

**MASTERING A CURRICULUM THROUGH
ADAPTIVE SEQUENCING: AI BASED MODELS AND
ALGORITHMS**

A THESIS

submitted by

SHABANA K M
(ROLL NO. 111914007)

for the

AWARD OF THE DEGREE

of

DOCTOR OF PHILOSOPHY



INDIAN INSTITUTE
OF TECHNOLOGY
PALAKKAD

Department of Computer Science and Engineering

JUNE 2025

Shabana K M: *Mastering a curriculum through adaptive sequencing: AI based Models and Algorithms*

©Indian Institute of Technology Palakkad

June 2025

CERTIFICATE

This is to certify that the thesis titled *Mastering a curriculum through adaptive sequencing: AI based Models and Algorithms*, submitted by *Shabana K M (Roll No. 111914007)* for the award of the degree of *Doctor of Philosophy* of *Indian Institute of Technology Palakkad*, is a record of bonafide work carried out by her under my guidance and supervision at *Indian Institute of Technology Palakkad*. To the best of my knowledge and belief, the work presented in this thesis is original and has not been submitted, either in part or full, for the award of any other degree, diploma, fellowship, associateship or similar title of any university or institution.

L. Chandrashekhar

Dr. Chandrashekhar Lakshminarayanan
Assistant Professor
Dept. of Data Science and AI
IIT Madras

Members of the Doctoral Committee

1. Dr. Piyush P. Kurur
Associate Professor, Dept. of Computer Science and Engineering
IIT Palakkad
2. Dr. Albert Sunny
Associate Professor, Dept. of Computer Science and Engineering
IIT Palakkad
3. Dr. Sandeep Chandran
Assistant Professor, Dept. of Computer Science and Engineering
IIT Palakkad
4. Dr. M. Ashok Kumar
Associate Professor, Dept. of Mathematics
IIT Palakkad
5. Dr. Lakshmi Narasimhan Theagarajan
Assistant Professor, Dept. of Electrical Engineering
IIT Madras

DECLARATION

I hereby declare that the work reported in this thesis is original and was carried out by me. Further, this thesis has not formed the basis, neither has it been submitted for the award of any degree, diploma, fellowship, associateship or similar title of any university or institution.

*Palakkad,
June 2025*



Shabana K M
(Roll No. 111914007)

Dedicated to my rock, my dear husband **Rameez**
and our lovely little boy **Farid**

"When life gets you down, you know what you gotta do? Just keep swimming." — Dory

"It means you can do whatever you put your mind to" — Jenny

"When something is too hard... There is always another way" — Charlie

"The best things in life happen by chance" — Dory

"Well, you never really know, but when they know, you know, y'know?" — Crush

from *Finding Nemo* (2003) and *Finding Dory* (2016) by Pixar

A B S T R A C T

The advent of Massive Open Online Courses (MOOCs) has tremendously increased the learning opportunities for the masses. However, one drawback of the current course setting (in both MOOCs and regular classroom teaching) is the ‘one-size-fits-all’ strategy of content delivery that does not cater to the individual learning needs and abilities of students. Personalized learning approaches tailored to address the unique needs, skills, and interests of each student have been shown to result in substantial improvements in learning outcomes of students, in addition to offering an immersive learning experience. Artificial Intelligence (AI) is being increasingly utilized in the development of personalized learning systems that use machine learning (ML) algorithms to create optimal customized learning paths or strategies tailored to suit each student’s cognitive abilities and learning requirements, that are learnt through the interactions of the student with the system.

As a part of our research, we developed novel AI based models and techniques for three foundational problems in personalized learning systems. This dissertation describes our research contributions to the problem of teaching a set of related topics, also referred to as a *curriculum*, to a student through adaptive sequencing of activities and learning resources. In this thesis, we examine two different settings for studying a *concept*, which is defined as the smallest unit of learning. The first is the *learning by doing (LbyD)* setting where concepts are learned by doing activities or solving problems that also serve as *assessments* of their mastery of the concept. The second is the *learning by passive engagement (LbyPE)* setting where learning takes place through engagement with various educational resources such as videos, textbook sections and worked-out examples. While much of the previous research on adaptive sequencing in these contexts lacks a rigorous mathematical construct, in our work, we develop personalized sequencing models utilizing principles from learning theory within the frameworks of optimization and decision-theory-based planning. Another novelty is that unlike many of the existing approaches for adaptive sequencing, our algorithms do not rely on any specific knowledge tracing paradigm for modeling the student learning, which improves personalization. Focused on the fundamental elements of *concepts*, *content* and *assessment items*, we propose AI based novel techniques for personalized tutoring of a curriculum to a student through optimal sequencing of content/concepts.

In the first part of the thesis, we consider intelligent tutoring in the LbD setting, wherein the dependencies between the concepts in the curriculum are available as a *curriculum graph*, and a given concept is learned by performing an activity related to that concept (such as solving/answering a problem/question). For this setting, recent works have proposed algorithms based on multi-armed bandits (MAB), where activities are adaptively sequenced using the student response as a direct feedback. We model the problem in the *piecewise stationary multi-armed bandit* framework and propose *CurriculumTutor*, a novel technique that combines a MAB algorithm and a *change point detection* algorithm for adaptive activity sequencing. Our algorithm improves upon prior MAB algorithms for the LbD setting by (i) providing better learning gains, and (ii) reducing hyper-parameters, thereby improving personalisation. We show that our tutoring algorithm significantly outperforms prior approaches in the benchmark domain of two operand addition up to a maximum of four digits.

In the second part of the thesis, we consider the setting where a concept is learned by engaging with different learning resources such as videos, tutorials and textbook sections. Specifically, we address the problem of computing a personalized sequence of learning resources/content to a student that maximizes the probability of mastery of a concept within a time horizon H . We assume that on engaging with a content, the evolution of student's skill depends on (i) the student's learning profile and (ii) the content presented in the past. Mastery is examined at each time step using an assessment, which forms a binary feedback. We formulate this problem as an *episodic fixed horizon Markov Decision Process (MDP)*. We also propose a novel online algorithm, *OPT-CSEQ*, based on upper confidence bounds for generalized linear contextual bandits. *OPT-CSEQ* learns an optimal personalized content sequence over episodes of interactions. We also present the regret analysis of *OPT-CSEQ* and demonstrate its effectiveness via numerical experiments.

In the final part of the thesis, we study the problem of automatically associating math problems with labels of concepts used to solve them, enabling such problems to be used for both practice and assessment of the associated concepts. We observe that in order to accurately identify all the pertinent concept labels for math questions, it is necessary to integrate the problem-solving technique into the tagging procedure. In this context, we designed and conducted an experiment in a graduate-level linear algebra course to understand if student explanations to solving mathematical problems can be employed to generate concept tags for those questions. Based on this exercise, we propose a method called Unsupervised Skill Tagging (UST) to tag math questions based on the most frequent unigrams and bigrams in the explanation text.

This thesis contributes novel AI based models and algorithms designed to assist learners in mastering an educational curriculum using optimal personalized sequencing of activities and content. Our research on optimal activity/content sequencing mechanisms advances the development of efficient adaptive tutoring algorithms for personalized learning systems. The study on concept tagging of mathematical questions opens up future research directions for designing improved tagging procedures for advanced math questions that incorporate problem solving logic into the tagging process.

Keywords: intelligent tutoring systems, adaptive activity sequencing, multi-armed bandits, upper confidence bound algorithm, content-sequencing, question-tagging

PUBLICATIONS

- [1] KM Shabana, Chandrashekhar Lakshminarayanan, and Jude K Anil. "CurriculumTutor: An Adaptive Algorithm for Mastering a Curriculum." In: *International Conference on Artificial Intelligence in Education*. Springer. 2022, pp. 319–331.
- [2] KM Shabana and Chandrashekhar Lakshminarayanan. "Unsupervised Concept Tagging of Mathematical Questions from Student Explanations." In: *International Conference on Artificial Intelligence in Education*. Springer. 2023, pp. 627–638.
- [3] KM Shabana and Chandrashekhar Lakshminarayanan. "OPT-CSEQ: Personalised sequencing of educational content using reinforcement learning." Under review.

*It does not matter how slowly you go
as long as you do not stop.*

— Confucius

ACKNOWLEDGMENTS

My PhD journey has been an incredibly challenging and transformative experience, and I am profoundly grateful to the blessings of the Almighty Allah and the many individuals who supported me along the way.

I am very thankful to my supervisor Dr. Chandrashekhar Lakshminarayanan for his guidance and support throughout my PhD journey. I am particularly grateful to him for exposing me to the domain of *artificial intelligence in education* and guiding my thesis work, despite this not being his primary research area. He has been immensely patient and kind, and has always supported and encouraged me. I sincerely thank him for all the guidance provided.

I am deeply grateful to Dr. Lakshmi Narasimhan Theagarajan for his mentorship during my PhD. He graciously permitted me to conduct one of my research experiments in his course, giving me a lot of flexibility and freedom to carry out the study while also providing very insightful suggestions. He always encouraged me to seek out new avenues to enhance my research experience. I am very much indebted to him for the constant support and firm belief in me, which has inspired me to consistently strive to become a better researcher.

I thank Dr. Albert Sunny, my administrative guide at IIT Palakkad for his support and assistance over the years. I would also like to thank the other members of my doctoral committee, Dr. Piyush P. Kurur, Dr. Sandeep Chandran and Dr. M. Ashok Kumar for their helpful feedback and suggestions. I express my sincere gratitude to Dr. Jasine Babu, the head of the Computer Science and Engineering department, for her constant support and help throughout the years. I am grateful to Mr. Arun Krishnan from the academics section for his timely and patient support in handling various official procedures related to my research work. Thanks to Ms Megha S Mathew, Department Assistant - CSE for all the help. I acknowledge the Prime Minister's Research Fellows (PMRF) scheme for funding my research.

This journey would not have been possible without the unwavering support of my family. I am very much indebted to my dear husband Rameez who has gone above and beyond to support me all these years. I would not have made it through this journey without his backing. He has been extremely kind and patient, constantly encouraging me to push my boundaries. Thank you for always believing in my abilities and motivating me to pursue my dreams. I am indebted to my parents for their love and support, and for always being there when I needed them the most. My mother, in particular, went through a great deal of struggle during my first year of PhD, managing her professional responsibilities in my hometown during her retirement year, while also helping us care for our six-month-old son in Palakkad. And later, whenever I needed to step away to concentrate on my research, she took great care of him so that I could continue to work without worry. I would not have been able to effectively manage my personal and academic life without her staunch support.

My beloved son Farid grew up alongside my research journey, having to share his mother with a demanding pursuit. I am grateful to him for being a consistent source of happiness in my life. His pure, unconditional love has helped me navigate the ups and downs of my PhD journey. I am also thankful to my dear brother Haroon and sister-in-law Raihana for their warm support. I am grateful to my maternal grandparents for their constant encouragement to pursue learning. I also express my thanks to my in-laws.

I am grateful to my dear friends Hari and Vikas for consistently supporting me, believing in me, and cheering for my success. Many thanks to Vinitha, Arun and Sreenath for the many stimulating conversations we shared on research and beyond. I am grateful to Rekha for all the help she provided during my time at IIT Madras. I also thank Ananya for her helpful advice and joyous company while at IITM. Heartfelt thanks to Reshma, Haritha, Ardra, Femin, Athira, Mithul, Anchu, Prashant, Jyothsna, Shafeequ, Sabarija, Reiva, Anupama, Sukanya and Vishnu for their cheerful company.

Pursuing a PhD has been a long-held dream of mine. While the journey has been quite daunting, looking back, it has provided me with invaluable life lessons.

*Life can only be understood backwards;
but it must be lived forwards.
— Soren Kierkegaard*

CONTENTS

1	Introduction	1
1.1	Background	3
1.1.1	Intelligent tutoring systems for personalized learning	3
1.1.2	Artificial Intelligence and Machine Learning in ITS	4
1.2	Fundamental questions addressed in this thesis	6
1.3	Major Contributions	8
1.3.1	Personalized sequencing of concepts to master a curriculum	9
1.3.2	Personalized optimal content sequencing	10
1.3.3	Concept tagging of mathematical questions	11
1.4	Organization of the thesis	12
2	Problem Setting	13
2.1	Domain Model	13
2.2	Tutoring Module	14
2.3	Student Model	16
1	Personalized sequencing of concepts to master a curriculum	
3	Adaptive activity sequencing to master a curriculum	19
3.1	Background: Activity Selection and Mastery Detection in ITS	20
3.1.1	Activity selection	20
3.1.2	Detection of mastery	23
3.2	Problem Setting	26
3.3	Related Work	27
3.4	Summary	29
4	CurriculumTutor: An adaptive algorithm for mastering a curriculum	31
4.1	Adaptive activity sequencing as a piecewise stationary MAB	31
4.1.1	Detecting mastery as a change point detection problem	33
4.2	CurriculumTutor	35
4.2.1	Activity selection using KL-UCB	35
4.2.2	Mastery detection using CUSUM	37
4.2.3	The algorithm	38
4.3	Experiment Setting	39
4.3.1	Generating the Curriculum Graph	40
4.3.2	Student models for simulation	41
4.3.3	Parameters	43

4.4	Results	43
4.5	Discussion	45
4.5.1	Validation with real students	46
4.5.2	Relaxing the assumptions on questions related to a concept	46
4.6	Conclusion	49
 II Personalized optimal content sequencing		
5	Personalized optimal sequencing of content to master a concept	53
5.1	Motivation	54
5.2	Related Work	56
5.3	Problem Setting	59
5.3.1	Student as Context and Effectiveness of Content	60
5.3.2	Skill and Mastery	61
5.4	Summary	62
6	OPT-CSEQ: Personalized optimal sequencing of educational content	63
6.1	Proposed Model	63
6.1.1	Generalised Linear Bandit	64
6.1.2	Markov Decision Process	66
6.1.3	Tree Structure of MDP	66
6.2	OPT-CSEQ algorithm	67
6.3	Theoretical Analysis	71
6.4	Experiments	74
6.5	Discussion	77
6.5.1	Deployment and testing with real students	77
6.6	Conclusion	79
 III Concept tagging of mathematical questions		
7	Automatic generation of concept tags for mathematical questions	83
7.1	Background	84
7.1.1	Skill Model	84
7.1.2	Skill Tagging/Question Tagging	84
7.1.3	Automatic Tagging: Generative vs. Supervised approaches	85
7.2	Automatic tagging of mathematical questions	86
7.2.1	Related work on tagging of mathematical questions	86
7.2.2	Challenges in automatic tagging of math questions	88
7.2.3	Integrating human knowledge into automatic tagging	88
7.3	Problem Definition	89
7.4	Experiment Setup	90

7.5	Manual tagging	92
7.6	Summary	94
8	Unsupervised Skill Tagging (UST)	95
8.1	Analyzing the set of student explanation text	95
8.2	Matching frequent n-grams with concept labels	96
8.3	Unsupervised Skill Tagging (UST)	98
8.4	Tagging using LLMs	99
8.5	Results	100
8.5.1	UST and Manual Tagging	100
8.5.2	LLM based tagging	102
8.6	Discussion	102
8.7	Conclusion	103
9	Conclusion and future directions	105
9.1	Future directions	106
9.1.1	Adaptive sequencing of concepts	106
9.1.2	Optimal adaptive sequencing of content	107
9.1.3	Concept tagging of mathematical questions	107

IV Appendix

A	Details of experiments in personalized concept sequencing	111
A.1	Parameters for the student models	111
B	Experimental Details for Content Sequencing	113
B.1	Setting I (without context)	113
B.2	Additional experiments for Setting I	113
B.3	Setting II (with context)	117
B.3.1	Ω for various student contexts	118
C	Concept Tagging Experimental Materials	121
C.1	Quiz questions used in our study	121
C.2	Frequent n-grams from explanation text of all questions	122
C.3	Concept labels obtained through different techniques	123
C.4	Result produced by GPT-4o for one of the tagging prompts	124
	Bibliography	127

LIST OF FIGURES

Figure 1.1	Architecture of an Intelligent Tutoring System.	4
Figure 1.2	The problem setting.	7
Figure 2.1	The key components of the personalized tutoring system considered in this study.	13
Figure 2.2	A curriculum graph for integer related concepts.	14
Figure 2.3	Zone of Proximal Development (ZPD).	15
Figure 2.4	Zone of Proximal Development (ZPD) defined based on a curriculum graph.	15
Figure 3.1	Zone of Proximal Development (ZPD).	22
Figure 3.2	Multi-armed bandit framework for adaptive activity sequencing in an ITS.	23
Figure 4.1	Detecting mastery using N-consecutive correct responses (NCCR) and moving average (MA) techniques.	34
Figure 4.2	Updation of the ZPD after a concept is mastered.	35
Figure 4.3	An example where CUSUM detects mastery one step ahead of moving average method	38
Figure 4.4	Curriculum graph generated for 2 operand addition up to a maximum of 4 digits.	40
Figure 4.5	Simulation results for Scenario 1 - Student not struggling with any of the concepts.	44
Figure 4.6	Simulation results for Scenario 2 - Student struggling with a few concepts in the intermediate phase.	44
Figure 4.7	Decomposing a concept node into multiple difficulty based sub-nodes.	48
Figure 4.8	Introducing <i>concept-combination</i> nodes for handling questions involving multiple concepts.	49
Figure 5.1	Concept sequencing vs. Content sequencing.	54
Figure 5.2	An illustration of how the order of presentation of content affects student's learning gains	55
Figure 5.3	An example of how the optimal content sequences differ for two students based on their profiles	55
Figure 5.4	The overall framework for personalized optimal content sequencing.	60

Figure 5.5	The skill levels q_t computed for a given sequence of content	61
Figure 6.1	The probability of mastery evolves over time based on Ω and the sequence of content presented.	65
Figure 6.2	Generation of the state space tree for k contents.	67
Figure 6.3	Heat map of the parameter matrix Ω used for experiment.	75
Figure 6.4	Setting I (Without context): Comparing the average number of cases where mastery was attained for the three algorithms.	75
Figure 6.5	Setting I (Without context): The expected probability of mastery of the policies generated by the algorithms at each episode.	76
Figure 6.6	Setting II (with context): The expected probability of mastery for the policies computed by <i>OPT-CSEQ</i> for four different student contexts over 2500 episodes.	76
Figure 7.1	Manual tagging of questions becomes difficult when experts have differing views on the tags to use.	85
Figure 7.2	A sample question from a graduate level linear algebra course.	88
Figure 7.3	A question asked in one of the regular weekly quizzes of the linear algebra course.	90
Figure 7.4	A drag-and-drop activity used in one of the secondary quizzes.	92
Figure 7.5	A sample question from one of the supplementary quizzes for the student to explain how they solved the quiz question.	92
Figure 7.6	Bar plots of concept labels selected by students who correctly solved the problems in Table 7.2.	93
Figure B.1	Setting I (Without context): Comparing the average number of cases where mastery was attained in Run 1.	114
Figure B.2	Setting I (Without context): The average probability of mastery for the policies computed at each episode in Run 1.	114
Figure B.3	Setting I (Without context): Comparing the average number of cases where mastery was attained in Run 2.	115
Figure B.4	Setting I (Without context): The average probability of mastery for the policies computed at each episode in Run 2.	115
Figure B.5	Setting I (Without context): Comparing the average number of cases where mastery was attained in Run 3.	116
Figure B.6	Setting I (Without context): The average probability of mastery for the policies computed at each episode in Run 3.	116
Figure C.1	One of the tagging prompts to GPT-4o.	125
Figure C.2	GPT-4o output for the tagging prompt in Figure C.1.	126

LIST OF TABLES

Table 4.1	Traces generated for addition problems.	40
Table 4.2	Knowledge Components defined for the student models.	41
Table 5.1	Comparison of the settings studied by prior works on educational content sequencing	59
Table 7.1	Concept-labels included in the master tag-list.	91
Table 7.2	Questions associated with the bar plots in Figure 7.6.	94
Table 8.1	Student explanations on solving one of the problems.	96
Table 8.2	The 10 most frequent n-grams in the student explanations for 3 questions along with the <i>expert-defined concept labels</i>	97
Table 8.3	Prompt given to LLMs to perform zero-shot CoT tagging.	100
Table 8.4	Evaluation metrics computed for the 3 settings under a) UST and b) Manual tagging.	101
Table 8.5	Evaluation metrics computed for GPT-4o, Gemini 1.0 and Llama 3.	102
Table A.1	Parameters for BKT, LFA and PFA student models used in our experiment.	111
Table C.1	Quiz questions used in the supplementary quizzes along with the <i>expert-defined concept labels</i> for each question.	121
Table C.2	The 15 most frequent n-grams in the student explanations along with the <i>expert-defined concept labels</i> of each question.	122
Table C.3	Comparing the <i>expert defined concept labels</i> with those obtained from <i>manual tagging</i> and <i>UST</i>	123
Table C.4	Comparing the <i>expert defined concept labels</i> with those obtained from <i>GPT-4o, Gemini</i> and <i>llama 3</i>	124

LIST OF ALGORITHMS

Algorithm 3.1	ZPDES	28
Algorithm 4.1	SELECT-ACTIVITY	36
Algorithm 4.2	DETECT-MASTERY	37
Algorithm 4.3	CURRICULUM-TUTOR	39
Algorithm 6.1	GET-POLICY	69
Algorithm 6.2	OPT-CSEQ	70
Algorithm 8.1	UST	99

INTRODUCTION

Massive open online courses (MOOCs) have dramatically changed the way the world learns by making education accessible at population scale. The advent of MOOC platforms such as Swayam and the National Programme on Technology Enhanced Learning (NPTEL) has revolutionized higher education in India by providing access to high-quality learning resources prepared by the best teachers in the country, available free of cost to any learner. Since they can potentially cater to large numbers of learners without even requiring their physical co-location, such systems are known as learning at scale (LAS) systems. LAS systems can fulfill the ever-increasing demand for upskilling large numbers of people in diverse areas of expertise.

Despite having a substantial amount of learning resources across a wide range of disciplines, and a large number of registered users, the course retention rates in these platforms are found to be alarmingly low, as in the case of MOOCs worldwide. One of the significant drawbacks of the current mode of instruction, in both MOOC and regular classroom teaching, is the lack of personalization - the same course material is delivered to a large population of students regardless of their prior knowledge, abilities, and learning speed. This 'one-size-fits-all' strategy of content delivery does not cater to the learners' individual learning needs. Consequently, a good fraction of the students struggle with learning, adversely impacting the overall quality of education.

Personalized learning approaches, where instructions are tailored to address the individual needs, skills, and interests of each student, have been found to lead to a significant improvement in learning gains for students, apart from providing a more engaging learning experience. Although one-to-one human tutoring can provide individual learning support to a student, it is costly and challenging to scale to a large group of students without multiple tutors. Intelligent tutors powered by Artificial Intelligence (AI) and Machine Learning (ML) can provide personalized learning at scale through adaptive sequencing of instructions based on the individual learning requirements of each student.

Intelligent tutoring systems (ITSs) draw upon principles and tools from three major areas namely human computer interaction, cognitive psychology and AI/ML.

