the reinforced classification model to classify a student who has been detected as under-performing into a cluster that is derived from the HMM clustering. The reinforced scaffolding model corresponding to this cluster was then used to determine what the deficiencies were about the student's current activities, based on which, scaffolds were provided accordingly.

Overall, we have developed a data-driven learner modeling approach which combines multiple techniques, to solve the challenges we met for accurate learner modeling. Experiments with the two different OELEs have produced promising results, showing that our approach has the potential to better understand students' learning behaviors and to support their learning in OELEs.

6.3 Future Work

Taking this work as a starting point, there are several research directions that can be considered as future works. For each of these future work, sufficient effort on research and experiments are needed to enhance and promote the strength of our learner modeling approach. We summarize some of the research directions below.

Designing sophisticated system for providing adaptive scaffolding. In this work, we presented empirical analyses and introduced some example scaffolds based on the experimental results. It is important to design a sophisticated system that can appropriately provide adaptive scaffolding to be used in the future study with OELEs. This process should keep the teachers involved by providing them the information about the learner models, as well as the outcome of reinforced classification (e.g., which group is each individual classified into) and reinforced scaffolding (e.g., what are the potential scaffolds in different circumstances).

Applying the learner modeling approach to multi-stream data. For example, the data we get from Meta Tutor consists not only the event log of learners activities and interactions with the system but also their emotion data collected when they are learning. The multi-stream HMMs (MS-HMM) can be applied to model learners' data with multiple observation sequences. However, there should be a different methodology for reinforcement learning to take into consideration the relation between actions and emotions.

Applying Alternative representation of learner models. In this work, we used the Hidden Markov Models to capture students' learning behaviors as well as their cognitive and metacognitive processes when they study in OELEs. It is interesting to see if other data-driven models can fit into this reinforcement learning schema. For example, the deep Convolutional Neural Network (CNN) can be used as the model for performing the classification task as well as predicting performance outcome. Compared to the HMM, the deep CNN can represent students' learning behaviors and cognitive and metacognitive process at different levels of abstraction by using multiple hidden layers. However, in order to use

the CNN appropriately, one should solve the:

1. Input problem. A neural network requires a set of inputs instead of the action sequences collected from OELEs that were used to learn HMMs.

2. Interpretation problem about neurons. A neural network is typically viewed as a black box, where the neurons are hard to interpret. In HMM, it is easier to interpret the hidden states and transitions by analyzing the probability matrices.

Applying the learner modeling approach to other learning environments. Because of the action-view representation, we solved the data heterogeneity problem across the two different OELEs, i.e., Betty's Brain and CTSiM. It will be interesting to see if it is applicable to other learning environments by building appropriate task models. The learner modeling approach should also have adaptive design of reinforcement learning framework to meet different specification and requirements. For example, the interaction between the virtual agent, the environment, and the interpreter for reinforcement learning might be different (e.g., learning environments with multiple agents and environments and different interpretation methodologies).

BIBLIOGRAPHY

[1] Satabdi Basu, Amanda Dickes, John S Kinnebrew, Pratim Sengupta, and Gautam Biswas. Ctsim: A computational thinking environment for learning science through simulation and modeling. In *CSEDU*, pages 369–378, 2013.

[2] James René Segedy, John S Kinnebrew, and Gautam Biswas. Modeling learner's cognitive and metacognitive strategies in an open-ended learning environment. In *AAAI Fall Symposium: Advances in Cognitive Systems*, 2011.

[3] Geraldine Clarebout, Jan Elen, W Lewis Johnson, and Erin Shaw. Animated pedagogical agents: An opportunity to be grasped? *Journal of Educational multimedia and hypermedia*, 11(3):267–286, 2002.

[4] Susan Land. Cognitive requirements for learning with open-ended learning environments. *Educational Technology Research and Development*, 48(3):61–78, 2000.