Naturally, research groups have contributed towards ITS by focusing on one or more of these three areas. Broadly speaking, there are two complementary approaches with respect to research in AI/ML for ITS. In one approach standard AI/ML algorithms are applied as off-the-shelf solutions in a ‘plug-and-play’ manner. This approach helps to build attractive new solutions that can be tested and experimented in real world scenarios. The other approach is to pose the underlying problem in ITS as an optimization problem and then design AI/ML algorithms with provable performance guarantees. This approach gives us insights by shedding light on foundational questions such as *optimal sequencing*, i.e., ‘what to teach?’ and *sample complexity*, i.e., ‘how much to teach?’. The problems addressed in this thesis belong to both approaches¹.

In this thesis, we study the problem of teaching a set of related topics, also referred to as a *curriculum*, to a student through adaptive sequencing of activities and learning resources. We examine two distinct learning contexts, namely (i) a *learning by doing* (*LbyD*) setting where concepts are learned by doing activities, and (ii) a *learning by passive engagement* (*LbyPE*) setting where learning takes place through engagement with various educational resources such as videos, textbook sections and worked-out examples. In both settings, we look at the problem of coming up with the optimal personalized sequencing of learning activities (be it doing problems or watching lecture videos). We pose the optimal activity sequencing problem in these two settings as optimization problems within a decision theoretic framework. We then derive two novel algorithms namely *CurriculumTutor* and *OPT-CSEQ* for the LbyD and LbyPE settings. The novelty in CurriculumTutor is that it separates the two key components of the problem: (i) *activity selection*, which is modeled as a multi-armed bandit problem, and (ii) *mastery detection*, which is framed as a change-point detection problem. The novelty in the OPT-CSEQ is that it combines contextual generalised linear bandits for personalisation and Markov decision process based planning to address the boredom vs forgetting trade-off. Unlike many of the existing approaches for adaptive sequencing, our algorithms do not rely on any specific knowledge tracing paradigm for modeling the student learning and work feedback obtained via direct interaction. In other words, our settings and algorithms are fairly generic and have wide applicability.

¹ The emergence of Generative Pre-Trained Transformer (GPT) based models has created opportunities for delivering tailored instruction to learners, as these models assist in addressing student’s questions, clarifying misunderstandings on topics, aiding in problem-solving, and more. However, as of date, these models lack the ability to optimize the sequence of activities, problems, or learning resources in a personalized manner, which limits their effectiveness in maximizing student learning outcomes.

Furthermore this thesis investigates the problem of unsupervised tagging of mathematical questions/problems with associated concept labels. Supervised tagging algorithms for math questions mostly utilize question text and any images or equations provided along with it, for tagging. We propose that in order to effectively identify all the relevant concept labels for advanced math questions, it is essential to incorporate the problem-solving technique into the tagging process. In this thesis, we propose *Unsupervised Skill Tagging (UST)*, a novel approach to tag mathematical questions based on the most frequent n-grams in the set of explanations for each problem.

In the rest of the chapter, we describe the framework under study and then state the major contributions of the thesis.

1.1 BACKGROUND

This section provides an overview of intelligent tutoring systems and describes how machine learning based techniques can be utilized to develop efficient personalized tutoring algorithms.

1.1.1 *Intelligent tutoring systems for personalized learning*

Intelligent tutoring systems (ITSs) were conceived with the aim of offering learners a fully personalized learning experience. ITSs are software programs that create educational settings in which personalized instruction and feedback are provided to learners. ITSs employs an ‘adaptive instructional design’, where the content delivery is automatically customized based on the student’s prior knowledge and learning curve. The learning path created by the ITS is aligned to the specific student needs and learning goals. ITSs can thus improve students’ learning experience, resulting in better learning outcomes compared to traditional teaching methods. The following are the major components of a tutoring system (Figure 1.1).

DOMAIN MODEL The domain model is a structured representation of the knowledge pertaining to the domain being taught. It encompasses the set of skills to be acquired, the inter-dependencies between these skills, the relationships between different forms of learning content, practice questions and the skills involved.

STUDENT MODEL The student model represents the student state that is inferred from performance data during the course of learning. These states typically represent summarized information about the student, which is later used for pedagogical

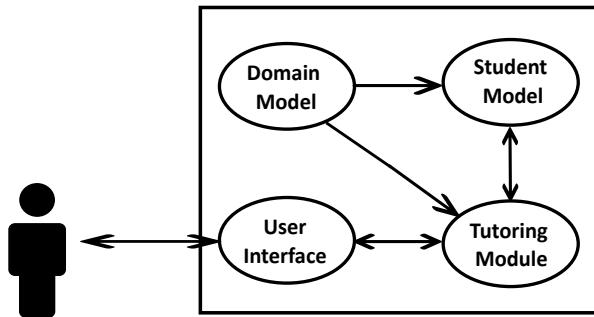


Figure 1.1: Architecture of an Intelligent Tutoring System.

decision-making. Our student model comprises of statistical estimates derived from the history of student responses.

TUTORING MODULE The tutoring module decides the optimal way of presenting the learning material to students - the basis on which the ITS offers activities to a student at each time step.

USER INTERFACE MODEL The user interface model accepts student input in various forms, such as text or clicks, and generates output in multiple formats, including text and diagrams.

An ITS initially determines the current knowledge level of the student against the lesson plan represented in the domain model. The system then walks the student through the schedule and updates the student model in the process. The tutoring module determines the content to be presented at each time step, based on the input from the domain and student models.

Intelligent tutoring systems have been successfully developed and deployed for imparting training across a wide range of domains such as computer programming, mathematics, and science [1–4]. Studies reporting an improvement in learning outcomes for students who use these systems have sparked considerable research interest in this area. For instance, an ITS, called Cognitive Tutor [5], has been widely adopted across schools in the US as a part of the mathematics curriculum, and has reported significant improvements in student learning outcomes in standardized tests [6].

1.1.2 Artificial Intelligence and Machine Learning in ITS

Intelligent tutoring systems have traditionally been challenging to develop, as it requires domain experts to consider every possible learning scenario that students

might encounter and then manually specify the corresponding learning actions in each case. Estimates from real-world projects indicate that developing a single hour of instructional content using an ITS can require approximately 200 to 300 hours of highly skilled labor [7][8].

Utilizing artificial intelligence (AI) based techniques have helped reduce the overall effort required to build ITSs. For instance, *rule-based tutors* such as Cognitive Tutor, capture the knowledge required for step-by-step solution generation by employing production rules that link specific actions to the conditions under which they are appropriate. These models provide tutoring using an algorithm called *model tracing* [9] that tries to interpret the student's solution steps based on some sequence of production firings in the cognitive model. This information allows the system to monitor a student's skill mastery progression over time using algorithms known as *knowledge tracing* [10]. Thus AI based algorithms significantly reduce the development time and effort required for building ITSs.

Machine learning (ML) driven personalized learning techniques utilize machine learning algorithms to analyze student data and produce estimates of student knowledge. These estimates are then employed to automatically generate customized learning schedules that address the specific strengths and weaknesses of each student. AI/ML tools have also been applied to implement features such as giving customized hints and feedback to students [11–13], recommending customized lessons, [14–19] and perform domain modeling by understanding the hidden patterns in a curriculum using natural language processing (NLP) techniques [20–22].

A significant challenge in the development of such systems is to devise tutoring algorithms that can determine the appropriate course of action for the ITS, based on the history of interactions with the student. Examples of decisions that the system must make include determining when to teach a new concept to the student and which learning material to deliver next.

In the context of a reinforcement learning framework, we have the intelligent tutor act as the *agent* that determines at every time step *what to teach*, based on the *environment* representing the cognitive state of the student, i.e., how proficient he/she is in various skills. The cost function might be the amount of time it takes the student to learn a set of skills or the fraction of quiz questions that a student answered incorrectly after fixed hours of training. The environment here is considered *uncertain* because the agent cannot know with complete certainty what the student knows. Instead, the

tutor must infer the student's knowledge state based on the student's responses to test questions and practice problems. Therefore the agent has to act under *uncertainty* and learn optimal teaching policies over interactions while not losing a lot in terms of the cost function during the learning phase.

This dissertation addresses the problem of generating *personalized, optimal* instruction sequences for a set of inter-related topics, also referred to as a *curriculum*. We consider two different learning settings and propose novel reinforcement learning based models and techniques to teach a curriculum using adaptive sequencing. Our algorithms namely *CurriculumTutor* for the *LbyD* setting and *OPT-CSEQ* for the *LbyPE* setting contribute towards the tutor module. *UST* contributes towards the domain model.

In the next section, we describe the fundamental questions studied in this thesis.

1.2 FUNDAMENTAL QUESTIONS ADDRESSED IN THIS THESIS

A concept or skill or knowledge component is a representational unit of educational content and is defined as the smallest element of learning. A Curriculum is a set of inter-dependent concepts that the students are expected to learn. In this dissertation we study an ITS for teaching a curriculum to a learner.

The following are the major components of the tutoring system considered in this study (Figure 1.1). They have been described in detail in Chapter 2.

- **Domain model:** Curriculum graph, Question bank, Content/learning resources
- **Tutoring module:** Reinforcement learning based algorithms
- **Student model:** Statistical estimates computed for each concept based on history of student responses

The *user interface* presents the questions/activities selected by the tutoring module to the student and passes the direct student feedback (answers/responses to questions/activities) as input to the tutoring module.

The tutoring algorithms use a *Zone of Proximal Development (ZPD)* to track the set of concepts that a student can learn at a given time. ZPD is based on an idea from classical psychology and education research, and represents a set of concepts on the boundary of the student's knowledge. In this thesis the ZPD is defined based on the

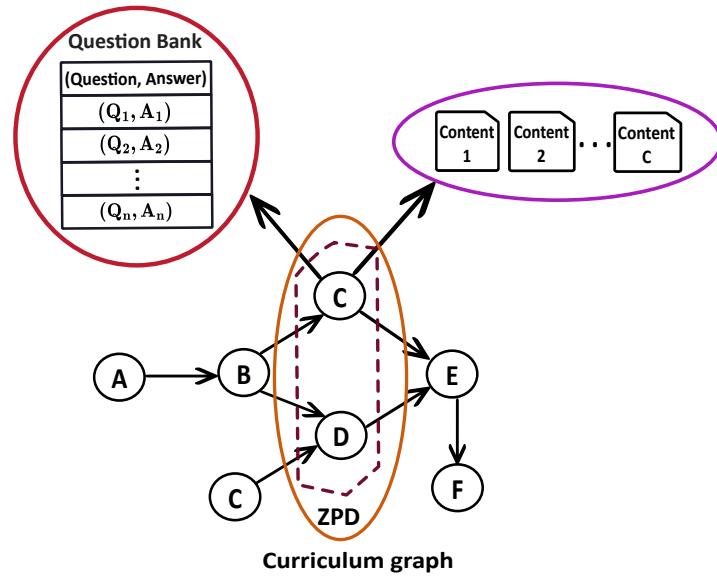


Figure 1.2: **The problem setting.** The core questions that we address in our research are:

Concept sequencing: How can we generate an adaptive sequence of activities based on the ZPD such that student learning gains are maximized? **Content sequencing:** How to find the right order to sequence learning resources that maximizes the probability of mastery within a given time horizon **Question tagging:** How can we tag mathematical questions with labels of concepts that are used to solve them?

curriculum graph that represents the pre-requisite relations between various topics. It gets updated as learning progresses.

Based on this formal setup, the following are the fundamental questions addressed in this thesis. Our work contributes to each of the three components (Figure 1.2).

1. **CONCEPT SEQUENCING:** In the *learning by doing* setting, how can the tutoring algorithm select a concept from the ZPD based on which an activity is presented to a student, such that the student learning is maximized over a given period of time? (1.3.1)
2. **CONTENT SEQUENCING:** For each concept in the ZPD, how can the tutoring algorithm determine an *optimal, personalized* sequence of content to be presented to a student (in the *learning by passive engagement* setting) such that the probability of the student mastering the concept within a given time horizon is maximized? (1.3.2)
3. **TAGGING MATHEMATICAL QUESTIONS:** How can mathematical questions be tagged with labels of concepts that are used to solve them? How can we perform concept tagging when labeled data is not available? (1.3.3)

1.3 MAJOR CONTRIBUTIONS

This section outlines the significant contributions of this thesis with regard to the fundamental questions raised in the previous section. The key contributions can be summarized as follows:

- The personalized sequencing models developed in this thesis utilize principles from learning theory within the frameworks of optimization and decision-theory-based planning.
- Our models do not rely on any specific knowledge tracing paradigm for modeling the student learning, which improves personalization.
- We propose AI based novel adaptive sequencing techniques for personalized tutoring of a curriculum focused on the fundamental elements of *concepts*, *content* and *assessment items*.
- Our tutoring algorithms operate without the need for large-scale student interaction datasets for initial training.
- Our study on concept tagging of mathematical questions paves the way for improved tagging techniques that integrate problem-solving logic into the tagging process.

Our *reinforcement learning-based algorithms* for *concept sequencing* and *content sequencing* automatically learn optimal teaching strategies through interactions with students, in contrast to *rule-based sequencing methods* [23], where domain experts predefine the instructional sequence and corresponding learning actions. Rule-based methods lack the ability to learn from experience, as they cannot automatically update their knowledge base, modify existing rules, or incorporate new ones. Consequently, the responsibility for revising and maintaining the system predominantly lies with the domain expert.

To tag advanced mathematical questions, we proposed a novel algorithm that leverages student explanation texts to identify and assign relevant concept labels —selected from a predefined list — corresponding to the concepts required to solve the question. The contributions are discussed in detail in the following sections.

1.3.1 Personalized sequencing of concepts to master a curriculum

An important problem in an intelligent tutoring system (ITS) is that of adaptive sequencing of learning activities in a personalised manner so as to improve learning gains. In this study, we consider intelligent tutoring of a curriculum in the *learning by doing* (LbyD) setting where a given concept is learned by performing an activity related to that concept. This setting assumes (i) a curriculum graph that represents the prerequisite relationships between the various concepts to be learned, and (ii) access to a question bank that contains a collection of question-answer pairs associated with each concept. In *personalized concept sequencing*, the tutoring algorithm chooses a concept at each time step by considering the student's history of previous interactions. The algorithm then presents the student with an activity from the question bank that is related to the selected concept. The objective here is to help the student master a given curriculum as quickly as possible by maximizing their learning gains.

Recent works [19, 24, 25] have posed the problem of concept sequencing in the framework of multi-armed bandits (MAB), where the arms correspond to the set of concepts and the rewards are computed based on student responses to activities related to the concepts. However these techniques use several hyper-parameters that adversely affects personalization and adds additional computational overhead for tuning.

Our Contribution. In this thesis, we model adaptive activity sequencing as a *piecewise stationary multi-armed bandit* problem and propose a *MAB* and *change point detection* based framework for the task of mastering a curriculum in the LbyD setting. Our main contribution is a novel tutoring algorithm called *CurriculumTutor*, that combines an upper confidence bound bandit algorithm with a log-likelihood based statistical test for change-point detection, to master a curriculum using adaptive activity sequencing. Here, the change point detector separately tracks the mastery level of each concept, while the bandit algorithm manages the exploration vs exploitation tradeoff in activity selection. Our algorithm improves upon prior MAB algorithms for the LbyD setting by (i) providing better learning gains, and (ii) reducing hyper-parameters thereby improving personalisation. We also show that *CurriculumTutor* significantly outperforms prior approaches in the benchmark domain of two operand addition up to a maximum of four digits, especially in scenarios where the learner faces difficulty in learning a subset of concepts in the curriculum graph.

1.3.2 Personalized optimal content sequencing

Digital learning platforms provide a variety of learning resources/materials/content such as videos, tutorials, worked out examples, visualizations, games/tasks, etc to help the student master a given concept. *Content sequencing* is the task of defining the specific order or sequence in which these educational resources are to be presented to a student. Research has demonstrated that the student's learning gains are substantially influenced by an optimal personalized sequencing of content [26].

Traditionally, content sequencing is done in one of the following two approaches. In the first approach, an expert chooses a sequence of content which they deem optimal. This approach lacks personalization, and further, given the diverse modality of the content, it is hard even for an expert to manually arrange them in the right order. In the second approach, the content sequencing is student-driven, wherein, the students are allowed to navigate the platform and engage with the content of their choice. While this achieves personalization, students being non-experts often do not know what is optimal. Thus the conventional approaches do not achieve personalization and optimality simultaneously.

In this study, we address the problem of **Personalized Optimal Content Sequencing** - *What is the optimal sequence of content to be presented to a student in order to learn a concept?* Specifically we address the problem of determining an optimal sequence of content that maximizes the probability of a student mastering the concept within a given time horizon. Here the horizon stands for the number of times the student is engaged with a concept/topic. Say, each week, a new topic is covered and there are 4 to 5 sessions/engagements per week per topic.

Our Contribution. We model the *personalized optimal content sequencing problem* as an episodic finite-horizon MDP. The probability of mastery at a given time depends on the student's knowledge level which is determined by the student's profile and the content presented in the past. Whether or not mastery was achieved is used as a binary reward. Our model captures *forgetting* of the student, *recency* of the content, the *effectiveness of presenting content j immediately after content i*. We also propose *OPT-CSEQ*, a novel online learning algorithm based on upper confidence bounds for generalized linear bandits. *OPT-CSEQ* learns optimal personalized policies over episodes of interactions. The algorithm works in phases comprising (i) determining the maximum likelihood estimate of the parameter θ (ii) optimistic planning based on the student profile and (iii) policy execution, which takes place over several episodes. Further we show

that *OPT-CSEQ* achieves an $\tilde{O}(d\sqrt{TH})$ regret bound over T episodes, where H is the horizon and d is the dimension of the student context. Through numerical experiments, we demonstrate the ability of *OPT-CSEQ* to generate optimal personalized content sequences for students with different profiles and compare its performance against techniques based on (i) random and (ii) greedy policies.

1.3.3 *Concept tagging of mathematical questions*

Educational technology systems such as intelligent tutoring systems captures a student's cognitive abilities or skill levels based on its domain model that contains a mapping of assessment items with tags of skills/concepts that are used to solve them. Assigning concept tags/labels to questions enables these systems to efficiently organize resources, assess student knowledge levels, and thereby decide the best activity or content to be recommended to the student. Therefore generating the correct tags for questions is important as incorrect tagging could adversely affect the performance of automatic tutoring systems.

Manual tagging of questions, performed by experts, is often expensive and can lead to consistency issues when there are different views on the tags to use. Automatic tagging techniques, on the other hand, can efficiently generate consistent tags at lower costs. Several techniques to generate automatic question tags for language related educational content have been proposed in literature. However, automatic generation of tags for mathematical questions is demanding as the question text is mostly succinct and not rich enough to support tagging. The answers to such questions are mostly numerical values, which do not add any value to tag generation. Moreover, generating accurate tags for math questions often requires the tagging algorithm to also include the problem solving logic into the tagging process.

Our Contribution. We study if student explanations on solving a problem could be utilized for unsupervised automatic tagging of advanced math questions. The key idea is that, when asked explicitly to explain their response to a given mathematical problem/question, students who have the right approach to solving a problem would mention the right concepts related to the question in their explanation. We tested our idea in a graduate-level linear algebra course where we asked the students to explain their problem-solving strategy for a selected set of questions in their weekly quizzes. It was observed that the most frequent unigrams and bigrams in the processed explanation text provided by students had good matching with the important concepts

used in solving the problem. We then developed a technique called *Unsupervised Skill Tagging (UST)* to tag questions with labels from a master concept list based on the most frequent n-grams in the set of explanations for each problem. UST demonstrates good performance on our dataset comprising the student explanations which reinforces the idea that student explanations indeed describe the right concepts required to solve a problem, which is exploited by *UST* to generate the correct question tags.

1.4 ORGANIZATION OF THE THESIS

The thesis is organized as follows. Chapter 2 describes the formal problem setting.

PART I of the thesis discusses *personalized sequencing of concepts to master a curriculum in the learning by doing setting*. Chapter 3 discusses the existing methods for adaptive activity sequencing in this context. In Chapter 4 we describe the proposed multi-armed bandit and change point detection based model for concept sequencing and also present our algorithm *CurriculumTutor* built on this framework.

PART II describes our work on *personalized optimal content sequencing*. In Chapter 5, we present the framework, mathematical model and assumptions used to formulate the problem. Chapter 6 describes our episodic finite-horizon MDP model built on generalized linear bandits and details our algorithm *OPT-CSEQ* developed within this framework.

PART III discusses our work on *concept tagging of mathematical questions*. Chapter 7 describes the challenges associated with existing tagging techniques for math questions and details our experiment in a graduate-level linear algebra course, where we collected student explanations on problem solving. In Chapter 8 we present our algorithm *UST* for automatic tagging of math questions based on the on the most frequent n-grams in the set of explanations for each problem.

Finally, the thesis concludes with some future directions in Chapter 9.

2

PROBLEM SETTING

In this chapter we position our setting within the ITS framework introduced in Figure 1.1 and provide a brief description of its key components (Figure 2.1).

2.1 DOMAIN MODEL

The components of the domain model are described below.

- A **concept or skill or knowledge component** is a representational unit of educational content and is defined as the smallest element of learning.
- A **concept/curriculum graph** consists of *concepts* as nodes, with edges representing pre-requisite relationships among the *concepts*. A directed edge from a concept **B** to **C** in Figure 2.2 indicates that concept **C** can be mastered only after mastering concept **B**.
- A *concept* can be learned by engaging with the **content/learning resources** such as video lectures, text book sections, tutorials, worked-out examples, etc. associated with the *concept*.

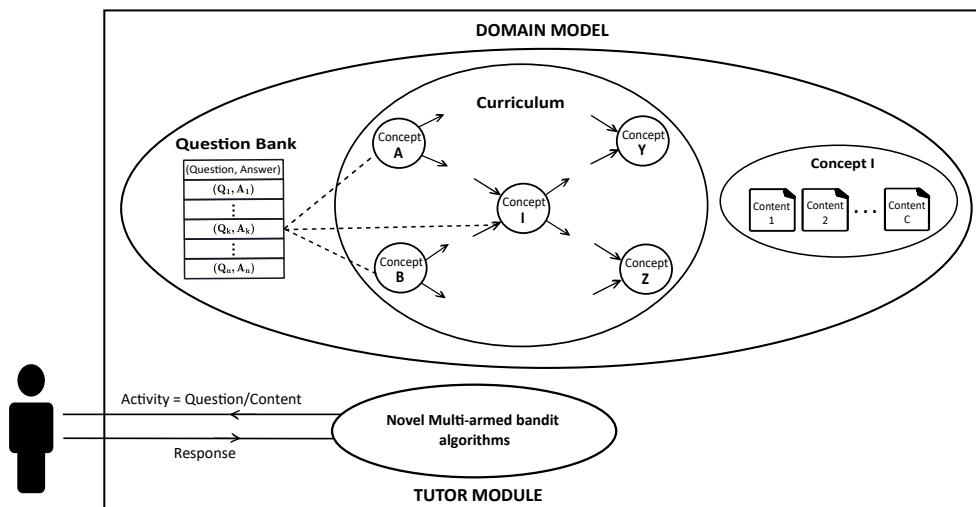


Figure 2.1: The key components of the personalized tutoring system considered in this study.