[5] Satabdi Basu, Gautam Biswas, and John S Kinnebrew. Using multiple representations to simultaneously learn computational thinking and middle school science. In *Thirtieth AAAI Conference on Artificial Intelligence*, pages 3705–3711, Phoenix, AZ, 2016.

[6] Gautam Biswas, James R Segedy, and Kritya Bunchongchit. From design to implementation to practice a learning by teaching system: Betty's brain. *International Journal of Artificial Intelligence in Education*, 26(1):350–364, 2016.

[7] James R Segedy, John S Kinnebrew, and Gautam Biswas. Using coherence analysis to characterize self-regulated learning behaviours in open-ended learning environments. *test*, 2(1):13–48, 2015.

[8] Satabdi Basu, Gautam Biswas, Pratim Sengupta, Amanda Dickes, John S Kinnebrew, and Douglas Clark. Identifying middle school students?challenges in computational thinking-based science learning. *Research and Practice in Technology Enhanced Learning*, 11(1):1–35, 2016.

[9] John Kinnebrew, James Segedy, and Gautam Biswas. Integrating model-driven and data-driven techniques for analyzing learning behaviors in open-ended learning environments. *IEEE Trans of Learning Technologies*, to appear, 2016.

[10] John S Kinnebrew, James R Segedy, and Gautam Biswas. Analyzing the temporal evolution of students' behaviors in open-ended learning environments. *Metacognition and learning*, 9(2):187–215, 2014.

[11] Janet Metcalfe and Bridgid Finn. Metacognition and control of study choice in children. *Metacognition and learning*, 8(1):19–46, 2013.

[12] Jere E Brophy. *Motivating students to learn*. Routledge, 2013.

[13] Philip H Winne. Improving measurements of self-regulated learning. *Educational Psychologist*, 45(4):267–276, 2010.

[14] Daniel L Schwartz and Taylor Martin. Inventing to prepare for future learning: The hidden efficiency of encouraging original student production in statistics instruction. *Cognition and Instruction*, 22(2):129–184, 2004.

[15] Jason Tan and Gautam Biswas. The role of feedback in preparation for future learning: A case study in learning by teaching environments. In *International Conference on Intelligent Tutoring Systems*, pages 370–381. Springer, 2006.

[16] David Silver et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

[17] Gautam Biswas, Hogyeong Jeong, John S Kinnebrew, Brian Sulcer, and ROD ROSCOE. Measuring self-regulated learning skills through social interactions in a teachable agent environment. *Research and Practice in Technology Enhanced Learning*, 5(02):123–152, 2010.

[18] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

[19] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[20] Hogyeong Jeong, Amit Gupta, Rod Roscoe, John Wagster, Gautam Biswas, and Daniel Schwartz. Using hidden markov models to characterize student behaviors in learning-by-teaching environments. In *International Conference on Intelligent Tutoring Systems*, pages 614–625. Springer, 2008.

[21] R Luckin et al. Modeling learning patterns of students with a tutoring system using hidden markov models. *Artificial intelligence in education: Building technology rich learning contexts that work*, 158:238, 2007.

[22] Chris Piech, Mehran Sahami, Daphne Koller, Steve Cooper, and Paulo Blikstein. Modeling how students learn to program. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 153–160. ACM, 2012.

[23] Girish Balakrishnan and Derrick Coetzee. Predicting student retention in massive open online courses using hidden markov models. *Electrical Engineering and Computer Sciences University of California at Berkeley*, 2013.

[24] Masun Homsi, Rania Lutfi, Rosa M Carro, and Barakat Ghias. A hidden markov model approach to predict students' actions in an adaptive and intelligent web-based

educational system. In *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, pages 1–6. IEEE, 2008.

[25] Luis Javier Rodríguez and Inés Torres. Comparative study of the baum-welch and viterbi training algorithms applied to read and spontaneous speech recognition. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 847–857. Springer, 2003.

[26] Cen Li and Gautam Biswas. A bayesian approach to temporal data clustering using hidden markov models. In *ICML*, pages 543–550, 2000.