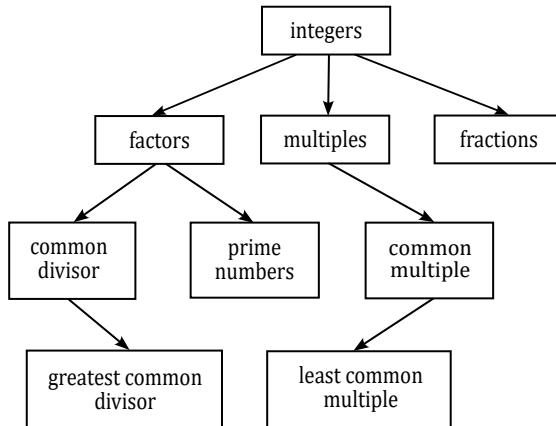


Figure 2.2: A curriculum graph for integer related concepts.

- A **Question bank** has a collection of questions and answers for each concept. The tutoring algorithm uses these question-answer pairs for practice as well as assessment. It is further assumed that *each question is related to exactly one concept* and that *all the questions associated with a concept are of the same difficulty level*.
- Learning systems typically contain a variety of **practice items** or **assessment items** that take many forms such as multiple-choice questions (MCQ), numerical answer type questions (NAT), short text answers, etc. In this dissertation, we consider items relevant mainly in elementary practice, i.e., bottom levels in the Bloom's taxonomy of educational objectives [27], such as recall, recognition and execution concerning factual, conceptual, and procedural knowledge. Students can typically answer these items quickly, and the answer can be automatically evaluated by a learning system.
- The *content* and *assessment items* are tagged with labels of associated *concepts*. The probability of correctly answering an assessment item a that involves concept c depends on the student's mastery of concept c . Conversely, the mastery of concept c is measured by the student's ability to successfully apply the skill to solve any (possibly unseen) item that involves concept c .

2.2 TUTORING MODULE

We consider two settings in which concepts are learned and propose different tutoring modules and algorithms for each context. The first is the **learning by doing** setting where *a concept is learned by performing an activity related to that concept (such as solving/answering a problem/question)*. Teaching primary school students to add two

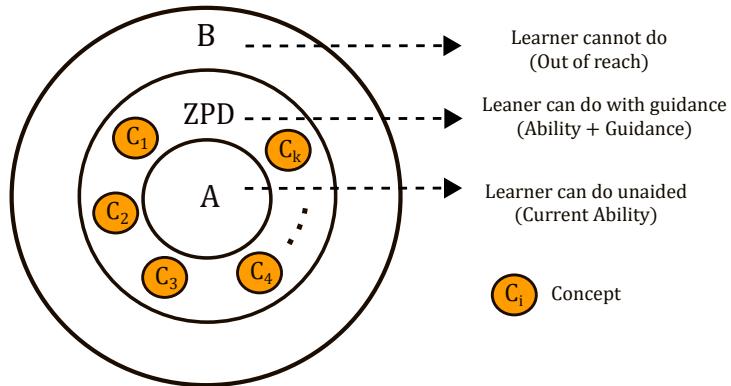


Figure 2.3: Zone of Proximal Development (ZPD).

4-digit numbers is a real world example that fits in this framework. The second setting under study involves *learning through engagement with various forms of content/learning resources, including textbook sections, videos, tutorials, illustrations, and so on.*

Similar to [19, 24, 25] the tutoring algorithms in our learning system employ a *Zone of Proximal Development* or **ZPD** to track the concepts that a student is likely to master at a given time point. **Zone of Proximal Development (ZPD)** is a set of concepts on the boundary of the student's knowledge. It is based on an idea from classical psychology and education research which states that learning is the fastest and most engaging when practicing on material slightly beyond the current abilities of the student (Figure 2.3).

The ZPD, based on the *curriculum graph*, is defined as the set of concepts that have not yet been mastered but whose required prerequisites, as outlined in the *curriculum graph*, have been successfully learned. In Figure 2.4, the ZPD consists of concepts **C, D**

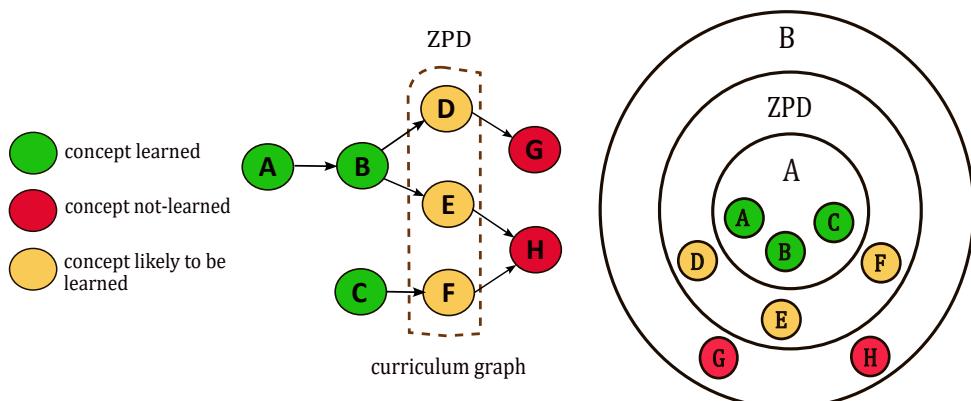


Figure 2.4: Zone of Proximal Development (ZPD) defined based on a curriculum graph.

and **E** as they are the concepts in the *not-learned* state with all of their pre-requisites in the *learned* state. The ZPD gets updated as learning progresses. When a concept within the ZPD is mastered, it is removed from the ZPD, and other concepts in the *not-learned* state, whose prerequisites have been mastered, are added to it.

2.3 STUDENT MODEL

The personalized learning system considered in our study does not employ any specific knowledge tracing model, such as Bayesian Knowledge Tracing (BKT) [28]. Instead the student model in our system tracks the history of correctness of student responses and various statistical estimates required by the reinforcement learning algorithms in the tutoring module, such as average correctness and number of attempts for each concept.

Part I

PERSONALIZED SEQUENCING OF CONCEPTS TO MASTER A CURRICULUM

3

ADAPTIVE ACTIVITY SEQUENCING TO MASTER A CURRICULUM

The tutoring module of an intelligent tutoring system determines the next activity to be presented to a student based on the history of student interactions with the system. This approach of adaptive sequencing of activities enhances the student's learning experience and increases the learning outcome. In this part of the thesis, we study the problem of teaching a student a set of related topics/concepts, also referred to as a *curriculum*, using personalized/adaptive sequencing of activities. We consider intelligent tutoring in the *learning by doing* (LbyD) setting, wherein the *concepts* to be learned along with their inter-dependencies are available as a *curriculum graph*, and a given concept is learned by performing an activity related to that concept (such as solving/answering a problem/question).

For this setting, recent works have proposed algorithms based on multi-armed bandits (MAB), where activities are adaptively sequenced using the student response to those activities as a direct feedback [19, 25, 29]. However, these techniques rely on numerous hyper-parameters that require tuning, which can hinder effective personalization. In this context, we propose a *multi-armed bandit (MAB) and change point detection* based framework for mastering a curriculum using adaptive activity sequencing. We also propose *CurriculumTutor*, a novel technique that combines an upper confidence MAB algorithm and a change point detection algorithm for the problem of adaptive activity sequencing. Our algorithm improves upon prior MAB algorithms for the LbyD setting by (i) providing better learning gains, and (ii) reducing hyper-parameters thereby improving personalisation. We show that our tutoring algorithm significantly outperforms prior approaches in the benchmark domain of two operand addition up to a maximum of four digits.

In this chapter, we begin by providing an overview of the two key functions performed by the tutoring module while teaching a curriculum (Section 3.1). (i) *Activity Selection*: choosing an activity associated with a concept from the set of concepts that a student is capable of mastering at a given time. (ii) *Mastery Detection*: determining when a student has mastered a concept, to avoid assigning additional activities related to it. We then describe the multi-armed bandit framework for adaptive activity sequencing

and then explain different techniques for detecting mastery. Next, we describe the problem setting in detail (Section 3.2) and present the research gaps in the existing approaches for adaptive activity sequencing to master a curriculum (Section 3.3). We present our algorithm *CurriculumTutor* in the next chapter.

3.1 BACKGROUND: ACTIVITY SELECTION AND MASTERY DETECTION IN ITS

Intelligent tutoring systems (ITS) are known to improve learning gains by providing personalized tutoring through adaptive instructional design. Teaching a curriculum to a student involves two key tasks. The first is *Activity Selection* where the tutoring module of an ITS, at each time step t , selects a concept c_t from the set \mathcal{A}_t , based on which a learning activity is presented to the student. \mathcal{A}_t is the set of concepts that can be mastered by the student at time t . The concept $c_t \in \mathcal{A}_t$ is selected based on the student's competence levels for each concept $c \in \mathcal{A}_t$, as estimated from their interactions with the system. This leads to better personalisation of learning and hence higher learning gains. By performing *Mastery Detection*, the tutoring module determines when the student has mastered a concept $c \in \mathcal{A}_t$. When a concept c is determined to be mastered at time t , c is removed from \mathcal{A}_{t+1} . This prevents the ITS from assigning unnecessary practice on concepts that the student has already mastered. In what follows, we delve into each of these aspects in detail.

3.1.1 Activity selection

At the heart of any ITS is the problem of optimal sequencing of activities to the learner. There exist several approaches to automatically choose exercises based on the current knowledge level of students. For instance teaching has been formulated as a planning problem with a Partially Observable Markov Decision Processes (POMDPs) used to model learner behavior over time [30]. Although POMDPs present an extremely flexible framework, they require the exploration of an exponentially large state space. Additionally, learning activity recommendations have been made using algorithms that model students and items based on rating systems like Elo [15]. Many adaptive sequencing approaches rely on knowledge tracing models such as Bayesian Knowledge Tracing (BKT) [16, 28], Deep Knowledge Tracing (DKT) [17] and their variants. Typically, these models have many parameters and identifying all such parameters for a single student is a very hard problem due to the lack of data. Therefore all learners end up sharing the same model with identical parame-

ters. Using the same model for different learners defeats the purpose of personalization.

Given the individual differences between students in terms of background knowledge, learning speed, etc., a sequence that is optimal for one may be inefficient for another student. Also, it has been shown that a sequence that is optimal for the average student is often sub-optimal for most students [31]. Thus, in order for the system to be better adaptable to various types of learners, it is desirable that the tutoring algorithm makes minimal assumptions about the underlying student learning model. A pre-defined population level student model may not be appropriate for each and every individual student and finding parameters that best describe each individual student is quite tedious and time-consuming. Therefore to improve personalization, the system must rely on the student responses to the assigned activities to track learning progress.

Thus an alternative strategy is to rely on methods that do not make any specific assumptions about how students learn and only require information about the estimated learning progress of each activity. Here the tutoring system, at each point in time, proposes to students the activities that are most likely to increase their average skill level, based on their previous interactions with the system. In order to select the right activity, the tutoring module must *explore* to estimate the student progress to different learning activities and at the same time *exploit* by assigning those activities that have been estimated to produce higher learning gains. This fits into the framework of a *multi-armed bandit* problem.

Activity Selection: Multi-armed bandit based algorithms

Multi-armed bandits (MAB) provides a fundamental framework for tackling the “exploration - exploitation” trade off. The vanilla MAB setting consists of a fixed set of actions (arms), and an unknown reward distribution associated with each arm. The goal is to select actions (arms) such that the long term cumulative reward is maximized. The key challenge in this problem is to balance exploiting the information that has already been gained about the rewards from each action and exploring actions where the estimates about their value are still relatively uncertain. Over time one learns which actions are more rewarding on an average, thereby maximizing the long-term cumulative reward.

The advantage of using MAB based tutoring techniques is that they are computationally efficient and provide a personalized learning experience relying on little

domain knowledge and performing the optimization online based on the students' performance. In the process of maximizing cumulative rewards, the MAB framework would pick activities from the concepts that the student tends to learn faster, thereby generating a personalized activity sequence. This formulation is less reliant on the underlying model of student learning and can therefore better adapt to different individual learning behaviors.

Several tutoring algorithms based on the MAB paradigm have been proposed in literature [18, 19, 25, 29]. For instance, MAPLE algorithm combined MAB with personalized difficulty ranking information to present students with challenging problems that had a high likelihood of being solved by the student [18]. The algorithm initially computes a difficulty ranking over questions per student that form the initial question weights. During each round, the question weights are updated based on an exploration factor and a question is randomly selected based on the computed weights. The question weights and the exploration factor are then updated based on the student response to the question. The weight update reflects the change in the algorithm's estimation as to the suitability of each question for the student based on expert guidelines.

Clement et. al.[19] proposed two MAB based tutoring algorithms, namely ZPDES (Zone of Proximal Development and Empirical Success) and RiARiT (Right Action at the Right Time) to select activities from different exercise types and varying difficulty levels based on a user-defined exploration graph. In ZPDES, the learning progress estimated from the student responses to activities is used as reward to perform problem selection. While in RiARiT, rewards calculated based on the difference between the estimated and expected (defined by expert) competence levels of knowledge components (skills) are used to update the weights associated with each activity.

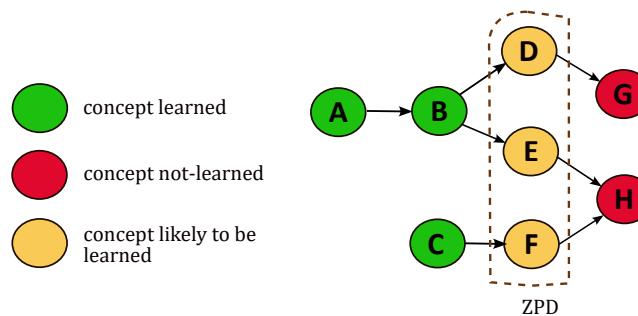


Figure 3.1: Zone of Proximal Development (ZPD).

Activity Selection: Multi-armed bandits and Zone of Proximal Development

The algorithms proposed by Clement et al. employed a Zone of Proximal Development (ZPD) to represent activity types that are both demanding and feasible to master. ZPD is a set of concepts on the boundary of the student's knowledge (Figure 3.1). It is based on an idea from classical psychology and education research which states that learning is the fastest and most engaging when practicing on material slightly beyond the current abilities of the student [32]. The activities/lessons in the ZPD form the arms of the MAB and rewards are computed based on student responses, as seen in Figure 3.2. The system maintains an estimate of the learning progress associated with each concept in the ZPD, based on which it decides the next activity to be offered to the student.

Later Brunskill et. al. [24], used ZPDES to propose an adaptive sequence of activities to advance students through a curriculum graph generated based on an algorithmic representation of the concepts for the domain of two operand addition upto a maximum of 4 digits.

3.1.2 Detection of mastery

Personalizing the learning experience for students also involves providing the right amount of practice to meet the student's needs. Determining the correct amount of

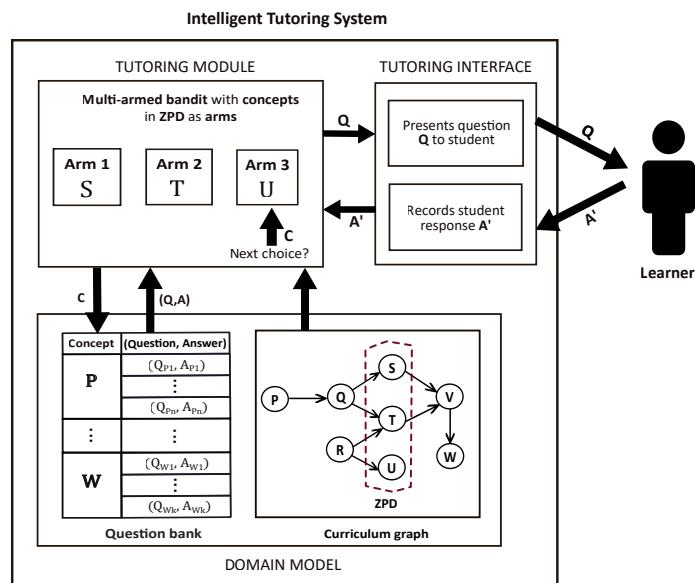


Figure 3.2: Multi-armed bandit framework for adaptive activity sequencing in an ITS.

practice is important because *over-practice* might bore students and lead to a loss in their learning gains while *under-practice* might not provide enough opportunities for a student to learn the concept. To determine an appropriate amount of practice, systems must identify the time instance when a student has learned the concept, also referred to as *reaching mastery*. The tutoring algorithm must utilize the student performance data to infer the mastery of a concept, which is a latent variable.

In an adaptive learning system, the data about student performance is summarized by a model of learner knowledge, also referred to as a knowledge tracing model, or by some summary statistic. When a learner solves a problem or answers a question in an adaptive learning system, the performance summary gets updated and is utilized by the *mastery criteria* to produce a binary verdict: *mastered* or *not-mastered*. The tutoring system adapts its behavior based on this assessment - in case of *non-mastery* more problems based on the same topic are given, otherwise the learner is moved to a different topic.

A typical mastery criterion is a *mastery threshold criterion* [33] that uses a threshold T and declares mastery when the skill estimate (or alternatively the probability of correct answer) is larger than T . Two types of errors are introduced when a *threshold* is used as a criteria to establish mastery. *False positives* occur when a student without sufficient skill level is judged to having mastered a concept. This results in *under-practice*. *False negatives* occur when a student receives additional practice despite having acquired mastery, leading to *over-practice* of the skill. An overly cautious criteria will lead to unnecessary practice (false negatives), whereas a less strict criteria will cause the student to not have adequate learning opportunities to learn the concept (false positives). The value of the mastery threshold influences the relative frequency of these errors. For instance, using a high threshold can result in a reduced number of *false positives* but it may also increase the number of *false negatives*.

Different systems use distinct measures to infer mastery [33–35]. The most commonly used methods to detect mastery include getting N consecutive correct responses or achieving a pre-defined average success rate from the last N tries. Other popular methods are based on models of learner knowledge and the use of probabilistic models that predict the probability of correctly answering the next question. While the majority of these models use the correctness of student responses as input, some learner models incorporate additional information such as the student's response time for detecting mastery [33, 36]. Some of the popularly used techniques for detecting mastery are listed below.

Mastery Detection based on simple statistics of student performance history

N-CONSECUTIVE CORRECT RESPONSES (NCCR). This is one of the simplest, but often practically used mastery criterion. Here the number of consecutive correct responses is counted and mastery is declared when the count reaches the threshold N.

MOVING AVERAGE. Here the average number of correct responses over a moving window of size N is tracked as $\theta_k = \frac{1}{N} \sum_{i=1}^N c_{k-i}$ where k is the time instant when the average is computed and c_i is the correctness of the student response at time i. Mastery is declared when $\theta_k \geq T$ where T is the *mastery threshold*.

EXPONENTIAL MOVING AVERAGE. Here the moving average approach is used with exponential weights so that more weightage is given to recent attempts. $\theta_i = \frac{\sum_{i=1}^k w_i \cdot c_{k-i}}{\sum_{i=1}^k w_i}$ where w_i is an exponential function. The mastery criterion remains $\theta_k \geq T$

These techniques have certain limitations. A drawback of the NCCR technique is that *slips* made by a student have significant impact on practice opportunities. A *slip* is defined as an incorrect response by a student who has mastered the topic. When using NCCR to determine mastery, students who slip are penalized heavily, requiring them to complete additional unnecessary practice. Similarly, for the moving window-based averaging methods, determining optimal values for the parameters N and T can be difficult. Choosing appropriate values for N and T requires striking a balance between *false positives* and *false negatives*, taking into account student guesses and slips.

Mastery Detection based on learner models

A more sophisticated approach to detecting mastery involves utilizing learner models. These models, also known as *knowledge tracing* models, track the student knowledge of a concept/skill over time and estimate the probability of correctly answering the next question related to the concept. Some of the popular knowledge tracing models include Bayesian Knowledge Tracing (BKT) [28], Performance Factors Analysis (PFA) [37] and Deep Knowledge Tracing (DKT) [17]. The learner models are naturally used with the mastery threshold rule i.e., mastery is declared once the estimate of knowledge or the probability of correctly answering a question surpasses a given threshold.

One disadvantage of using learner models for mastery detection is that a substantial amount of student performance data is required to learn the parameters and fit a model. Also knowledge tracing may not be particularly effective in the first few student attempts when the model is very susceptible to initial parameter values [35].

3.2 PROBLEM SETTING

In this part of the dissertation, we consider the *learning by doing* (LbyD) setting used by Brunskill et al. [24], wherein the concepts to be mastered are available in the form of a curriculum graph and the students learn a concept by performing activities related to the concept.

In the *Learning by Doing* (LbyD) setting, a *concept* is learned by performing an activity related to that concept (such as solving/answering a problem/question). A *concept/curriculum graph* has concepts as nodes and the edges represent dependency/pre-requisite relations between concepts (Figure 2.2). Several methods have been proposed in the literature to construct concept graphs from text corpus [20][21]. While the exploration graph used in [19] was expert-defined, Brunskill et al. constructed knowledge graph automatically from execution traces of problems [38]. In this thesis, we assume access to a concept graph for the curriculum to be mastered. The research problem of interest can then be stated as follows:

Curriculum sequencing in the LbyD setting: Given a curriculum of *inter-dependent* topics to be mastered, along with a partial ordering of the pre-requisite relations between the concepts, generate a personalized sequence of activities to master the curriculum such that the *learning gain* of the student is maximized.

Terminology

- A curriculum is said to be mastered by a student when the student has *mastered all the concepts* within the curriculum.
- An *Activity* refers to a question or a problem presented to a student in order to practice a concept.
- *Learning gain* is defined as the number of concepts mastered over a given number of activities.

Assumptions

- In the *learning by doing* setting, students first have access to some theoretical knowledge about the concepts, but learning happens with practice.

- Activities are selected from a question bank that contains a collection of question answer pairs associated with each concept.
- For a student s , the mastery of a concept at time t is measured by the ability of s to correctly perform an *activity* involving that concept, i.e. by their ability to generalize to unseen exercises.
- A student does not forget a concept once it has been mastered.
- Each activity is associated with exactly one concept.
- All activities are of the same difficulty level.
- The student response to an activity can be deemed to be either *correct* or *incorrect*.

3.3 RELATED WORK

We first describe the ZPDES algorithm proposed by Clement et. al. [19] and then look at the research gaps in ZPDES, in particular about how the hyper-parameters in ZPDES adversely affect personalization.

Zone of Proximal Development and Empirical Success (ZPDES) (see Algorithm 3.1) is based on the multi-armed bandits framework for adaptive activity sequencing where pre-conditions between activities are provided in the form of an expert defined curriculum graph. At each time-step, ZPDES selects the next activity a' based on the normalized weights of activities ($w'_{a'}$) in the ZPD as well as an exploration factor γ , $0 \leq \gamma \leq 1$. The correctness of student response to the activity a' at the i^{th} attempt, $C_{a',i}$, is a binary random variable - it takes a value of 1 for a correct response and 0 for incorrect response. The reward is computed as the difference between the number of successes in the most recent $d/2$ samples and the number of successes in the preceding $d/2$ samples, where d is the window-size. The reward provides an empirical measure of how the success rate is increasing. The reward becomes close to zero when either a concept has been mastered or the student struggles to master it. A high positive reward indicates good learning progress, whereas a negative reward implies stalled learning. These rewards are then used to update the activity weights (w_a) based on the hyper-parameters $\eta > 0$ and $\beta \in [0, 1]$. ZPDES doesn't propose a method to determine if a concept has been mastered or not. Results from user studies conducted with school students demonstrated that the algorithm led to a significant increase in learning speed by presenting personalized activity sequences.