[27] Konstantina Chrysafiadi and Maria Virvou. Student modeling approaches: A literature review for the last decade. *Expert Systems with Applications*, 40(11):4715–4729, 2013.

[28] Elaine Rich. User modeling via stereotypes. *Cognitive science*, 3(4):329–354, 1979.

[29] Kurt VanLehn. Student modeling. *Foundations of intelligent tutoring systems*, 55:78, 1988.

[30] Mark Elsom-Cook. Student modelling in intelligent tutoring systems. *Artificial Intelligence Review*, 7(3-4):227–240, 1993.

[31] James L Stansfield, Brian P Carr, and Ira P Goldstein. Wumpus advisor 1: A first implementation program that tutors logical and probabilistic reasoning skills. 1976.

[32] Peter Brusilovsky and Eva Millán. User models for adaptive hypermedia and adaptive educational systems. In *The adaptive web*, pages 3–53. Springer-Verlag, 2007.

[33] ASHRAF A Kassim, SABBIR AHMED Kazi, and SURENDRA Ranganath. A web-based intelligent learning environment for digital systems. *International Journal of Engineering Education*, 20(1):13–23, 2004.

[34] Cristina Carmona and Ricardo Conejo. A learner model in a distributed environment. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 353–359. Springer, 2004.

[35] Lisa N Michaud and Kathleen F McCoy. Empirical derivation of a sequence of user stereotypes for language learning. *User Modeling and User-Adapted Interaction*, 14(4):317–350, 2004.

[36] Stellan Ohlsson. Constraint-based student modeling. In *Student modelling: the key to individualized knowledge-based instruction*, pages 167–189. Springer, 1994.

[37] Kyparisia A Papanikolaou, Maria Grigoriadou, Harry Kornilakis, and George D Magoulas. Personalizing the interaction in a web-based educational hypermedia system: the case of inspire. *User modeling and user-adapted interaction*, 13(3):213–267, 2003.

[38] Raymund Sison and Masamichi Shimura. Student modeling and machine learning. *International Journal of Artificial Intelligence in Education (IJAIED)*, 9:128–158, 1998.

[39] Victoria Tsiriga and Maria Virvou. Modelling the student to individualise tutoring in a web-based icall. *International Journal of Continuing Engineering Education and Life Long Learning*, 13(3-4):350–365, 2003.

[40] Ryan SJd Baker. Modeling and understanding students' off-task behavior in intelligent tutoring systems. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1059–1068. ACM, 2007.

[41] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[42] Hamed Shakouri and Mohammad B Menhaj. A systematic fuzzy decision-making process to choose the best model among a set of competing models. *IEEE Transac-*

tions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 38(5):1118–1128, 2008.

[43] Konstantina Chrysafiadi and Maria Virvou. Evaluating the integration of fuzzy logic into the student model of a web-based learning environment. *Expert Systems with Applications*, 39(18):13127–13134, 2012.

[44] Cristina Conati, Abigail Gertner, and Kurt Vanlehn. Using bayesian networks to manage uncertainty in student modeling. *User modeling and user-adapted interaction*, 12(4):371–417, 2002.

[45] D Abderrahim et al. Using bayesian networks for student modeling. In *Proceedings of the 6, h IEEE International Conference on Advanced Learning Technologies*, pages 1002–1007, 2006.

[46] Eva Millán, Tomasz Loboda, and Jose Luis Pérez-de-la Cruz. Bayesian networks for student model engineering. *Computers & Education*, 55(4):1663–1683, 2010.

[47] Krittaya Leelawong and Gautam Biswas. Designing learning by teaching agents: The betty's brain system. *IJ Artificial Intelligence in Education*, 18(3):181–208, 2008.

[48] Michael J Hannafin. Open-ended learning environments: Foundations, assumptions, and implications for automated design. In *Automating instructional design: Computer-based development and delivery tools*, pages 101–129. Springer, 1995.