Algorithm 3.1: ZPDES

Initialization: For each activity a in ZPD in the not-learned state set $w_a = w_0$

Selecting the next activity: The weights (w_a) of the activities in the ZPD are normalized (w'_a) and an activity is sampled from the probability distribution p_a

$$w'_a = \frac{w_a}{\sum_{a' \in \text{ZPD}} w_{a'}}, \quad p_a = w'_a(1 - \gamma) + \gamma \frac{1}{|\text{ZPD}|} \quad a \in \text{ZPD}$$

Reward for the chosen activity a' : $r = \sum_{k=n_{a'}-d/2+1}^{n_{a'}} \frac{C_{a',k}}{d/2} - \sum_{k=n_{a'}-d}^{n_{a'}-d/2} \frac{C_{a',k}}{d/2}$

n_a is the number of times activity a has been chosen

Updating weight of activity a' : $w_{a'} \leftarrow \beta w_{a'} + \eta r$

A variant of ZPDES was also proposed by Clement et al. [29] where the ZPD was expanded to include another activity type when the student success rate over all active activities in the ZPD reached a threshold and an activity type gets removed from the ZPD when its recent success rate crosses a threshold. Later Brunskill et. al. [24] proposed a novel system that combines automatic curriculum ordering [38] with ZPDES for automatically and adaptively advancing a student through a curriculum. A ‘sliding windowed’ average was used in this framework for inferring mastery, where a concept is determined to be *mastered* when the accuracy over the past d attempts reaches above a specified threshold t . This system was then extended to account for the effects of forgetting [25].

Scalability: For fixed hyperparameter settings, ZPDES and other bandit-based approaches for curriculum sequencing offer efficient scalability. This is because activity selection at each time step only involves utilizing the statistics of arms corresponding to the concepts within the learner’s ZPD, resulting in a computational complexity that is linear in the number of concepts in the ZPD. ZPDES has previously been validated through user studies involving 400 students aged 7–8 years across 11 schools. [19].

Research Gap: The major drawback of ZPDES is the use of many hyper-parameters such as d, w_0, γ, β and η , which are critical for its deployment, and eventual success. Using fixed values for these hyper-parameters would have an adverse effect on personalisation since the same set of hyper-parameters may not be effective for the entire population. On the contrary, tuning these hyper-parameters for different individuals is an additional computational overhead and a time consuming process.

3.4 SUMMARY

Online planning of activity sequences provides a personalized learning experience to students that significantly influences their motivation and learning. This chapter explored the multi-armed bandit framework for adaptive activity sequencing and discussed several existing techniques to assess mastery of a concept. We then described the problem of utilizing adaptive activity sequencing to master a curriculum in the learning by doing setting, where concepts are learned by doing activities related to that concept. The existing approaches for this problem were discussed and the important research gaps were examined.

4

CURRICULUMTUTOR: AN ADAPTIVE ALGORITHM FOR MASTERING A CURRICULUM

In this chapter, we first present our *multi-armed bandit (MAB) and change point detection* based approach for mastering a curriculum using adaptive activity sequencing (Section 4.1). We observe that the reward function in the MAB framework for activity selection is *non-stationary* due to improvements in student performance measures as the student learns a concept. We address this issue by modeling the adaptive activity sequencing problem in the *piecewise stationary multi-armed bandit* framework where an efficient bandit algorithm is combined with a change point detection technique to handle the non-stationarity of reward distribution. In our approach we separately address two important components of the problem where (i) the MAB takes care of the exploration vs exploitation tradeoff in *activity selection*, and (ii) the change-point detector addresses the non-stationarity of the student responses by *detecting mastery*.

We then describe CurriculumTutor, a novel adaptive tutoring algorithm to master a curriculum, based on the proposed approach (Section 4.2). We first discuss the activity selection and change point detection procedures used in CurriculumTutor and then present the complete algorithm. Next we present the experiment setup to evaluate the performance of CurriculumTutor with simulated students based on 3 different knowledge tracing models (Section 4.3). We show that our algorithm significantly outperforms prior approaches in the benchmark domain of two operand addition up to a maximum of four digits, especially in scenarios where the learner faces difficulty in learning a subset of concepts in the curriculum graph (Section 4.4). In (Section 4.5) we present a detailed discussion of our algorithm.

4.1 ADAPTIVE ACTIVITY SEQUENCING AS A PIECEWISE STATIONARY MAB

As discussed in the previous chapter, prior works have proposed algorithms that utilize multi-armed bandits for activity selection in the curriculum sequencing setting. In this section, we describe how this problem can be modeled as a *piecewise-stationary MAB*, where the reward distribution of each arm is piecewise constant with respect to time.

In the MAB based framework for curriculum sequencing, the concepts/topics in the ZPD form the arms and the reward is computed based on student performance measures such as correctness of student responses. However, a particularity here is that as the student learns a topic, the student performance measures improve leading to an increase in rewards. This makes the reward function non-stationary, which necessitates the implementation of specific mechanisms to track its evolution.

Learning progress as a piecewise-stationary model: Bayesian Knowledge Tracing (BKT) [28], a widely used student learning algorithm [10, 16, 39], assumes a binary knowledge state — either mastered or not—implying that the transition from non-mastery to mastery occurs in a single time step following a practice opportunity. In contrast, logistic knowledge tracing models, such as Learning Factors Analysis (LFA) [40] and Performance Factor Analysis (PFA) [37], characterize learning as a gradual process, modeling a power-law relationship between error rate and practice. BKT can be considered as a special case of a piecewise-stationary learning function characterized by a single transition, whereas logistic models correspond to functions with multiple transitions. Therefore we assume a piecewise-stationary learning model that generalizes and accommodates both these perspectives.

Our problem can thus be more precisely modeled as a *piecewise-stationary MAB problem* where the reward distribution of each arm is piecewise constant with respect to time, i.e., remains constant for a certain period, and shifts at some unknown time step referred to as a change-point. The change-point or break-point here corresponds to the time instance when a student learns/masters a concept.

The piecewise-stationary bandit model has been extensively studied. Several algorithms have been proposed under this setting where an efficient bandit algorithm such as UCB or KL-UCB is combined with a change point detection framework [41–43]. The change point detection algorithm monitors the distribution of each arm, and sends out a positive signal to restart the bandit algorithm once a breakpoint is detected in any of the arms. The restart can be made in two ways - *global restart*, where the history of all arms are reset once a change-point is detected on one of them, or *local restart* where the history of an arm is reset when a change-point is detected on that arm.

The main reason for the presence of hyper-parameters in ZPDES and other prior techniques is that they try to address together both the exploration vs exploitation tradeoff as well as the non-stationarity of the student responses. On the contrary, we propose a scheme based on the bandit and change point detection framework

for the piecewise-stationary MAB problem. Our approach involves breaking down the adaptive activity sequencing problem into two components: (i) **activity selection** which is performed by a *multi-armed bandit framework* that balances exploration and exploitation and (ii) **mastery detection** performed using a *change point detector* that deals with the non-stationarity of the student responses. The MAB framework for activity selection has been described in detail in the previous chapter. In the next section, we present our approach to modeling detecting mastery as a change point detection problem.

4.1.1 Detecting mastery as a change point detection problem

The change-point detection problem can be formally described as follows [44]: Suppose X_1, X_2, \dots are independent random variables observed sequentially and X_1, \dots, X_{m-1} have distribution function F_0 for $m \in \{1, 2, \dots\}$ while X_m, X_{m+1}, \dots have distribution function $F_1 \neq F_0$. A sequential change-point detection procedure is identified with a stopping time $\tau \in \mathbb{N}$ for an observed sequence $\{X_n\}_{n \geq 1}$, i.e., the time of alarm τ , at which it is declared that a change has occurred. Thus $\tau \in \mathbb{N}$ is a random variable depending on the observations X_1, \dots, X_τ . A good detection procedure should have a low false alarm rate (FAR) and small values of the expected detection delay, provided that there is no false alarm. The performance of the detection procedure is given by the quantities *average detection delay* (ADD) and *FAR* defined as follows [45]:

$$\text{ADD}_m(\tau) = E_m(\tau - m | \tau \geq m)$$

$$\text{FAR}(\tau) = \frac{1}{E_0(\tau)}$$

where $E_0(\tau)$ denotes the expectation of the sequence $\{X_n\}_{n \geq 1}$ when there is no change, i.e., $m = \infty$

To model detection of mastery in the change point detection framework, we assume that each concept is associated with two states - *learned* and *not-learned*. In the *not-learned* state, a student can guess the solution of a problem correctly with probability $p_{\text{guess}} < 0.5$. In the *learned* state, a student can slip and answer incorrectly with probability $p_{\text{slip}} < 0.5$. Let r_i denote the correctness of the student response at the i^{th} attempt. When the concept has not been mastered, $r_i \sim P_{\text{not-learned}}$, where

$$P_{\text{not-learned}}(r = 1) = p_{\text{guess}} \quad P_{\text{not-learned}}(r = 0) = 1 - p_{\text{guess}} \quad (4.1)$$

When the concept has been mastered $r_i \sim P_{\text{learned}}$, where

$$P_{\text{learned}}(r = 1) = 1 - p_{\text{slip}} \quad P_{\text{learned}}(r = 0) = p_{\text{slip}} \quad (4.2)$$

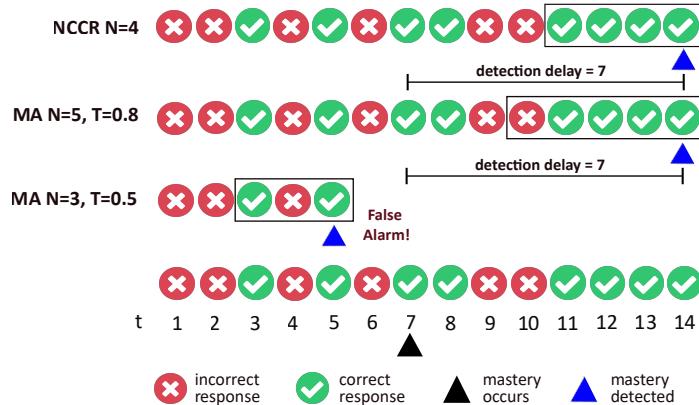


Figure 4.1: Detecting mastery using N-consecutive correct responses (NCCR) and moving average (MA) techniques. In Figure 4.3 we compare moving average with CUSUM, a statistical change-point detection method.

When a student learns a concept, it moves from the *learned* state to the *not-learned* state and there is an associated shift in the distribution of correctness of student responses r_i from $F_0 = P_{\text{not-learned}}$ to $F_1 = P_{\text{learned}}$. Here the change-point corresponds to the unknown time step at which the concept is learned i.e., a transition occurs from the *not-learned* to *learned* state.

NCCR and moving window based techniques can also be considered as change-point detection techniques where a change-point is detected at time t when N consecutive correct 1s are obtained and $\theta_t \geq T$ respectively. These strategies are agnostic to the initial distribution F_0 and try to model F_1 by utilizing the parameters N and T . Here lower values of the parameters N and T increases the probability of false alarm (due to the possibility of guessing), whereas the parameters taking higher values leads to an increase in the detection delay (resulting from slipping), as shown in Figure 4.1. Here, moving average (MA) with $N = 3$ and $T = 0.5$ raises a false alarm at $t = 5$ before mastery occurs at $t = 7$, due to the student correctly guessing answers to two questions. With $N = 4$ and $T = 0.8$, both NCCR and MA detect mastery at $t = 14$ with a detection delay of 7. Therefore, it is important to choose values of N and T that result in a minimal FAR and a small expected detection delay in the absence of false alarms. We compare MA with CUSUM, a statistical change-point detection method, in Figure 4.3.

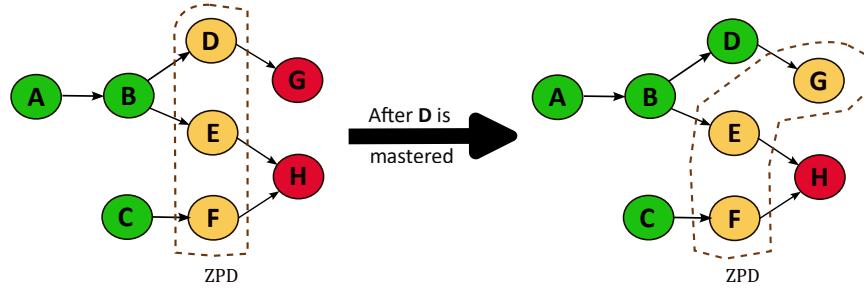


Figure 4.2: Updation of the ZPD after a concept is mastered.

4.2 CURRICULUMTUTOR

In this section, we describe our algorithm *CurriculumTutor* that combines an upper confidence bound bandit algorithm (for *activity selection*) with a change point detection technique (for *mastery detection*). CurriculumTutor employs the correctness of student responses as the reward in the bandit framework for *activity selection*, and also uses it as an input for the change point detection scheme to *detect mastery*.

Similar to prior approaches [19, 24, 25], *CurriculumTutor* uses a ZPD to monitor the concepts that the student is likely to master at each time instant. The ZPD encompasses concepts that have not yet been mastered, but for which the necessary pre-requisites, as specified by the *curriculum graph*, have been mastered. The ZPD gets updated as learning progresses. When a concept in the ZPD is mastered, it is removed from the ZPD and other concepts which have their pre-requisites already mastered gets added to the ZPD, as shown in Figure 4.2, where the nodes colored green and red represent the concepts that have been *mastered* and *not-mastered* respectively. This continues until the student has mastered all the concepts in the curriculum.

4.2.1 Activity selection using KL-UCB

As discussed in the previous chapter, CurriculumTutor uses a multi-armed bandit framework to perform activity selection where the concepts in the ZPD form the arms. In order to frequently select concepts that are providing higher learning progress to the student, the correctness of student response is used as reward. The rewards are binary, i.e., a reward of one is obtained if an activity related to the selected concept is correctly performed and otherwise zero. This fits into the scheme of a *Bernoulli multi-armed bandit*, where for each arm, a reward of 1 is obtained with an unknown probability p that is dependent on the knowledge level of the associated

Algorithm 4.1: SELECT-ACTIVITY

Input: $t, ZPD_t, (n_c(t), s_c(t))$ for each concept $c \in ZPD_t$
 \mathcal{A}_c : a set of activities/problems for each concept $c \in ZPD_t$
if $\exists c \mid n_c(t) = 0$ **then return** $a \in \mathcal{A}_c$;
 $n_t := \sum_{c \in ZPD_t} n_c(t)$
 $\mu_c := \frac{s_c(t)}{n_c(t)}$ for $c \in ZPD_t$
 $z := \arg \max_{c \in ZPD_t} \max \left\{ \tilde{\mu} \in [0, 1] \mid D_{KL}(\mu_c \parallel \tilde{\mu}) \leq \frac{\log f(t)}{n_c} \right\}$
return $a \in \mathcal{A}_z$

concept(arm). Therefore CurriculumTutor utilizes the *Kullback-Leibler Upper Confidence Bound (KL-UCB)* algorithm to perform activity selection, as it is shown to have better performance bounds than UCB and its variants [46] in the Bernoulli setting.

Since the ZPD gets updated whenever the mastery of a concept is detected by the change point detector, a *local restart* is done similar to [41]. Here the history of all arms in the ZPD except the one which has been mastered is retained.

The *activity selection* procedure based on KL-UCB has been described in Algorithm 4.1. The algorithm takes as input:

- (i) ZPD_t that includes the concepts that a student is capable of mastering at time t .
- (ii) for each concept $c \in ZPD_t$, a tuple $(n_c(t), s_c(t))$ where $n_c(t)$ is the total number of activities presented based on c until time t and $s_c(t)$ is the total reward attained at time t .
- (iii) a set of activities \mathcal{A}_c for each concept $c \in ZPD_t$.

If there exists a concept c in ZPD_t based on which no activity has been presented to the student yet, i.e., $n_c = 0$, then an activity based on c is presented next. Otherwise, the UCB estimates for each of the concepts in the ZPD are computed as shown in Equation 4.3. Due to local restart, the UCB indices are computed based on $f(n_t)$ where $n_t = \sum_{c \in ZPD} n_c(t)$. The concept with the highest UCB estimate is chosen based on which an activity is presented to the student.

$$UCB_c(t) = \max \left\{ \tilde{\mu} \in [0, 1] \mid D_{KL}(\mu_c \parallel \tilde{\mu}) \leq \frac{\log f(t)}{n_c} \right\} \quad (4.3)$$

where $f(t) = 1 + t \log^2(t)$, $D_{KL}(\cdot, \cdot)$ is the relative entropy between Bernoulli distributions and $\mu_c = \frac{s_c}{n_c}$.

Algorithm 4.2: DETECT-MASTERY

```

Input: c, obs, h, pg, ps
if obs = 1 then est :=  $\frac{1-p_s}{p_g}$ ;
else est :=  $\frac{p_s}{1-p_g}$ ;
 $\tilde{U}_c := \max(0, \tilde{U}_c + \log(\text{est}))$ 
if  $\tilde{U}_c \geq h$  then return True;
return False

```

4.2.2 *Mastery detection using CUSUM*

CurriculumTutor uses *Cumulative Sum (CUSUM)* as the change-point detection algorithm for detecting mastery. CUSUM uses the log-likelihood ratio (LLR) to test the hypotheses that a change occurred at the point λ and that there is no change at all ($\lambda = \infty$) which is defined as:

$$Z_{n,\lambda} = \sum_{k=\lambda}^n \log \frac{p_1(X_k | X_1, \dots, X_{k-1})}{p_0(X_k | X_1, \dots, X_{k-1})}, \quad n \geq \lambda \quad (4.4)$$

Here, p_0 and p_1 are the pre-change and post-change probability density functions respectively. The maximum LLR statistic $U_n = \max_{1 \leq \lambda \leq n} Z_{n,\lambda}$ is compared with a threshold h and a change is detected when the value of U_n exceeds h . When $h > 0$ and the observations are independent and identically distributed (i.i.d.), U_n can be replaced by the statistic \tilde{U}_n which obeys the recursion:

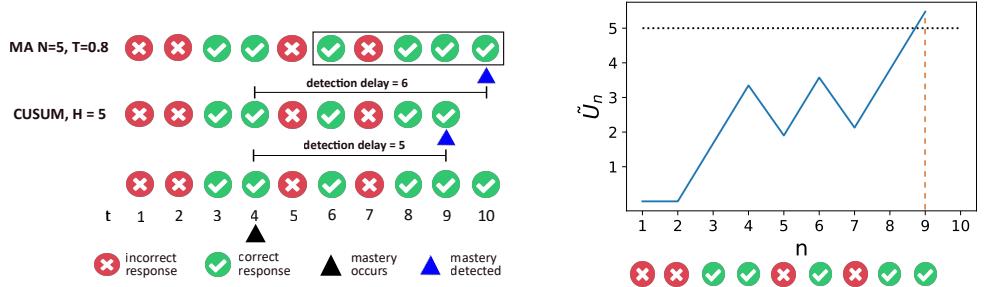
$$\tilde{U}_n = \max \left\{ 0, \tilde{U}_{n-1} + \log \frac{p_1(X_n)}{p_0(X_n)} \right\} \quad (4.5)$$

with the initial condition $\tilde{U}_0 = 0$. With p_{guess} and p_{slip} defined by the user, the pre and post change distributions are as follows:

$$p_0(k) = p_{\text{guess}}^k (1 - p_{\text{guess}})^{(1-k)} \quad (4.6)$$

$$p_1(k) = (1 - p_{\text{slip}})^k (p_{\text{slip}})^{(1-k)} \quad \text{for } k \in \{0, 1\} \quad (4.7)$$

It has been shown [44] that in the i.i.d. case CUSUM minimizes the worst case average detection delay among all the detection algorithms for which the FAR is fixed at a given level $\overline{\text{FAR}}$. See Figure 4.3 for an instance where CUSUM detects mastery one time step earlier than a moving average technique with window size $N = 5$ and mastery threshold $T = 0.8$. Here mastery is assumed to have occurred at $T = 4$. While the moving average technique determines detects mastery at $T = 10$, CUSUM with $H = 5$, $p_{\text{guess}} = 0.15$, $p_{\text{slip}} = 0.2$ determines mastery at $T = 9$. In this case, the detection delay is 5 for CUSUM, compared to 6 for the moving average method.



(a) The detection delay for CUSUM and Moving Average (MA) technique.
(b) A plot illustrating the variation of \tilde{U}_n with each response.

Figure 4.3: CUSUM (with $H = 5$, $p_{\text{guess}} = 0.15$, $p_{\text{slip}} = 0.2$) detects mastery at $T = 9$, one step ahead of moving average (MA) technique (with $N = 5$, $T = 0.8$) at $T = 10$.

The *mastery detection* procedure based on CUSUM has been described in Algorithm 4.2. The algorithm uses the \tilde{U}_n statistic for change-point detection, as the correctness of student responses are independent random variables given the state of each skill. The \tilde{U}_c statistic is tracked for each concept $c \in \text{ZPD}$, which is initialized to 0 when a concept gets added to the ZPD. The algorithm takes as input:

- (i) the concept c for which mastery is to be detected
- (ii) the observation obs , which denotes the correctness of student's response to the activity, and takes values True or False.
- (iii) the threshold h for the LLR statistic
- (iv) the guess probability of the activity p_g
- (v) the slip probability p_s

Based on the value of obs , \tilde{U}_c is updated using (4.5). If the updated \tilde{U}_c exceeds h , the algorithm returns True indicating that a change-point has been detected, and False otherwise.

4.2.3 The algorithm

CurriculumTutor has been described in Algorithm 4.3. The algorithm maintains a ZPD that consists of the concepts in the student's knowledge boundary. Initially the ZPD consists of the minimal elements of the partial ordering curriculum P . The initial values of n_c , s_c and \tilde{U}_c are set to zero for each concept $c \in \text{ZPD}$. At each time step t , the algorithm uses the SELECT-ACTIVITY procedure (Algorithm 4.1) to select a concept

Algorithm 4.3: CURRICULUM-TUTOR

```

Initialize zpd := Minimal elements of the partial ordering curriculum, P
for  $c \in zpd$  do  $n_c := 0, s_c := 0, \tilde{U}_c := 0;$ 
 $t := 0$ 
repeat
|    $t := t + 1$ 
|    $(a_t, c_t) := \text{SELECT-ACTIVITY}(t, zpd)$     (Algorithm 4.1)
|    $r_t := \text{Correctness of student response to } a_t$ 
|    $n_{c_t} := n_{c_t} + 1, s_{c_t} := s_{c_t} + r_t$ 
|   if DETECT-MASTERY( $c_t, r_t$ ) then      (Algorithm 4.2)
|       |   Mark  $c_t$  as learned
|       |   Remove  $c_t$  from  $zpd$ 
|       |    $S \leftarrow \{q' \in P \mid (c_t \leq q') \wedge (\nexists p \in P \text{ such that } p \leq q' \wedge p \text{ is not-learned})\}$ 
|       |    $zpd \leftarrow zpd \cup S$ 
|       |   for  $c \in S$  do  $n_c := 0, s_c := 0, \tilde{U}_c := 0;$ 
until  $zpd$  is empty;

```

c_t from the ZPD based on which an activity a_t is presented to the student. Based on r_t , the correctness of the student's response to a_t , the values of $c_t - n_{c_t}$ and s_{c_t} are updated. Next DETECT-MASTERY procedure (Algorithm 4.2) is used to determine if mastery is achieved for c_t , based on the observation r_t . The concept c_t is removed from the ZPD when it is *determined* to have been transitioned into the learned state. The ZPD is then updated with concepts in the not-learned state all of whose prerequisites are in the learned state, as shown in Figure 4.2. This continues until the student has mastered all the concepts in the curriculum graph.

4.3 EXPERIMENT SETTING

We will now describe the experimental setup to evaluate our algorithm in the domain of *two operand addition up to a maximum of four digits* with simulated students. The curriculum graph generation for the domain, the models for simulated students and hyper-parameter tuning are described below. While the curriculum graph is used by our algorithm, the simulated students are used only to evaluate our algorithm, and not by the algorithm itself.

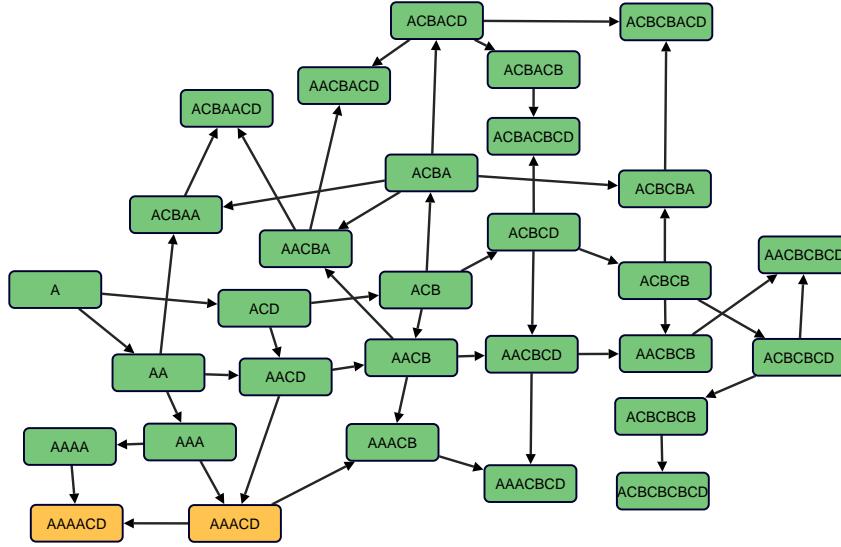


Figure 4.4: Curriculum graph generated for 2 operand addition up to a maximum of 4 digits.

4.3.1 Generating the Curriculum Graph

The trace based framework proposed by Andersen et al. [38] was used to generate a curriculum graph for the domain of two operand addition up to a maximum of four digits. Given a procedure for solving an addition problem, the actions taken to solve the problem can be labelled to build an execution trace of the solution. Towards this, four basic operations were identified to be required to solve an integer addition problem: one digit addition without a carry (A), one-digit addition with a carry (B), writing a carry (C), and bringing down a final carry (D). For instance, the traces associated with various problems are shown in Table 4.1:

These traces can then be ordered by complexity based on the N-gram-based partial ordering as follows: *Let n be any positive integer. A trace T_1 is said to be at least as complex as trace T_2 if every n -gram of trace T_2 is also present in trace T_1 .* To account for the detail

Table 4.1: Traces generated for addition problems.

Problem	$2+3$	$15+18$	$93+15$	$298+865$
Trace	A	ACB	AACD	ACBCBCD

Table 4.2: Knowledge Components defined for the student models.

Sl	Knowledge Component	Pre-requisite
1	One digit addition without overflow	
2	One digit addition with overflow	1
3	Two digit addition without carry and no overflow	1
4	Two digit addition without carry and overflow	2,3
5	Two digit addition with carry and no overflow	2
6	Two digit addition with carry and overflow	5
7	Three digit addition without carry and no overflow	3
8	Three digit addition without carry and overflow	4,7
9	Three digit addition with carry and no overflow	5
10	Three digit addition with carry and overflow	9
11	Four digit addition without carry and no overflow	7
12	Four digit addition without carry and overflow	8,11
13	Four digit addition with carry and no overflow	9
14	Four digit addition with carry and overflow	13

that ACB (two digit addition with carry) is a more complex problem-type as compared to AA (two digit addition without carry) and ACD (one digit addition with overflow and no carry), the partial ordering was updated to include the following: *A trace T1 is said to be at least as complex as trace T2 if T1 and T2 only differed at the last character being a B and D respectively.* The curriculum graph generated with n=3 is given in Figure 4.4.

4.3.2 Student models for simulation

Students were simulated using three popular knowledge tracing models, as described below. For simulations, knowledge components were defined corresponding to addition of different lengths, presence of carry and overflow, that were used by each of the student models. Prerequisites were also imposed on knowledge components, as shown in Table 4.2. Each problem was associated with exactly one knowledge component and all the problems were assumed to be of the same difficulty level. See Appendix A.1 for the values of parameters used in our experiment for different student models.

Bayesian Knowledge Tracing (BKT). Here a student's knowledge state is modelled as a two-state Hidden Markov Model (HMM) [28], one per knowledge component/skill, where the skill is either mastered by the student or not. The model transitions from the not-learned to learned state with probability p_T , and once mastered a knowledge component cannot be forgotten. In the not-learned state, a student can guess the solution of a problem correctly with probability $p_g < 0.5$. In the learned state, a student can slip an answer incorrectly with probability $p_s < 0.5$.

Similar to Mu et al. [24], pre-requisites between knowledge components were enforced by varying the transition probability between states p_T , depending on whether the prerequisites were learned or not. For example p_T for the knowledge component of two-digit addition with carry is $p_{T_{high}}$ if the prerequisite of one-digit addition with carry is learned, and is $p_{T_{low}}$ if the prerequisite is not learned, with $p_{T_{high}} > p_{T_{low}}$. To sample the student's response to a given problem, the overall probability of the student answering correctly is computed by taking the product of the probability of correctness of the student on each knowledge component relevant to the problem, along with their prerequisites.

Learning Factors Analysis (LFA). In LFA [40], the probability p of a student performing correctly on an activity, based on knowledge component kc is modeled by the following logistic function.

$$\ln\left(\frac{p}{1-p}\right) = \frac{\alpha}{|S_{kc}|} \sum_{i \in S_{kc}} n_i \beta_i - \gamma \quad (4.8)$$

where n_i is the number of times knowledge component i has been practiced, α is the student learning parameter, β_i is the learning rate of knowledge component i , γ is the difficulty of kc and S_{kc} is the set containing kc along with its pre-requisite knowledge components.

Performance Factor Analysis (PFA). PFA [37] is another popular model of student learning that uses the counts of correct and incorrect responses to track past practice. Here, the probability p of a student performing correctly on an activity based on knowledge component kc is modeled by the following logistic function:

$$\ln\left(\frac{p}{1-p}\right) = \frac{\alpha}{|S_{kc}|} \sum_{i \in S_{kc}} (s_i \beta_i + f_i \eta_i) - \gamma \quad (4.9)$$

where s_i and f_i are the number of times knowledge component i has been used correctly and incorrectly respectively, α is the student learning parameter, γ is the difficulty of kc , β_i and η_i are respectively the coefficients for the success count and

failure count of knowledge component i , and S_{kc} is the set containing kc along with its pre-requisite knowledge components.

4.3.3 Parameters

Hyper-parameters for ZPDES: The best value of hyper-parameters such as η , β , γ and w_0 were found using grid search separately for each student model.

Activity Selection in CurriculumTutor does not rely on any hyperparameters.

Change-point detection: To ensure a fair comparison between the sliding window and CUSUM approaches, the false alarm rate (FAR) was fixed at $5e - 05$. The corresponding accuracy threshold for the sliding window method was then determined with a window size $d = 8$, and the CUSUM threshold parameter h was selected such that both methods exhibited approximately the same (FAR). To find the FAR of sliding window based method, a markov chain was constructed with the states corresponding to the number of ones to be obtained to reach the accuracy threshold t . With zero set as the absorbing state, the mean time to absorption for the chain was computed, which corresponds to $E_0(\tau)$. Using this technique, for sliding window with $d = 8$ the accuracy threshold was set as 0.7 for the chosen FAR. Thus, out of the last 8 attempted activities, a student has to get at least 6 of them correct in order to infer that a concept has been mastered.

Similarly for CUSUM, a markov chain was constructed with the states corresponding to the possible values taken by the statistic \tilde{U}_n and states with values greater than h were designated as the absorbing states. Again, based on the mean time to absorption for this chain, for the chosen FAR, the threshold h for CUSUM was set as $\ln(1/0.0004)$. Since the guess and slip probabilities for each problem were not available to the algorithm, they were sampled from a Beta distribution, $Beta(20, 40)$, which serves as a prior for the Bernoulli distribution.

4.4 RESULTS

In this section we present the experimental results that compare the performance of three tutoring algorithms namely *CurriculumTutor*, *ZPDES* and *Blocking* (a basic scheduling scheme where activities associated with the same concept are presented to a student until the concept is mastered, uses a sliding window to infer mastery).

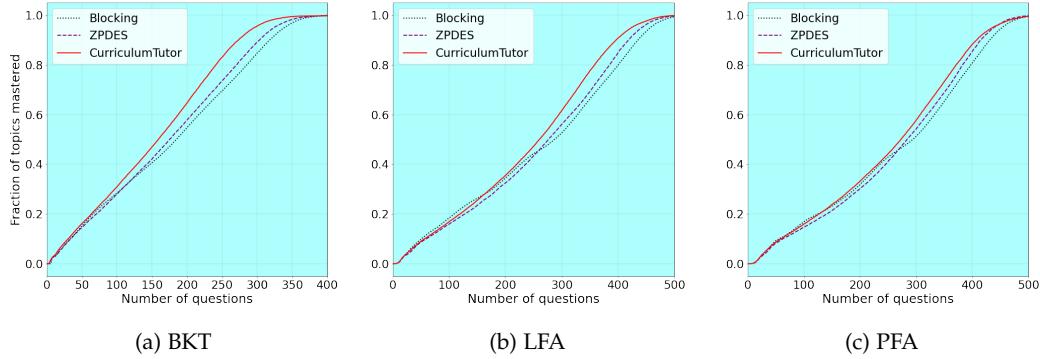


Figure 4.5: Simulation results for Scenario 1 - Student not struggling with any of the concepts.

For these three algorithms, the following two scenarios were simulated using the three different student models:

SCENARIO 1 *Student not struggling with any of the concepts* Here similar model parameters (transition probabilities, learning coefficients, intercept values, etc.) were used for knowledge components (KC) of similar complexity and the parameter values were made smaller with increasing complexity. For instance, the KC parameters associated with two digit addition were lower as compared to those associated with one-digit addition.

SCENARIO 2 *Student struggling with a few concepts in the intermediate phase* This scenario was simulated by using a lower transition probability as well as learning parameters for the student model for one of the KCs - *three digit addition without carry and overflow*, as compared to Scenario 1. This causes the students to struggle on the concepts *AAACD* and *AAAACD* in Figure 4.4.

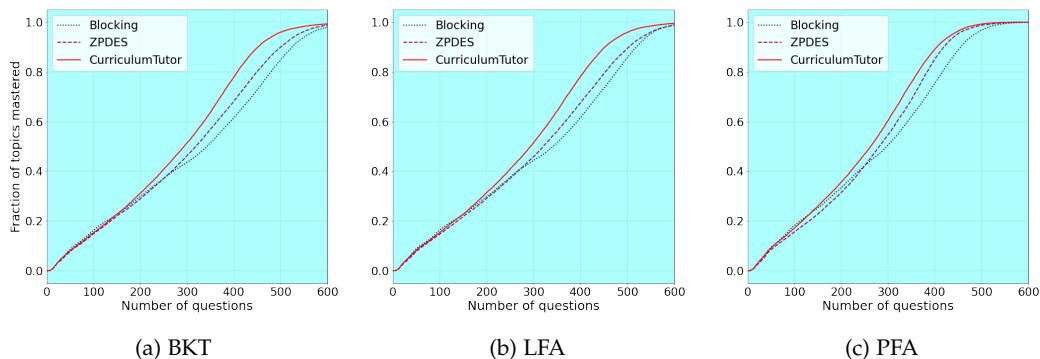


Figure 4.6: Simulation results for Scenario 2 - Student struggling with a few concepts in the intermediate phase.

Learning gains were measured by computing the fraction of concepts in the curriculum graph that was mastered by the student over a given number of questions. The average curves for 500 runs were plotted for each student model, as seen in Figures 4.5 and 4.6. It could be observed that for all the student models in both the scenarios, the learning curves of *CurriculumTutor* lies above those of ZPDES and Blocking, implying higher learning gains. In scenario 2, it could be observed that the learning curves of both the bandit algorithms, ZPDES and CurriculumTutor, are quite higher than that of Blocking (Figure 4.6). This is because while Blocking continuously tries to present activities related to a concept that the student is struggling with, the bandit algorithms performs exploration to find another concept that gives better rewards and thus contributes to higher learning progress. Thus CurriculumTutor performs better than ZPDES and Blocking in both the scenarios, irrespective of the student models chosen.

4.5 DISCUSSION

In this section, we provide a comprehensive discussion on several key aspects of our algorithm, including its complexity, limitations, validation with real students, and the necessary modifications when relaxing certain assumptions in the problem setting.

Computational complexity: CurriculumTutor (Algorithm 4.3) at each time step invokes SELECT-ACTIVITY (Algorithm 4.1) which has a complexity of $O(|ZPD|)$ where $|ZPD|$ is the number of concepts in the ZPD. DETECT-MASTERY (Algorithm 4.2) has $O(1)$ complexity as it involves updating and comparing the maximum LLR statistic of the concept chosen by SELECT-ACTIVITY. Therefore, the computational complexity of CurriculumTutor at each time instant is $O(|ZPD|)$.

Limitations of the approach: CurriculumTutor performs sequencing based on a curriculum graph that models the pre-requisite relations between the concepts to be learned. When the curriculum graph has a chain-like structure, the algorithm's performance gains over *Blocking* is constrained due to the limited scope for exploration among concepts. Also, a concept once mastered is assumed to be never forgotten. However it is known that if a previously learnt skill is not used for a while, there is a higher chance of forgetting it. Therefore, the adaptive sequencing algorithm has to periodically revisit the concepts that have been already mastered by a student in order to ensure that the curriculum has been learned efficiently.

4.5.1 Validation with real students

CurriculumTutor has not yet been validated with real students. Real world validation would involve deploying the algorithm on educational technology platforms that support student learning of mathematical questions through practice with a variety of question types—such as multiple-choice and numerical. The curriculum to be mastered may consist of a set of 6–8 interrelated concepts or skills drawn from domains such as introductory fractions, basic statistics, or geometry, which are appropriate for learners aged 8 and above. An expert-defined curriculum graph can be used for sequencing the concepts.

The performance of CurriculumTutor can be compared to that of ZPDES and an *expert-defined sequence*. Around 400 - 500 participants can be used for the evaluation study. A pre-test can be administered initially to assess the students' baseline knowledge, after which students are assigned to the three tutoring conditions in such a way that the average pre-test scores across the conditions are equivalent. Approximately 40% of the students are assigned to each of CurriculumTutor and ZPDES, with the remaining 20% assigned to the *expert-defined sequence* condition. The participants engage with their assigned tutoring algorithm for a fixed duration, such as 1 hour, after which a post-test is administered to assess their knowledge state upon completing the tutoring session. An exit survey can also be administered to gather participants' feedback on various aspects, including their overall learning experience, the effectiveness of the activity sequencing, and their level of engagement with the tutoring process.

Learning gain is computed as the difference in a learner's score between the post-test and pre-test. The performance of the three tutoring algorithms can be compared by assessing the average learning gain achieved by the participants under each condition. Furthermore, learner traces collected throughout the session can be used to analyze and compare various factors, such as the number of problems completed by the learner, the types of problems tackled, and the correctness of their responses. The participant feedback can also be analyzed to evaluate and compare the effectiveness of the three tutoring algorithms.

4.5.2 Relaxing the assumptions on questions related to a concept

The problem setting outlined in Section 3.2 is based on the following assumptions:

- Each activity is associated with exactly one concept, and

- All activities are of the same difficulty level.

In this section, we discuss how the proposed framework can be adapted to accommodate scenarios in which these assumptions are relaxed or no longer hold. It is well-established that question difficulty tends to increase with the number of underlying concepts required to solve it. [47–49]. Therefore, we consider questions associated with multiple concepts as a specific instance of increased question difficulty. Accordingly, we analyze the following two scenarios.

Increased difficulty of questions based on a single concept

The difficulty of questions associated with a concept can vary due to multiple factors. Educational taxonomies such as Bloom’s taxonomy [27] divides cognitive objectives into levels (knowledge, comprehension, application, analysis, synthesis, evaluation) that are sorted from easier to more complex. Questions targeting higher cognitive levels in such taxonomies might be more difficult than those addressing lower cognitive levels [49, 50]. The difficulty of a question can also be amplified by increasing the complexity of its presentation [50]. This may involve strategies such as altering the type of vocabulary used, introducing irrelevant information to distract students, or, conversely, providing insufficient information that necessitates multiple deductions.

To account for questions of varying difficulty levels associated with the same concept, we propose the following technique. Let **I** denote a concept node in the curriculum graph, with both incoming and outgoing prerequisite relationships. Suppose that the questions linked to concept **I** can be classified into three levels of difficulty: *low*, *medium*, and *high*. To capture this variation, the concept node **I** can be partitioned into three distinct difficulty-augmented nodes, each representing questions of a specific difficulty level associated with concept **I**, as shown in Figure 4.7.

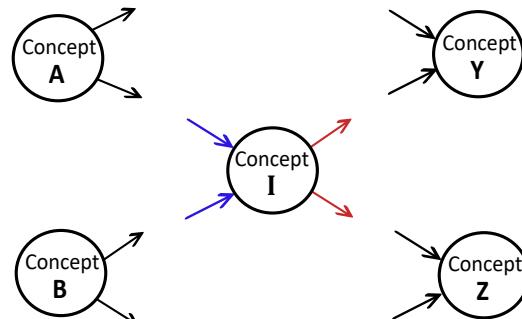
Since students typically progress sequentially from lower to higher difficulty questions, the prerequisite relationships among the difficulty-augmented nodes are defined accordingly. Specifically, a node with lower difficulty serves as a prerequisite for the node with the next highest difficulty in the hierarchy. For example, in Figure 4.7, the *low-difficulty* node serves as a prerequisite for the *medium-difficulty* node, which in turn acts as a prerequisite for the *high-difficulty* node.

The incoming prerequisite relations for concept node **I** are directed to the difficulty-augmented node representing the lowest difficulty level. The outgoing prerequisite relations from **I** can be assigned to any of the difficulty-augmented nodes, as illustrated

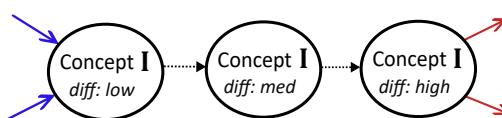
in Figure 4.7b and Figure 4.7c, depending on the type of difficulty level a student must master before advancing to a higher concept. For example, in Figure 4.7b, the student progresses to higher concepts only after mastering high-difficulty level problems associated with concept I, whereas in Figure 4.7c, the student can advance to higher concepts upon mastering low-difficulty problems.

Increased difficulty of questions based on multiple concepts

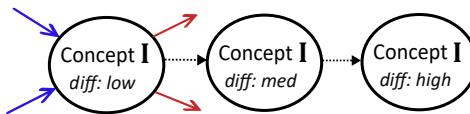
An approach for handling questions associated with multiple concepts is to create a new concept node in the curriculum that represents the combination of the involved concepts. For example, for questions that integrate concepts P, Q, and R, a *combination node* PQR can be created, with P, Q, and R as its prerequisites, as shown in Figure 4.8. This approach is effective when there are sufficient practice questions that involve all the concepts in the combination, indicating the significance of this concept combination. Additionally, the combination node C'_i can serve as a prerequisite for another combina-



(a) Concept node I in the curriculum graph



(b) Descendants of Concept I are added to the ZPD only upon mastery of high-difficulty problems associated with Concept I



(c) Descendants of Concept I are added to the ZPD upon the mastery of low-difficulty problems associated with Concept I

Figure 4.7: Decomposing a concept node into multiple difficulty based sub-nodes. Here the values *low*, *med* and *high* for *diff* represent low, medium and high difficulty problems associated with the concept.

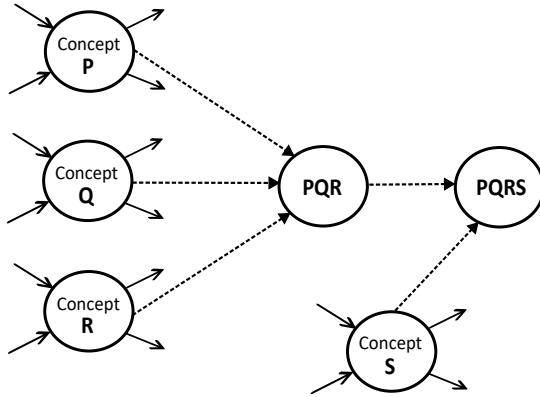


Figure 4.8: Introducing *concept-combination* nodes for handling questions involving multiple concepts.

tion node C'_j , provided there is no existing combination node c in the curriculum such that C'_i is a prerequisite for c and c is a prerequisite for C'_j .

4.6 CONCLUSION

In this part of the thesis, we studied the problem of generating adaptive sequences of activities to master a curriculum in the learning by doing setting, wherein the students learn a concept by performing activities related to the concept. Based on a curriculum graph that depicts the inter-dependencies between the various concepts to be learned, zone of proximal development (ZPD) represents the concepts that a student can master at each time instant. We have shown that the adaptive sequencing problem in the learning by doing setting can be modelled as a piecewise stationary multi-armed bandit problem on the ZPD. Based on this framework, the sequencing problem can be decomposed into two key components: (i) selecting the concept based on which an activity is presented next, which is performed by a bandit algorithm ,and (ii) determining if a concept has been mastered based on the correctness of student response to the activity, formulated as a statistical change point detection problem. Our main contribution is a novel tutoring algorithm called *CurriculumTutor*, that combines an upper confidence bound bandit algorithm with a log-likelihood based statistical test for change-point detection, to master concepts in a curriculum through adaptive activity sequencing. The change point detector separately monitors the mastery level of each concept while the bandit algorithm manages the exploration vs exploitation tradeoff for the concepts in the ZPD. Using simulated students we show that CurriculumTutor significantly out-performs prior approaches in the benchmark domain of two operand addition up to a maximum of four digits. We also explore how the proposed algorithm

can be validated with real students and discuss approaches for handling scenarios where assumptions regarding question difficulty and the association of a question with a single concept are relaxed.