[49] John D Bransford, A Brown, and R Cocking. How people learn: Mind, brain, experience, and school. *Washington, DC: National Research Council*, 1999.

[50] Gautam Biswas, Hogyeong Jeong, John S Kinnebrew, Brian Sulcer, and ROD ROSCOE. Measuring self-regulated learning skills through social interactions in a teachable agent environment. *Research and Practice in Technology Enhanced Learning*, 5(02):123–152, 2010.

[51] Ningyu Zhang, Gautam Biswas, and Yi Dong. *Characterizing Students' Learning Behaviors Using Unsupervised Learning Methods*, pages 430–441. Springer International Publishing, Cham, 2017.

[52] James R Segedy. *Adaptive scaffolds in open-ended computer-based learning environments*. Vanderbilt University, 2014.

[53] J. S. Kinnebrew, J. R. Segedy, and G. Biswas. Integrating model-driven and data-driven techniques for analyzing learning behaviors in open-ended learning environments. *IEEE Transactions on Learning Technologies*, 10(2):140–153, April 2017.

[54] Ningyu Zhang, Gautam Biswas, and Yi Dong. Characterizing students' learning behaviors using unsupervised learning methods. In *International Conference on Artificial Intelligence in Education*, pages 430–441. Springer, 2017.

[55] Satabdi Basu, Gautam Biswas, and John S Kinnebrew. Using multiple representations to simultaneously learn computational thinking and middle school science. In *AAAI*, pages 3705–3711, 2016.

[56] James R Segedy, John S Kinnebrew, and Gautam Biswas. Coherence over time: understanding day-to-day changes in students open-ended problem solving behaviors. In *International Conference on Artificial Intelligence in Education*, pages 449–458. Springer, 2015.

[57] Lawrence R. Rabiner. Readings in speech recognition. chapter A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[58] Ka-Chun Wong, Tak-Ming Chan, Chengbin Peng, Yue Li, and Zhaolei Zhang. Dna motif elucidation using belief propagation. *Nucleic Acids Research*, 41(16):e153, 2013.

[59] Cheng Ye, John S. Kinnebrew, James R. Segedy, and Gautam Biswas. Learning behavior characterization with multi-feature, hierarchical activity sequences. In *8th International Conference on Educational Data Mining*, June 2005.

[60] Yi Dong, John S Kinnebrew, and Gautam Biswas. Comparison of selection criteria for multi-feature hierarchical activity mining in open ended learning environments. 2016.

[61] Cen Li and Gautam Biswas. Temporal pattern generation using hidden markov model based unsupervised classification. In *Proceedings of the Third International Symposium on Advances in Intelligent Data Analysis*, IDA '99, pages 245–256, London, UK, UK, 1999. Springer-Verlag.

[62] LR Bahl, Peter F Brown, Peter V De Souza, and Robert L Mercer. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *proc. icassp*, volume 86, pages 49–52, 1986.

[63] Abhijit Gosavi. Reinforcement learning: A tutorial survey and recent advances. *INFORMS Journal on Computing*, 21(2):178–192, 2009.

[64] Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. Monte-carlo tree search: A new framework for game ai. In *AIIDE*, 2008.

[65] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, pages 282–293, Berlin, Heidelberg, 2006. Springer-Verlag.

[66] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[67] Stephen H Edwards. Using software testing to move students from trial-and-error to reflection-in-action. *ACM SIGCSE Bulletin*, 36(1):26–30, 2004.

[68] Satabdi Basu. *Fostering Synergistic Learning of Computational Thinking and Middle School Science in Computer-based Intelligent Learning Environments*. Vanderbilt University, 2016.

[69] Uri Wilensky and I Evanston. Netlogo: Center for connected learning and computer-based modeling. *Northwestern University, Evanston, IL*, 4952, 1999.

[70] Satabdi Basu and Gautam Biswas. Providing adaptive scaffolds and measuring their effectiveness in open ended learning environments. Singapore: International Society of the Learning Sciences, 2016.