Part II

PERSONALIZED OPTIMAL CONTENT SEQUENCING

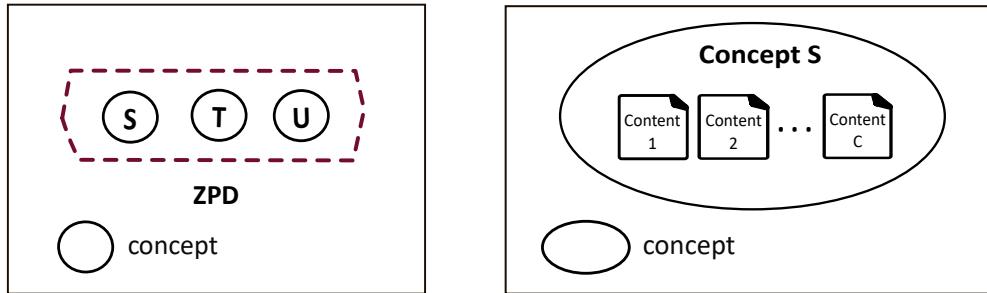
5

PERSONALIZED OPTIMAL SEQUENCING OF CONTENT TO MASTER A CONCEPT

The *Zone of Proximal Development* (ZPD) encompasses the concepts that the student can master at a given time. In the previous part of the thesis, we examined the problem of adaptive sequencing of concepts in the ZPD to master a curriculum in the learning by doing setting (LByD) setting where a concept is learned by performing an activity related to that concept (Figure 5.1a). Next we study the learning by passive engagement (LbyPE) setting where learning takes place through engagement with various educational content/resources such as videos, textbook sections and worked-out examples. When multiple learning resources are available to master a concept, the sequence in which these resources are presented becomes crucial. Here, we examine how to *optimally* sequence educational content for a concept in the ZPD, tailoring it to the specific learning needs and characteristics of each student (Figure 5.1b).

Conventionally, the educational content in learning systems are either presented in a (i) fixed order defined based on an expert's intuition, or (ii) an order chosen by the learners on their own (self-directed navigation) [51]. Fixed ordering of content lacks personalization, whereas in self-directed navigation, the students being non-experts often do not know what is optimal. Consequently, the conventional methods are unable to simultaneously accomplish optimality and personalization. Furthermore, with the increasing quantity of available content, it becomes challenging to determine the optimal sequence for presenting the material to facilitate efficient learning of a concept.

In this part of the thesis, we study the problem of computing a *personalized, optimal* sequence of learning resources/content to a student that maximizes the probability of mastery of a concept within a time horizon H . We model *personalized optimal content sequencing* as an episodic finite-horizon Markov Decision Process (MDP), where the probability of mastery at a given time depends on the student's knowledge level which is determined by the student's profile and the content presented in the past. Whether or not mastery was achieved is used as a binary reward. A key component of our model is a matrix which captures the effectiveness of presenting content j immediately after content i is presented. We also propose *OPT-CSEQ*, a novel bandit algorithm with



(a) **Concept sequencing:** adaptive sequencing over the concepts in the ZPD.
 (b) **Content sequencing:** adaptive sequencing over the content/learning resources available to learn a concept.

Figure 5.1: Concept sequencing vs. Content sequencing.

MDP based planning, for generating optimal personalized content sequences.

In this chapter, we first describe the problem motivation (Section 5.1) and then discuss related work on educational content sequencing (Section 5.2). The framework, mathematical model and assumptions used to formulate the personalized optimal content sequencing problem in this study is presented in Section 5.3.

5.1 MOTIVATION

Educational technology platforms provide students with a variety of learning resources/materials/content such as videos, tutorials, worked out examples, visualizations, games/tasks, etc to help the student master a given concept. Studies have shown that the order in which the content is presented significantly affects the student's learning gains [26], as illustrated in Figure 5.2 with two students : Student 1 and Student 2. It can also be observed from the figure that the learning gain from the same content sequence vary for different students.

Figure 5.3 illustrates how the optimal content sequences vary for different students based on their profiles. Here, the sequence *A-B-C* provides better learning gains for Student 1 than *B-A-C* and *A-B-D*, while for Student 2, *A-B-D* provides the highest learning gains. This might be because content *A* covers some prerequisite material for content *B* and hence presenting *A* before *B* helps both the students. However content *C* is better than content *D* for Student 1 (and vice versa for Student 2) because, say, content *C* provides visualizations which is more effective for Student 1 as opposed to content *D* which provides worked-out examples (and vice-versa for Student 2).

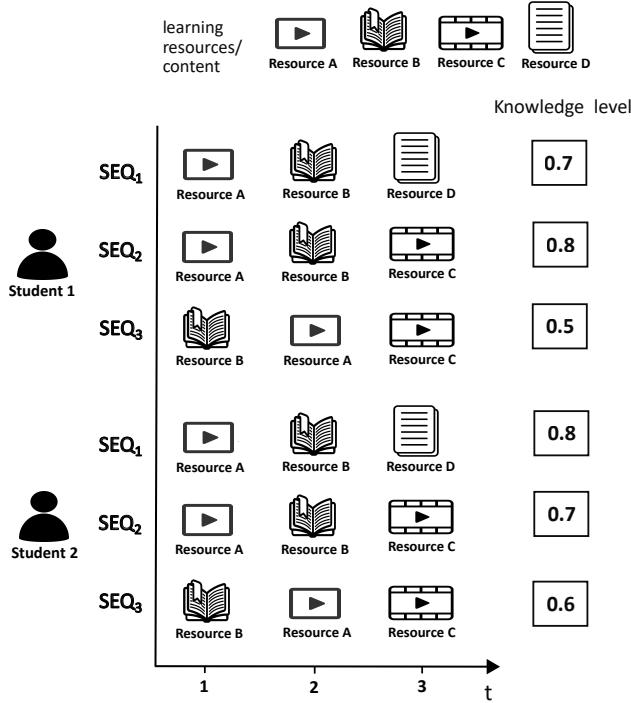


Figure 5.2: An illustration of how the order of presentation of content affects student's learning gains with two students: Student 1 and Student 2.

Traditionally, content sequencing is done using one of the following two approaches. In the first approach, an expert chooses a sequence of content that they deem optimal. This approach lacks personalization, and, further, given the diverse modality of the content, it is difficult even for an expert to manually arrange them in the right order. In the second approach, the content sequencing is student-driven, wherein, the students are allowed to navigate the platform and engage with the content of their choice. While this achieves personalization, students being non-experts often do not know what is optimal. Thus, the conventional approaches do not achieve personalization and

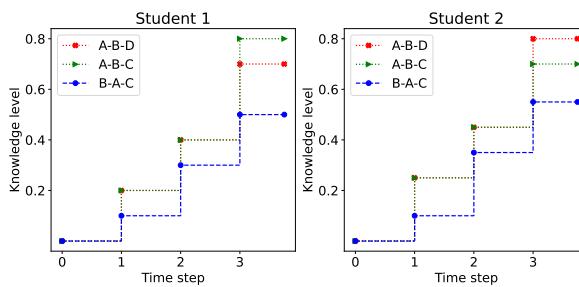


Figure 5.3: An example of how the optimal content sequences differ for various students based on their profiles, with two students: Student 1 and Student 2.

optimality simultaneously.

In this dissertation, we study the problem of *personalized optimal content sequencing*. Specifically, we address the problem of determining an optimal sequence of content that maximizes the probability that a student masters the concept within a given time horizon H . Here, the horizon stands for the number of times the student is engaged with a concept/topic. Say, each week, a new topic is covered and there are $H = 4$ to 5 sessions/engagements per week per topic. The following are the two key challenges involved in addressing this problem.

- **context based personalization:** the effectiveness of a content for a certain student would depend on the student's learning profile (i.e. context), such as the prior knowledge or other learning preferences/traits. Therefore, similar content sequences must be generated for students with similar profiles. This fits into the framework of a contextual bandit problem in which the student provides the context and the content form the arms.
- **temporal optimality:** the learning gain for a student due to a content at a given time depends on the past content that was presented to the student. In other words, the content presented controls the evolution of the student knowledge level (which is an internal state). Therefore this problem involves sequential decision making, as the choice of a content at a given time step has long-term effects down the horizon.

5.2 RELATED WORK

Educational content sequencing involves *recommending* content at each time point to maximize student learning gains, based on feedback from the student on previously presented material. In this section, we first discuss the related work on MDP-based recommender systems and then describe the prior work on content sequencing in education.

Recommender systems based on MDPs

Prior works have modeled the problem of generating recommendations as a sequential decision problem using MDPs [52–54]. Shani et al. [52] proposed a sequential decision based recommender system with an n-gram based model for generating the MDP. The states correspond to the sequences of the most recent k selections made by a

user, and actions represent the items recommended to the user. The rewards are the profits obtained by selling items and an optimal solution to the MDP is a policy that maximizes profits. Here the recommendations are not personalized as the user context is not considered while generating the policy. Moreover, there is no uncertainty in the transition probabilities, as the values are empirically computed from the user data.

Warlop et al. [53] used an MDP to model the dynamics of user preferences caused by recommendations in the context of a movie recommender system. Here the rewards are modeled as a linear function of the state which captures the recent history of actions (genres of movies watched). A novel learning algorithm based on contextual bandits has been introduced to learn an optimal policy within this framework. Basu et al. [54] proposed a simple and effective approach motivated by the pharmacology literature, to convert the non-Markovian prolonged action effects of medications in precision drug dosing problem into a Markov decision process. The revised MDP is then solved using model-free reinforcement learning algorithms such as deep Q-networks (DQN).

Sequencing educational content: Related studies

Next, we discuss prior sequencing techniques for educational content, developed for settings comparable to the one examined in this thesis.

Tekin et al. [55] introduced ETutor, a method for sequencing teaching materials aimed at maximizing student performance in the final exam while minimizing the time students spend learning the course content. The materials presented to a student are selected dynamically based on the student's context, prior materials viewed, responses to previously shown questions (correct or incorrect), and the aggregated knowledge from past students with similar contexts, responses, and scores. ETutor uses mean reward estimates obtained from past student contexts and other features similar to the current student to determine the effectiveness of presenting a content. Our approach, on the contrary, uses a generalized linear bandit framework to model the efficacy of the content and MDP planning to obtain the optimal content sequence.

The problem of selecting the optimal personalized learning action (PLA) in order to maximize a student's performance on the follow-up assessment has been studied [56, 57]. Here a student's learning schedule consists of a series of assessments with PLAs embedded in between. Each PLA corresponds to engaging with a learning resource, such as reading a textbook section or watching a lecture video. The goal here is to

choose the optimal PLA for each student based on their learning history (graded responses to previous assessments) in order to maximize their score on the subsequent assessment. In these works, PLA selection was posed as a contextual multi-armed bandits (MAB) problem with each available PLA as an arm, the student's estimated knowledge level as the context and the students' responses to questions in the follow-up assessment as rewards. Here a reliable estimate of each student's latent concept knowledge vector is assumed to be available to the PLA selection algorithm. Our approach differs from this setting in two ways: (i) we perform MDP planning for a horizon of H , and (ii) our context incorporates both the student feature vector and information about previously viewed content.

Other works on sequencing educational content

Bassen et al. [51] proposed Reinforcement Scheduling (RS), an approach to assign course materials, and empirically demonstrated that it achieves better learning gains. However, the algorithmic and analytical contributions are limited in that proximal policy optimisation (PPO) is applied in a standard mode without any sample complexity analysis. A novel video recommendation system for algebra was proposed by Leite et al. [58] where new video recommendations were made by a reinforcement learning algorithm based on the ability estimate of the student from an item response theory model, important weights for videos, student scores on a quiz and an engagement detection model. Chakraborty et al. [59] designed an algorithm to recommend tutoring videos on algebra to prepare students for an End-Of-Course algebra test, where the strategy is to recommend a video based on the topic with the largest predicted growth in the score. These works are not based on reinforcement learning, in that, their recommendations are mostly based on a predictive model.

Iglesias et al. [60] used Q-learning to maximize the number of skills/concepts mastered by a student. Tang et al. [61] used Q-learning for the scenario in which a student has k skills to learn within T time epochs. The main goal in these works is to improve various skill levels, in contrast to our goal of maximizing the mastery of a single concept.

Our work differs from the existing methods on the following points: (i) we consider the problem of learning a single concept using multiple learning resources (ii) the time horizon is fixed, and (iii) the student context is also considered while determining the *optimal* sequence. Table 5.1 summarizes the In the next section, we describe our problem setting in detail.

Table 5.1: Comparison of the settings studied by prior works on educational content sequencing

	No: of concepts	Fixed horizon	Student context	Feedback
ETutor[55]	> 1	✗	✓	final exam scores, student feedback on content presented
PLA selection algorithms [56, 57]	k	✗	✗	scores in followup assessment
Reinforcement Scheduling [51]	> 1	✗	✗	difference between pre and post-test scores
Tang et al. [61]	k	✓	✗	mastery of concepts
RLATES [60]	k	✗	✓	mastery of concepts
Chakraborty et al. [59]	k	✗	✗	scores in follow-up mini assessment
OPT-CSEQ	1	✓	✓	mastery of concept

5.3 PROBLEM SETTING

The overall framework comprising the educational platform, the student and our algorithm *OPT-CSEQ* is shown in Figure 5.4.

Platform and Content. Our setup is motivated by the typical scenario wherein each course has weekly topics and for each topic, there is a finite set of learning resources/content denoted by $\mathcal{C} = \{C^1, C^2, \dots, C^k\}$. In a given week, the student engages with the platform at times $t = 1, \dots, H$ ¹.

Student. The student is modeled by a context vector $x^{\text{feat}} \in \mathbb{R}^m$. This context vector could represent the student's prior knowledge level such as performance in the courses attended in the past, amount of engagement in the platform and the type of content frequently engaged with, etc.

Engagement and skill. At time t , the student is provided with content $c_t \in \mathcal{C}$. Due to c_t , the student's skill level changes from q_{t-1} to q_t . The skill level is an internal state and is not directly available to the algorithm *OPT-CSEQ*.

¹ $H = 3, 4$ or 5 are the typical number of sessions per week per topic per course.

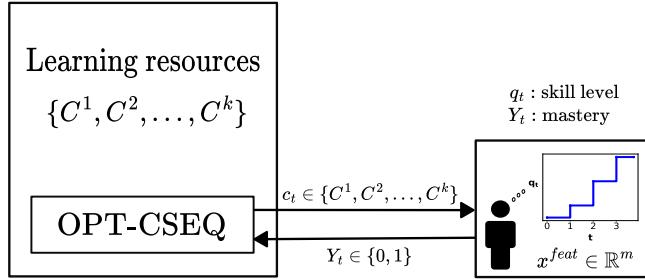


Figure 5.4: The overall framework for personalized optimal content sequencing.

Mastery. The student attains mastery $\in \{0, 1\}$ at time t depending on q_t (see the next subsection). The probability of attaining mastery monotonically increases with the skill level. If the student attains mastery at time $t = H' < H$, the student will not continue to engage with the content related to the topic from $t = H' + 1, \dots, H$.

Feedback. After engaging with content c_t , we assume that the student's mastery $\in \{0, 1\}$ is measured by an assessment test and is presented as a feedback to our algorithm *OPT-CSEQ*. This feedback is denoted by $Y_t \in \{0, 1\}$.

5.3.1 Student as Context and Effectiveness of Content

In this thesis, we assume an additive learning model in lines with the constructivism learning theory [62] which is established on the premise that students learn by fitting new information together with what they already know. Thus, the effectiveness of an educational content at any given time is presumed to be dependent on the order of content presentation — as previously discussed by Katona [63]. As students tend to *forget* what they have learned over time, accounting for this effect is crucial when modeling skill progression. Besides, as illustrated in Figure 5.3, the learning gains from the same content sequence can vary significantly across different students. Therefore the effectiveness of a content also depends on the student profile. We incorporate all these factors into our skill dynamics model which dictates the evolution of the student skill level q_t as they engage with the content c_t , as detailed below.

Without loss of generality, we expand the content set to $\mathcal{C} = \{\text{NULL}, C^1, \dots, C^k\}$ with the understanding that at time $t = 1$, the previous content shown is $c_0 = \text{NULL}$ and $c_t \neq \text{NULL } \forall t = 1, \dots, H$. Let s_0 be a given student, and $x_0^{feat} \in \mathbb{R}^m$ be the associated context vector. Then

$$\Omega_0(x_0^{feat}; i, j) = \langle x_0^{feat}, \theta^*(i, j, .) \rangle \quad (5.1)$$

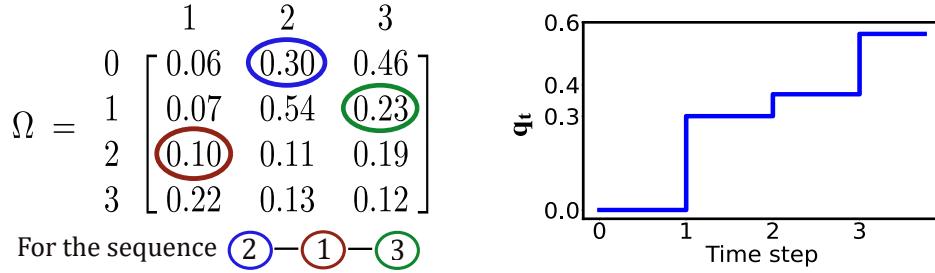


Figure 5.5: The skill levels q_t for the given sequence of content computed using (5.3) and (5.4) based on the parameter Ω_0 (5.1) and $\kappa = 0.9$

is the corresponding effectiveness matrix where $\Omega_0(x^{\text{feat}}; i, j)$ is the effect of presenting content j after content i to student s_0 with context/feature x^{feat} . Ω_0 is expressed as:

$$\Omega_0 = \begin{bmatrix} \langle x_0^{\text{feat}}, \theta^*(1, 1, .) \rangle & \dots & \langle x_0^{\text{feat}}, \theta^*(1, k, .) \rangle \\ \vdots & \ddots & \vdots \\ \langle x_0^{\text{feat}}, \theta^*(k+1, 1, .) \rangle & \dots & \langle x_0^{\text{feat}}, \theta^*(k+1, k, .) \rangle \end{bmatrix}$$

Here $\theta^*(i, j, .) \in \mathbb{R}^m$ is a hidden parameter vector. We use $\theta^* = \{\theta^*(i, j, .), i = 0, 1, \dots, k; j = 1, \dots, k\} \in \mathbb{R}^{(k+1) \times k \times m}$ to denote the overall hidden parameter tensor.

As shown in Figure 5.5, the skill level q_t is determined by the sequence of content presented and Ω_0 . For illustration purposes, here we have directly used an Ω_0 , omitting the specifics of x_0^{feat} and θ^* .

Effect of forgetting. We model the forgetting/slip of skill level by a factor $\kappa \in (0, 1)$. In Figure 5.5, the value of κ used is 0.9

Recency of usage of specific learning resource. We use the recency function proposed in [53] where the effect induced by a content c fades as $\frac{1}{\tau}$ so that the recency is large if a content is often selected and it decreases when it is not selected for a while.

$$\rho_c(t) = \sum_{\tau=1}^t \mathbb{I}(c_{t-\tau} = c) / \tau \quad (5.2)$$

5.3.2 Skill and Mastery

Skill dynamics. Thus the skill level q_t on engaging with content c at time t is given by:

$$q_t(c) = \kappa q_{t-1} + \Omega(x^{\text{feat}}; c_{t-1}, c) \gamma^{-\rho_c(t)} \quad (5.3)$$

where γ is a positive constant. The skill level q_t is given by:

$$q_t = q_t(c_t) \quad (5.4)$$

When engaged with content c at time t , the probability of mastering the concept is given by.

$$p_{\text{mastery}}(q_t(c)) = \sigma(q_t(c)) = \frac{1}{1 + e^{-(q_t(c) - D)}} \quad (5.5)$$

where D represents the difficulty of the concept. Thus $Y_t \sim \text{Bernoulli}(p_{\text{mastery}}(q_t))$. As shown in Figure 5.5, the probability of mastery evolves over time based on q_t which is determined by the sequence of content presented and Ω .

5.4 SUMMARY

Studies have shown that the sequence in which educational content is presented to students has a substantial impact on their learning. In this chapter, we presented the problem of computing a personalized sequence of learning resources/content for a student that maximizes the probability of mastery of a concept within a time horizon H . We assume that on engaging with a content, the evolution of student's skill depends on (i) the student's learning profile and (ii) the content presented in the past. Mastery is examined at each time step using an assessment, which forms a binary feedback. Our model captures *forgetting* of the student, *recency* of the content, the *effectiveness of presenting content j immediately after content i*.

6

OPT-CSEQ: PERSONALIZED OPTIMAL SEQUENCING OF EDUCATIONAL CONTENT

In the previous chapter, we introduced the problem of *personalized optimal content sequencing* which involves determining a tailored sequence of learning resources for a student to maximize the probability of mastering a concept within a specified time horizon H . We also outlined the framework, mathematical model, and assumptions employed to define the personalized optimal content sequencing problem.

In this chapter, we first describe our approach to formulate the *personalized optimal content sequencing* problem as an *episodic fixed horizon Markov Decision Process (MDP)* with a generalised linear bandit model representing the probability of mastery at each time step (Section 6.1). We then introduce *OPT-CSEQ*, a novel online planning algorithm for content sequencing, that is based on upper confidence bounds (Section 6.2). *OPT-CSEQ* learns optimal personalized policies on sequencing content, over episodes of interactions. Next we show that *OPT-CSEQ* achieves an $\tilde{O}(d\sqrt{TH})$ regret bound over T episodes, where H is the horizon and $d = k(k+1)m$ (Section 6.3). Through numerical experiments, we demonstrate the ability of *OPT-CSEQ* to generate optimal personalized content sequences for students with different profiles and compare its performance against techniques based on (i) random and (ii) greedy policies (Section 6.4). In Section 6.5, we discuss future work on validating *OPT-CSEQ* in real educational settings.

6.1 PROPOSED MODEL

This section outlines our approach to formulating the *personalized optimal content sequencing* problem using an MDP. Here a *generalized linear bandit* framework is used to model the progression of skill and the achievement of mastery, presented in the previous chapter.

6.1.1 Generalised Linear Bandit

We first define *trace* $x_t \in \mathbb{R}^{(k+1) \times k \times m}$ which keeps track of the history of the content presented till t . The set of k arms at t is given by $\{x_{t,c}, c \in \mathcal{C}\} \in \mathbb{R}^{(k+1) \times k \times m}$ (recall from (5.1) that the underlying parameter is $\theta^* \in \mathbb{R}^{(k+1) \times k \times m}$). The relationship between $x_t, x_{t,c}, \theta^*$ is given in the set of equations below. (6.1) states that the trace is 0 at $t = 0$. (6.2) models the forgetting in one time step, and (6.3) states that the arm corresponding to action c is obtained by adding the user feature x^{feat} to the trace after scaling it with the recency coefficient $\gamma^{-\rho_c(t)}$. (6.4) states the computation of skill level and (6.5) that the probability of mastery is a generalised linear bandit in the arms $\{x_{t,c}, c \in \mathcal{C}\}$. Finally, (6.6) states that once an arm c_t is chosen the trace is equal to the arm itself.

$$x_0 = \mathbf{0} \quad (6.1)$$

$$x_{t,c} = \kappa x_{t-1} \quad (6.2)$$

$$x_{t,c}(c_t, c, .) = x_t(c_t, c, .) + \gamma^{-\rho_c(t)} x^{\text{feat}} \quad (6.3)$$

$$q_t(c) = \langle x_{t,c}, \theta^* \rangle \quad (6.4)$$

$$p_{\text{mastery}}(x_{t,c}) = \sigma(\langle x_{t,c}, \theta^* \rangle) = \sigma(q_t(c)) \quad (6.5)$$

$$x_t = x_{t,c_t} \quad (6.6)$$

Illustrative example

Consider a toy example with $k = 3$ contents and a two-dimensional feature vector for student context ($m = 2$). Let $\kappa = 0.9$. We have

$$x^{\text{feat}} = \begin{bmatrix} 0.1 & 0.5 \end{bmatrix}$$

$$\theta^* = \begin{bmatrix} [0.1 & 0.1] & [0.5 & 0.5] & [0.1 & 0.9] \\ [0.2 & 0.1] & [0.9 & 0.9] & [0.3 & 0.4] \\ [0.5 & 0.1] & [0.1 & 0.2] & [0.9 & 0.2] \\ [0.7 & 0.3] & [0.3 & 0.2] & [0.2 & 0.2] \end{bmatrix}$$

$$x_0 = \begin{bmatrix} [0 & 0] & [0 & 0] & [0 & 0] \\ [0 & 0] & [0 & 0] & [0 & 0] \\ [0 & 0] & [0 & 0] & [0 & 0] \\ [0 & 0] & [0 & 0] & [0 & 0] \end{bmatrix}$$

Let $c_1 = C^2$

$$x_1 = \begin{bmatrix} [0 & 0] & [0.1 & 0.5] & [0 & 0] \\ [0 & 0] & [0 & 0] & [0 & 0] \\ [0 & 0] & [0 & 0] & [0 & 0] \\ [0 & 0] & [0 & 0] & [0 & 0] \end{bmatrix}$$

$$q_1(c_1) = \langle x_1, \theta^* \rangle = 0.1 \times 0.5 + 0.5 \times 0.5 = 0.3$$

Let $c_2 = C^1$

$$x_2 = \begin{bmatrix} [0 & 0] & [0.09 & 0.45] & [0 & 0] \\ [0 & 0] & [0 & 0] & [0 & 0] \\ [0.1 & 0.5] & [0 & 0] & [0 & 0] \\ [0 & 0] & [0 & 0] & [0 & 0] \end{bmatrix}$$

$$q_2(c_2) = \langle x_2, \theta^* \rangle = 0.09 \times 0.5 + 0.45 \times 0.5 + 0.1 \times 0.5 + 0.5 \times 0.1 = 0.37$$

Let $c_3 = C^3$

$$x_3 = \begin{bmatrix} [0 & 0] & [0.081 & 0.405] & [0 & 0] \\ [0 & 0] & [0 & 0] & [0.1 & 0.5] \\ [0.09 & 0.45] & [0 & 0] & [0 & 0] \\ [0 & 0] & [0 & 0] & [0 & 0] \end{bmatrix}$$

$$\begin{aligned} q_3(c_3) = \langle x_3, \theta^* \rangle &= 0.081 \times 0.5 + 0.405 \times 0.5 + 0.09 \times 0.5 + 0.45 \times 0.1 \\ &+ 0.1 \times 0.3 + 0.5 \times 0.4 = 0.563 \end{aligned}$$

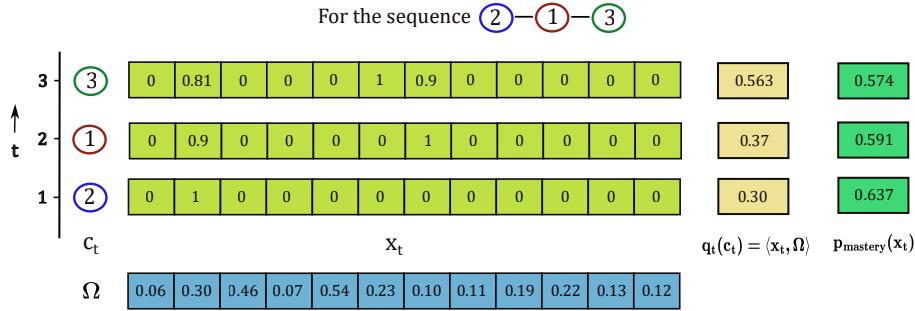


Figure 6.1: The probability of mastery evolves over time based on Ω and the sequence of content presented. Here Ω is computed based on the student profile and hence $q_t(c_t) = \langle \Omega, x_t \rangle$ and $p_{\text{mastery}}(x_t) = \sigma(q_t(c_t))$ with $D = 0$ and $\kappa = 0.9$ (see (5.5)). Since each resource is being used for the first time, the recency $\rho_{c_t} = 0$ for $t \in [3]$.

Figure 6.1 shows the evolution of the tensors x_t and computation of $q_t(c_t)$ and $p_{\text{mastery}}(x_t)$ for the given values of x^{feat} and θ^* . For illustration purposes, we directly computed $\Omega = \langle x^{\text{feat}}, \theta^* \rangle$ and structured x_t accordingly.

6.1.2 Markov Decision Process

We now define the MDP represented by the 5-tuple (S, A, P, R, H) .

State Space: $S = \tilde{S} \cup T$ where $\tilde{S} = \mathbb{R}^{(k+1) \times k \times m} \times [k]^H$ and T is a special terminal state which denotes mastery has been achieved. Here the state $s_t = (x_{t-1}, c_{1:t-1})$, where $x_t \in \mathbb{R}^{(k+1) \times k \times m}$ is a 3 dimensional tensor defined above and $c_{1:t-1}$ is the content sequence presented from 1 to $t-1$ ¹.

Action space: $A = C$ and $a_t = c_t$, the content presented at time t .

Transition probability: On choosing an action $c_t = c$ at state s_t , the student can master the concept and the MDP reach the terminal state T with probability

$$P(T|s_t, c) = \sigma(\langle x_{t,c}, \theta^* \rangle) \quad (6.7)$$

and with $1 - P(T|s_t, c)$ the state transitions to $s_{t+1} = (x_{t,c}; c_{1:t-1}, c)$.

Reward is 1 in the terminal state T and is 0 otherwise.

6.1.3 Tree Structure of MDP

The MDP has a tree structure, as shown in Figure 6.2. At each state, there are k possible actions, corresponding to the k learning resources C_1, \dots, C_k . On choosing the content c at state s , the terminal state T is reached with probability $p(T|s, c)$ and a non-terminal state s' is reached with probability $1 - p(T|s, c)$. The non-terminal states are further expanded in a similar manner until the tree height reaches H . Even though the state space grows exponentially with the time horizon, H in practice takes nominal values as the available time steps are usually limited ($H = 3, 4$ or 5) in the problem setting.

Given that the discount factor is set to 1 and only the terminal state T offers a non-zero reward, the optimal action at each state will be the content C^i that maximizes

¹ Since x_{t-1} is a deterministic function of $c_{1:t-1}$, $s_t = c_{1:t-1}$ can itself completely specify the state. However including x_{t-1} as part of s_t adds clarity.

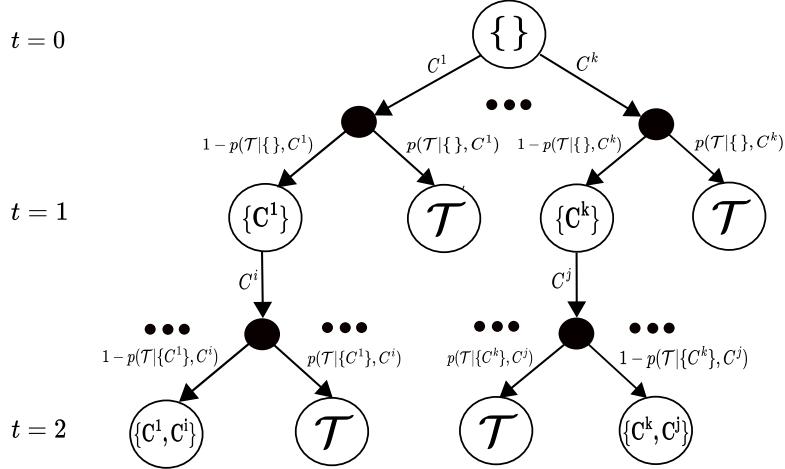


Figure 6.2: Generation of the state space tree for k contents.

the probability of reaching \mathcal{T} . Therefore the optimal policy from the root node of this MDP represents the sequence of content that maximizes the probability of mastery within horizon H . However since the underlying parameter θ^* is unknown, there is an associated uncertainty in the transition probabilities of the MDP, which must be taken into account while determining the optimal policy. In the next chapter, we discuss our proposed approach to learn an optimal policy that maximizes the probability of mastery under this setting.

6.2 OPT-CSEQ ALGORITHM

In this section, we introduce our algorithm, *OPT-CSEQ*, to generate an optimal sequence of content customized based on the student context, which maximizes the probability of mastering a concept within a horizon H . While algorithms such as UCRL [64] and Q-learning with UCB [65] can be applied to this setting, their general-purpose design limits their ability to exploit the deterministic MDP structure specific to our problem. Furthermore, the proposed setting incorporates several domain-specific characteristics — such as forgetting, the effects of recency or boredom, and binary feedback based on mastery or non-mastery — which our algorithm is specifically designed to leverage. Deep reinforcement learning algorithms are general-purpose, black-box methods that often fail to exploit problem-specific structure. In contrast, *OPT-CSEQ* provides greater interpretability, making it more suitable for educational applications where transparency is important.

OPT-CSEQ performs learning in the episodic finite horizon setting with binary rewards, where the parameters determining the transition probabilities are unknown. Similar to [66] and [53], *OPT-CSEQ* works in phases comprised of (i) determining the maximum likelihood estimate of θ (ii) optimistic planning and (iii) policy execution, that takes place over several episodes. *OPT-CSEQ* uses the principle of optimism in the face of uncertainty to determine the transition probabilities based on the student context and executes an optimistic policy until an update is performed after gathering sufficient samples.

Episodes and Vectorization: *OPT-CSEQ* is executed for $\tau = 1, \dots, T$ episodes and within each episode $t \in [H]$. At the beginning of episode τ , the trace $x_0^\tau = \mathbf{o}$ and within the episode, the evolution of the trace follows (6.1) to (6.6). Let $H_\tau \in [H]$ denote the time at which mastery is attained in episode τ . The reward obtained at time t in episode τ , r_t^τ , has the distribution:

$$r_t^\tau \sim \text{Bernoulli}(\sigma(\langle x_t^\tau, \theta^* \rangle)) \quad (6.8)$$

where θ^* is the true parameter and $\epsilon_t^\tau = r_t^\tau - \sigma(\langle x_t^\tau, \theta^* \rangle)$ is independent zero-mean noise with variance Z . Also, in what follows, we will assume that the tensors $x_t, x_t^\tau, \theta \in \mathbb{R}^{(k+1) \times k \times m}$ have been *flattened/vectorized* appropriately to vectors of dimension $(k+1)km$.

Parameter Estimation: The maximum likelihood estimate of the parameter θ after τ episodes, $\hat{\theta}_\tau \in \mathbb{R}$ is the unique solution of the following equation:

$$\sum_{s=1}^{\tau} \sum_{t=1}^{H_s} (r_t^s - \sigma(\langle x_t^s, \theta \rangle)) x_t^s = 0 \quad (6.9)$$

Let $\text{CI}_{\tau+1} = \{\theta \in \mathbb{R}^{(k+1)km} : \|\theta - \hat{\theta}_\tau\|_{V_\tau}^2 \leq \beta_{\tau+1}\}$ be the confidence interval based on $\hat{\theta}_\tau$ that contains θ^* with probability at least $1 - \delta$. The feature matrix V_τ is defined as

$$V_\tau = \lambda_1 \mathbb{I}_d + \sum_{s=1}^{\tau} \sum_{t=1}^{H_s} (x_t^s)(x_t^s)^\top \quad (6.10)$$

To balance exploration and exploitation, *OPT-CSEQ* uses the upper confidence bound of the reward (transition probability in our case) as used in the UCB-GLM algorithm [67] for generalized linear bandits. Thus the optimistic estimate of the probability

Algorithm 6.1: GET-POLICY

```

Get-Policy( $x_{\tau}^{\text{feat}}, \hat{\theta}, V$ ):
     $x_0 := \mathbf{o} \in \mathbb{R}^{(k+1)km}$ ,  $s_0 = (x_0, \{\})$ 
    queue :=  $[s_0]$ ,  $S := [\mathcal{T}]$ ,  $R[\mathcal{T}] := 1$ 
    repeat
         $(x_{\text{cur}}, \text{cseq}_{\text{cur}}) := \text{queue.pop}()$ 
         $s_{\text{cur}} := (x_{\text{cur}}, \text{cseq}_{\text{cur}})$ 
        for  $c \leftarrow 1, 2, \dots, k$  do
             $\text{cseq}_{\text{next}} := \text{cseq}_{\text{cur}}, a$ 
             $s_{\text{next}} := (x_{\text{cur},c}, \text{cseq}_{\text{next}})$ 
             $\tilde{P}(s_{\text{cur}}, c, \mathcal{T}) = \sigma(\langle x_{\text{cur},c}, \hat{\theta} \rangle + \alpha \|x_{\text{cur},c}\|_{V^{-1}})$ 
             $\tilde{P}(s_{\text{cur}}, c, s_{\text{next}}) := 1 - \tilde{P}(s_{\text{cur}}, c, \mathcal{T})$ 
             $S := [S, s_{\text{next}}]$ 
             $R[s_{\text{next}}] := 0$ 
            if length(cseqnext) < H then
                queue := [queue, snext]
        until queue NOT EMPTY;
        Compute optimal policy  $\tilde{\pi}$  for MDP( $S, [k], \tilde{P}, R$ )
    return  $\tilde{\pi}$ 

```

of attaining mastery based on the context $x_t^{\tau+1}$ is provided by the upper confidence bound

$$\begin{aligned} \tilde{p}(x_t^{\tau+1}) &= \max_{\theta \in CI_{\tau+1}} \sigma(\langle x_t^{\tau+1}, \theta \rangle) \\ &= \sigma(\langle x_t^{\tau+1}, \hat{\theta}_\tau \rangle + \beta_{\tau+1} \|x_t^{\tau+1}\|_{V_\tau^{-1}}) \end{aligned} \quad (6.11)$$

Solving the optimistic MDP: The policy computation procedure has been illustrated in Algorithm 6.1. The optimistic policy $\tilde{\pi}_\tau$ is computed by growing a state space tree of depth H , as shown in Figure 6.2. The root node s_0 is the initial state corresponding to the $(k+1)km$ dimensional zero vector and an empty action list - $(\mathbf{o} \in \mathbb{R}^{(k+1) \times k \times m}, \{\})$. The states are iteratively generated and added to a queue until the tree depth reaches H . A state $s_{\text{cur}} = (x_{\text{cur}}, \text{cseq}_{\text{cur}})$ is popped from the queue and expanded based on every possible action at that state. For each action in $[k]$, the child nodes of a state include the terminal state \mathcal{T} and a non-terminal state $(x_{\text{cur},c}, [\text{cseq}_{\text{cur}}, c])$ where $x_{\text{cur},c}$ is computed using (6.3). The optimistic transition probability $\tilde{p}(\mathcal{T}|s_{\text{cur}}, c)$ is computed using Equation (6.11) with $\beta_\tau = \alpha > 0$. The reward R for all the states is 0, except for the terminal state \mathcal{T} which has a reward of 1. Once the state space tree along with the optimistic transition probabilities \tilde{P} are computed, value iteration is performed to compute the optimal policy. Owing to the tree structure of the state space, it can be

Algorithm 6.2: OPT-CSEQ

Input: Tuning parameters T_0, λ_1, α and G

$$V = \lambda_1 I_d$$

for $\tau \leftarrow 1, 2, \dots, T_0$ **do**

- $\pi_\tau := \text{GET-RANDOM-POLICY}()$
- Play-Policy**(π_τ)

Set $j := 1, p := \det(V)$ **for** $\tau \leftarrow T_0 + 1, T_0 + 2, \dots$ **do**

- if** $\tau = T_0 + 1$ or $\det(V) \geq (1 + G)p$ **then**

 - Set $p := \det(V)$
 - Calculate the MLE $\hat{\theta}_j$ using (6.9)
 - $V_j := V$
 - $j := j + 1$

- $x_\tau^{\text{feat}} := \text{GET-STUDENT-CONTEXT}()$
- $\tilde{\pi}_\tau := \text{GET-POLICY}(x_\tau^{\text{feat}}, \hat{\theta}_j, V_j)$
- Play-Policy**($\tilde{\pi}_\tau$)

Procedure **Play-Policy**(π):

- $t = 0, x_0 := \mathbf{o} \in \mathbb{R}^{(k+1)km}, \text{seq} := \{ \}$
- repeat**

 - $t := t + 1$
 - $s_t := (x_{t-1}, \text{seq})$
 - Choose action $c_t := \pi(s_t)$
 - Observe reward r_t
 - Update:** $X \leftarrow [X, x_{t,c_t}], R \leftarrow [R, r_t]$
 - $\text{seq} \leftarrow [\text{seq}, c_t], V := V + (x_{t,c_t})(x_{t,c_t})^\top$

- until** $r_t = 1$ or $t = H$;

easily seen that value iteration would converge in $\Theta(H)$ iterations.

OPT-CSEQ has been described in Algorithm 6.2. Random policies are executed during the initial T_0 episodes so as to initialize the feature matrix V and get an initial estimate of θ based on the contexts observed. *GET-RANDOM-POLICY()* returns a policy where random actions are chosen at each state. After T_0 episodes, the maximum likelihood estimate, $\hat{\theta}_1$ is computed using (6.9). To save computations, the estimate of θ is next recomputed only when the determinant of V , $\det(V)$ increases by a constant factor $(1 + G)$. When $\hat{\theta}$ is recomputed for the j^{th} instance, it is denoted as $\hat{\theta}_j$ and the episodes that follow until the next re-computation of $\hat{\theta}$ form the j^{th} phase. For each episode $\tau > T_0$, the optimistic policy in the j^{th} phase is computed based on the student context $x_\tau^{\text{feat}}, \hat{\theta}_j$ and the feature gram matrix V_j . The policy $\tilde{\pi}_\tau$ is played until mastery

is achieved or the horizon H is reached. The contexts and rewards in each episode are used to update the estimate of θ and perform optimistic planning in the next phase.

If the student contexts are grouped into g clusters with a representative context vector for each cluster, then the optimistic policies for each group can be computed based on the representative vectors whenever $\hat{\theta}_j$ is estimated. These policies can be used in the j^{th} phase. This helps to save computing the policy in each episode for every student context.

Exploration-exploitation tradeoff: In OPT-CSEQ, the optimistic estimate of the probability of attaining mastery based on the context $x_t^{\tau+1}$ is provided by the upper confidence bound in (6.11) where $(\beta_\tau)_\tau$ is an increasing sequence of constants with $\beta_1 > 1$. However the quantity $\|x_t^{\tau+1}\|_{V_\tau^{-1}}$ decreases as more samples are obtained and the feature matrix V_τ grows. Thus the exploration gradually diminishes as more observations are gathered.

6.3 THEORETICAL ANALYSIS

Theorem 6.3.1 (Regret of the OPT-SEQ algorithm). *Assume that $\|x_t^\tau\|_2 \leq 1 \forall \tau, t \in [H]$ and $\|\theta^*\|_2 \leq S, \lambda_1 = 1$. Then with probability at least $1 - \delta$, for all $T > 0$, the regret of the OPT-SEQ algorithm satisfies*

$$\Delta_T \leq T_0 + 2L\sqrt{G_1(1+G)} \sqrt{2THd \log\left(\frac{TH}{d}\right)} \left(S + Z\sqrt{d \log\left(1 + \frac{2TH}{d}\right) + 2 \log \frac{1}{\delta}}\right)$$

where T_0 is the number of initial episodes where actions are randomly chosen and L is the Lipschitz constant of $\sigma(\cdot)$.

Consider the true MDP M_* , an empirical MDP \hat{M}_τ and an optimistic MDP \tilde{M}_τ which are identical except for their transition probabilities p, \hat{p}_τ and \tilde{p}_τ . Let π^* and $\tilde{\pi}_\tau$ denote the optimal policies on M_* and \tilde{M}_τ respectively and let $\eta^\pi(M)$ represent the value of policy π in M .

Lemma 6.3.2. *Let p_r be the probability of attaining mastery at time step r , $1 \leq r \leq H$, based on a policy π in M . Then*

$$\begin{aligned} \eta^\pi(M) &= p_1 + (1 - p_1)p_2 + \dots + \prod_{r=1}^{H-1} (1 - p_r)p_H \\ &= 1 - \prod_{r=1}^H (1 - p_r) \end{aligned}$$

Let p_i^* and \tilde{p}_i be the probabilities of attaining mastery at time step i , $1 \leq i \leq H$, based on π in M_* and \tilde{M}_t respectively.

Lemma 6.3.3. *For any policy π and $\tau > 0$, with probability at least $1 - \delta$, $\eta^\pi(M_*) \leq \eta^\pi(\tilde{M}_\tau)$*

Proof. With probability at least $1 - \delta$,

$$\begin{aligned} p_i^* \leq \tilde{p}_i \quad \forall i \in [H] &\implies \prod_{i=1}^H (1 - p_i^*) \geq \prod_{i=1}^H (1 - \tilde{p}_i) \\ &\implies 1 - \prod_{i=1}^H (1 - p_i^*) \leq 1 - \prod_{i=1}^H (1 - \tilde{p}_i) \\ &\implies \eta^\pi(M_*) \leq \eta^\pi(\tilde{M}_\tau) \quad (\text{using Lemma 6.3.2}) \end{aligned}$$

□

Lemma 6.3.4. *With probability at least $1 - \delta$,*

$$\eta^\pi(\tilde{M}_\tau) - \eta^\pi(M_*) \leq \sum_{t=1}^H (\tilde{p}_t - p_t^*)$$

Proof.

$$\begin{aligned} \eta^\pi(\tilde{M}_\tau) - \eta^\pi(M_*) &= [\tilde{p}_1 - p_1^*] + [(1 - \tilde{p}_1)p_2^* - (1 - p_1^*)p_2^*] + \dots + \\ &\quad \left[\prod_{i=1}^{H-1} (1 - \tilde{p}_i) \tilde{p}_H - \prod_{i=1}^{H-1} (1 - p_i^*) p_H^* \right] \quad (\text{using Lemma 6.3.2}) \\ &= [\tilde{p}_1 - p_1^*] + (1 - \tilde{p}_1) \left[\tilde{p}_2 - \frac{(1 - p_1^*)}{(1 - \tilde{p}_1)} p_2^* \right] + \dots + \\ &\quad \prod_{i=1}^{H-1} (1 - \tilde{p}_i) \left[\tilde{p}_H - \frac{\prod_{i=1}^{H-1} (1 - p_i^*)}{\prod_{i=1}^{H-1} (1 - \tilde{p}_i)} p_H^* \right] \end{aligned}$$

With probability at least $1 - \delta$,

$$p_i^* \leq \tilde{p}_i \quad \forall i \in [H] \implies \frac{\prod_{i=1}^j (1 - p_i^*)}{\prod_{i=1}^j (1 - \tilde{p}_i)} \geq 1 \quad 1 < j < H$$

Therefore,

$$\begin{aligned} \eta^\pi(\tilde{M}_\tau) - \eta^\pi(M_*) &\leq (\tilde{p}_1 - p_1^*) + (1 - \tilde{p}_1)(\tilde{p}_2 - p_2^*) \\ &\quad + \dots + \prod_{i=1}^{H-1} (1 - \tilde{p}_i)(\tilde{p}_H - p_H^*) \\ &\leq \sum_{i=1}^H \tilde{p}_i - p_i^* \quad (\text{since } \prod_{i=1}^j (1 - \tilde{p}_i) \leq 1 \quad H > j > i) \end{aligned}$$

□

Proof of Theorem 1. We first bound the one episode regret for $\tau > T_0$. Let $\tilde{\pi}_\tau$ be the policy played in episode τ and let x_t^τ be the context associated with time step t of episode τ based on $\tilde{\pi}_\tau$. Then,

$$\begin{aligned} \text{Regret of episode } \tau, \Delta_\tau &= \eta^{\pi_\tau^*}(M_*) - \eta^{\tilde{\pi}_\tau}(M_*) \\ &\leq \eta^{\pi_\tau^*}(\tilde{M}_\tau) - \eta^{\tilde{\pi}_\tau}(M_*) \quad (\text{Lemma 6.3.3}) \\ &\leq \eta^{\tilde{\pi}_\tau}(\tilde{M}_\tau) - \eta^{\tilde{\pi}_\tau}(M_*) \end{aligned} \quad (6.12)$$

$$\leq \sum_{t=1}^H \tilde{p}_t^\tau - p_t^{*\tau} \quad (\text{Lemma 6.3.4}) \quad (6.13)$$

where \tilde{p}_t^τ and $p_t^{*\tau}$ are the probabilities of attaining mastery at time step t , $1 \leq t \leq H$, of episode τ based on $\tilde{\pi}_\tau$ in \tilde{M}_τ and M_* respectively. Let $\tilde{\theta}_t^{\tau-1} = \arg \max_{\theta \in C_{\tau-1}} \langle x_t^\tau, \theta \rangle$. Then from (6.13), we have,

$$\begin{aligned} \Delta_\tau &\leq \sum_{t=1}^H \sigma(\langle x_t^\tau, \tilde{\theta}_t^{\tau-1} \rangle) - \sigma(\langle x_t^\tau, \theta^* \rangle) \\ &\leq L \sum_{t=1}^H |\langle x_t^\tau, \tilde{\theta}_t^{\tau-1} \rangle - \langle x_t^\tau, \hat{\theta}_{\tau-1} \rangle| + |\langle x_t^\tau, \hat{\theta}_{\tau-1} \rangle - \langle x_t^\tau, \theta^* \rangle| \end{aligned}$$

(σ is increasing Lipschitz function with Lipschitz constant L)

Let H_τ be the number of samples obtained in the τ^{th} episode. Define $V_\tau = \sum_{s=1}^\tau \sum_{t=1}^{H_s} (x_t^s)(x_t^s)^\top$ and $V_{\tau,h} = V_\tau + \sum_{t=1}^h (x_t^{\tau+1})(x_t^{\tau+1})^\top$. Note that $\tilde{\theta}_t^\tau \in CI_{\tau-1}$ and $\theta^* \in CI_{\tau-1}$ with probability at least $1 - \delta$. From Lemma 12 of [68], for all $\theta \in CI_{\tau-1}$

$$\begin{aligned} |\langle \theta - \hat{\theta}_{\tau-1}, x_t^\tau \rangle| &\leq \sqrt{G_1} \|V_{\tau-1}^{1/2}(\theta - \hat{\theta}_{\tau-1})\|_2 \sqrt{x_t^\tau \top V_{\tau-1,t-1}^{-1} x_t^\tau} \\ &\left(\text{where } G_1 > 0 \text{ is a constant such that } \frac{\det(V_{\tau-1,t-1})}{\det(V_{\tau-1})} \leq G_1 \forall \tau \in [T], t \in [H] \right) \end{aligned}$$

In *OPT-CSEQ*, the maximum likelihood estimate of θ is recomputed only when $\det(V_\tau)$ increases by a constant factor $(1 + G)$. Let T_τ be the smallest episode less than or equal to τ such that $\hat{\theta}_\tau = \hat{\theta}_{T_\tau}$. Then by plugging $\hat{\theta}_\tau = \hat{\theta}_{T_\tau}$ in the above equation we have

$$|\langle \theta - \hat{\theta}_{T_\tau}, x_t^\tau \rangle| \leq \sqrt{G_1(1+G)\beta_{T_\tau}} \|x_t^\tau\|_{V_{\tau-1,t-1}^{-1}}$$

Therefore,

$$\Delta_\tau \leq 2L \sqrt{G_1(1+G)\beta_{T_\tau}} \sum_{t=1}^H \|x_t^\tau\|_{V_{\tau-1,t-1}^{-1}} \quad (6.14)$$

Total regret in T episodes,

$$\begin{aligned}\Delta_T &= \sum_{\tau=1}^{T_0} [\eta^{\pi_\tau^*}(M_*) - \eta^{\pi_\tau}(M_*)] + \sum_{\tau=T_0+1}^T [\eta^{\pi_\tau^*}(M_*) - \eta^{\tilde{\pi}_\tau}(M_*)] \\ &\quad (\pi_\tau \text{ is the random policy used in the } \tau^{\text{th}} \text{ episode } \tau \leq T_0) \\ &\leq T_0 + \sum_{\tau=T_0+1}^T \sum_{t=1}^H 2L |\langle (\theta - \hat{\theta}_{\tau-1}), x_t^\tau \rangle| \\ &\leq T_0 + 2L \sqrt{G_1(1+G)\beta_T} \sum_{\tau=T_0+1}^T \sum_{t=1}^H \|x_t^\tau\|_{V_{\tau-1,t-1}^{-1}}\end{aligned}$$

From Lemma 2 of [67]

$$\sum_{\tau=T_0+1}^T \sum_{t=1}^H \|x_t^\tau\|_{V_{\tau-1,t-1}^{-1}} \leq \sqrt{2THd \log(TH/d)}$$

$$\Delta_T \leq T_0 + 2L \sqrt{G_1(1+G)\beta_T} \sqrt{2THd \log(TH/d)}$$

From Theorem 2 of [68]

$$\Delta_T \leq T_0 + 2L \sqrt{G_1(1+G)} \sqrt{2THd \log\left(\frac{TH}{d}\right)} \left(S + Z \sqrt{d \log\left(1 + \frac{2TH}{d}\right) + 2 \log \frac{1}{\delta}} \right)$$

□

6.4 EXPERIMENTS

This section describes the experiments performed on two settings to demonstrate the effectiveness of our algorithm. Since the proposed skill evolution model is novel and ours is the first algorithm for the setting, there are no prior established baselines as such. In the absence of such baselines, we have compared *OPT-SEQ* with well founded heuristic policies.

Setting I (without context): We first validated *OPT-CSEQ* for a fixed Ω (shown as a heat map in Figure 6.3) with $k = 10$ contents and horizon $H = 5$. Here, Ω does not vary based on student context. The value of the optimal policy η^{π^*} (see Lemma 6.3.2) associated with Ω is 0.91. We compared the performance of *OPT-CSEQ* with a random policy and a greedy policy which, from each state s_t , chooses an action that has the highest probability of transitioning to the terminal state based on the optimistic estimate \tilde{p} . Simulations were run for $T = 2000$ episodes with $T_0 = 150$, $C = 4$ and the results were averaged over 5 runs. We also made use of the fact that the components

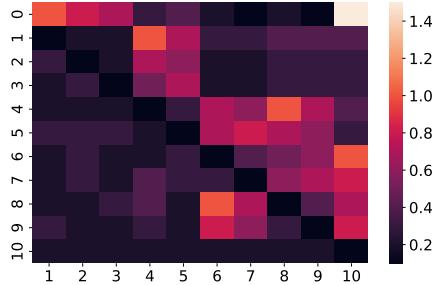


Figure 6.3: Heat map of the parameter matrix Ω used for experiment.

of θ are non-negative, and employed *projected gradient descent* while computing the maximum likelihood estimate of θ (i.e., the negative components of θ were set to 0 during each update). The value Ω as well as the other parameters used in this experiment are given in Appendix B.1.

It can be seen from the bar chart in Figure 6.4 that *OPT-CSEQ* outperformed the other two policies in terms of the average number of cases where mastery was achieved. Figure 6.5 shows the expected probability of mastery for the policies computed by each of the three techniques against the episode number. The values in the plot have been smoothed using a window of size 150. It can be seen from the figure that the 3 algorithms perform similarly for the first 150 episodes when random sequences are generated. Both *OPT-CSEQ* and Greedy algorithms eventually learn from the interaction data and generate better policies. However *OPT-CSEQ* learns policies with greater value as compared to the greedy algorithm, and eventually converges to the optimal policy. See Appendix B.2 for the results of additional experiments that were conducted in this setting with different values of Ω .

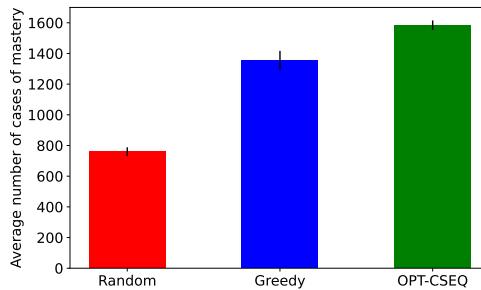


Figure 6.4: **Setting I (Without context)**: Comparing the average number of cases where mastery was attained for the three algorithms.

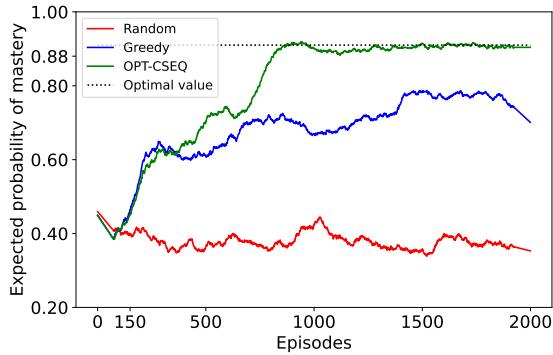


Figure 6.5: **Setting I (Without context):** The expected probability of mastery of the policies generated by the three algorithms (averaged over 5 runs) at each episode.

Setting II (with context): Next we looked at the performance of *OPT-CSEQ* in learning an optimal sequencing policy for students with context vectors of dimension $m = 3$ ($x^{\text{feat}} \in [0, 1]^3$). The contexts, in this case, can represent the normalized student scores in assessments based on related concepts. In this setting, we considered $k = 6$ contents and a horizon of $H = 3$ (here, $\theta \in \mathcal{R}^{7 \times 6 \times 3}$). We measured the expected probability of mastery for 4 different student contexts over 2500 episodes as shown in Figure 6.6. The values in the plot have been smoothed using a window of size 100. Black dotted lines represent the optimal policy values. It can be observed from the figure that *OPT-CSEQ*

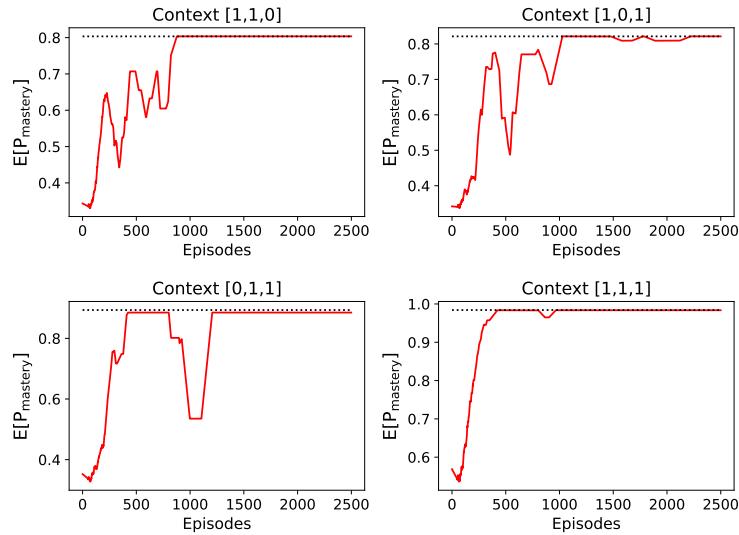


Figure 6.6: **Setting II (with context):** The expected probability of mastery for the policies computed by *OPT-CSEQ* (averaged over 5 runs) for the student contexts $[1, 1, 0]$, $[1, 0, 1]$, $[0, 1, 1]$ and $[1, 1, 1]$ over 2500 episodes.

is able to learn the optimal policy for all the four contexts. See Appendix B.3 for the value of θ and other parameters used in this experiment.

6.5 DISCUSSION

The performance of *OPT-CSEQ* could not be compared to any of the prior work due to the novelty of our skill dynamics and mastery model. The experiments in *Setting I* studied the performance of *OPT-CSEQ* against *Greedy* and *Random* policies for a fixed student profile and thereby a fixed Ω . The results show that *OPT-CSEQ* eventually converges to an optimal policy, whereas this is not always the case with the *Greedy* and *Random* policies.

Next in *Setting II*, we examined the effectiveness of *OPT-CSEQ* in learning the best sequences for various student profiles. During the experiment in this setting, the algorithm interacted with students having profiles ($x_{\tau}^{\text{feat}} \in [0, 1]^3$) and we tracked how the policies learned by *OPT-CSEQ* evolved with time for four different student contexts. And it was observed that in all of these four cases, *OPT-CSEQ* was able to learn the optimal policy using the upper confidence bounds for generalized linear bandits. The regret bound derived in Section 6.3 shows that the algorithm doesn't lose much in terms of the number of students achieving mastery, during the process of learning the value of θ and, consequently, the optimal policies for different student contexts.

Computational complexity: GET-POLICY (Algorithm 6.1) involves growing the state space tree and computing the optimal policy using value iteration. Growing the state space tree for time horizon H involves cost $O(k^H)$ and value iteration converges in $O(H)$ iterations due to the tree structure of the MDP. Thus the total complexity of Algorithm 6.1 is $O(Hk^H)$. Algorithm 6.2 *OPT-CSEQ* involves computing the maximum likelihood estimate of θ using gradient-descent when the determinant of the feature matrix V increases by a constant factor $(1 + G)$. The procedure Play-Policy has complexity $O(H)$.

6.5.1 Deployment and testing with real students

OPT-CSEQ has not yet been validated in a real educational setting involving student participants. Deploying and testing the algorithm would require the development of a learning platform that supports diverse content formats, such as videos and text, and enables student assessments wherein *OPT-CSEQ* can be applied to perform content

sequencing. Multiple learning resources covering a target concept must be integrated into the platform. Testing can be conducted over a time horizon $H = 5$ (corresponding to the number of days in a week), which necessitates the availability of at least five distinct types of learning content.

Experimental design: The efficiency of *OPT-CSEQ* can be evaluated by comparing the performance of the algorithm against two baselines. The first baseline is the *expert-defined assignment*, where learners are provided content in a sequence prescribed by a domain expert. The second baseline, *self-directed assignment*, allows learners to choose their own sequence of learning materials within the fixed horizon H . A proper evaluation of the algorithm would require testing with a sample size of about 500-1000 learners, as has been done in several previous studies. [51][55]. A pre-test is administered to assess the participants' initial competency related to the target concept. Similar to Bassan et al. [51], students are then assigned to one of the three conditions such that approximately 90% participants are assigned to *OPT-CSEQ*, with 5% each to *expert-defined assignment* and *self-directed assignment* conditions. The assignments are made in such a way that the average pre-test scores of participants across all three groups are equivalent.

For *OPT-CSEQ*, the context vectors of participants can be constructed using performance metrics from assessments on concepts related to the target concept, or by incorporating the learner's self-evaluation of their confidence in the subject area, following the approach adopted in the validation study of ETutor [55]. The participants then interact with the platform, wherein *OPT-CSEQ* dynamically determines the personalized sequence of content to be presented based on the student's context and the algorithm's prior interactions with other learners. Following the presentation of each instructional resource, assessments are administered to evaluate mastery of the target concept, with the results serving as feedback for the algorithm. For the other two conditions, assessments are performed at the end of the time horizon.

Evaluation: The impact of *OPT-CSEQ* can be assessed by comparing the percentage of participants who achieved mastery of the target concept using *OPT-CSEQ* with those in the two baseline conditions. Furthermore, the efficiency of the algorithm can also be assessed by comparing the percentage of students who achieved mastery across different values of n , where n represents the total number of participants in the *OPT-CSEQ* condition. Ideally, the percentage of students achieving mastery should be an increasing function of n , as the algorithm is expected to improve its sequencing policies with more interactions with the learners. A similar analysis can be conducted

based on the participants' profiles/contexts to determine if the algorithm performs consistently across various student profiles.

6.6 CONCLUSION

In this part of the thesis, we studied the problem of generating a personalized optimal sequence of content that maximizes the probability of a student mastering a concept within a given time horizon. With the assumption that the student's skill level at a given time step evolves based on the student's learning profile and the content presented in the past, we modeled the problem as an episodic, fixed time horizon MDP, with a generalised linear bandit model representing the probability of mastery at each time step. Based on this framework, we proposed *OPT-CSEQ*, a reinforcement learning based algorithm that uses an upper confidence bound algorithm to generate optimal personalized sequences of content. We have demonstrated *OPT-CSEQ* through numerical experiments and have also presented the regret analysis of the same. Finally we outlined future work focused on deploying and testing *OPT-CSEQ* with real students.

Part III

CONCEPT TAGGING OF MATHEMATICAL QUESTIONS

AUTOMATIC GENERATION OF CONCEPT TAGS FOR MATHEMATICAL QUESTIONS

Assigning concept tags/labels to questions enables intelligent tutoring systems to efficiently organize resources, assess student knowledge levels, and thereby decide the best activity or content to be recommended to the student. Manual tagging of questions by experts is a very laborious process. Automatic tagging techniques, on the other hand, can efficiently generate consistent tags at lower costs. Several automated tagging methods leveraging machine learning (ML) and natural language processing (NLP) have achieved moderate success in tagging questions in diverse subjects, including chemistry and English language [69, 70]. However, generating automatic tags for mathematical questions is demanding as the question text is mostly succinct and not rich enough to support tagging.

In this part of the thesis, we study the problem of automatic tagging of mathematical questions. We investigate if student explanations on solving a mathematical problem can be utilized for unsupervised automatic tag generation. Towards this, we conducted a study in a graduate-level linear algebra course where we asked the students to explain their problem-solving strategy for a selected set of questions in their weekly quizzes. We then developed a technique called *Unsupervised Skill Tagging (UST)* to tag questions with labels from a master concept list based on the most frequent n-grams in the set of explanations for each problem. UST demonstrates good performance on our dataset comprising the student explanations which reinforces the idea that student explanations indeed describe the right concepts required to solve a problem, which is exploited by *UST* to generate the correct question tags.

In this chapter, we first describe question tagging and then compare generative and supervised approaches for automatic tagging (Section 8.1). Next we discuss about automatic methods for tagging mathematical questions and explain why it is necessary to also consider the steps involved in problem-solving to generate accurate tags for math questions (Section 8.2). The problem is formally presented in Section 8.3. We then describe our study conducted in a graduate-level linear algebra course, to examine if student explanations to solving mathematical problems can be employed to generate

concept tags associated with those questions (Section 8.4). Finally we present an analysis of the manual tagging exercise conducted as a part of our experiment (Section 8.5).

7.1 BACKGROUND

7.1.1 Skill Model

Adaptive learning systems often employ a *skill model* that consists of the target knowledge components (KCs) or skills (addition of fractions in math, capitalization rules in English, etc.) to be learned by the students. A *skill model* supports several critical aspects of learning systems, including personalized topic recommendations for students, adaptive practice techniques such as spaced repetition, student modeling through skill estimation, and effective content management.

Despite the importance of creating a good quality skill model, (i) identifying the skills associated with the various topics covered in a course, and (ii) mapping course contents and assessment items to the skills, is extremely costly. Developing a skill model for a domain involves addressing important questions such as (i) how are skills defined? (ii) what is a suitable granularity of skills? (iii) how to model relations among skills? The answers to these questions typically requires considerable input from the domain experts and have a significant impact on many different aspects of the learning system [71].

7.1.2 Skill Tagging/Question Tagging

The question items in personalized learning systems are tagged with the labels of associated skills/concepts (defined in the *skill model*) required to solve each item. *Question tagging* or *skill tagging* facilitates organization of content wherein questions associated with the same set of concepts can be grouped together. Such a mapping is also essential for assessing the student knowledge on various concepts based on the student's response to a question and thereby recommend suitable exercises/learning resources to the student [10]. For instance, if a student learning English language provides an incorrect response to a question about *verb tenses*, it is necessary to provide the student with more practice specifically focused on verb tenses.

Labeling questions with skill labels is typically carried out by learning science practitioners such as subject matter experts, who inspect the question items and assign

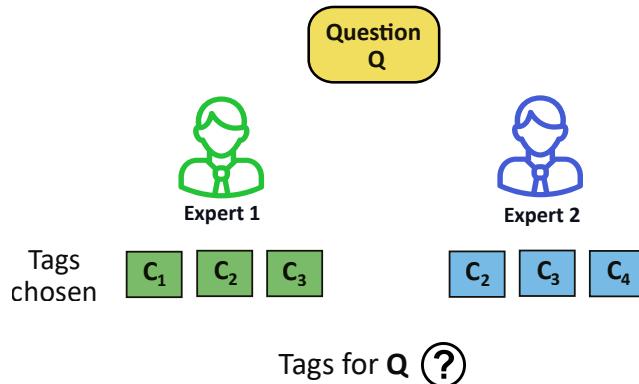


Figure 7.1: Manual tagging of questions becomes difficult when experts have differing views on the tags to use.

one or more skill tags to each question. This tagging procedure, also referred to as *manual tagging*, is time-consuming and labour intensive [69]. Manual tagging could also lead to consistency issues due to the difference in perspectives of individual taggers with respect to a question (Figure 7.1). Skill tagging might also get affected by the expert overlooking a skill that is intrinsic to them, but one that is critically assessed by the given problem. This happens due to automatization of certain cognitive processes in experts, referred to as *expert blind spot*, where the explicit steps taken to solve a problem can get grouped together when it becomes second-nature [72].

7.1.3 Automatic Tagging: Generative vs. Supervised approaches

Automatic tagging procedures can generate consistent concept tags faster at a lower cost. Several techniques to generate automatic question tags for educational content have been proposed in literature. Automated approaches for generating skill tags follow two main approaches: generation or classification [73].

The *generation* based techniques focus on creating skill tags based on extracting concepts from digital textbooks, student response data, etc. For instance, Matsuda et al. [70] proposed an unsupervised technique to generate skill model by mining latent skills from instructional text of assessment items in existing Biology and Chemistry courseware, and annotate the contents with the discovered skills. Text-mining based tagging approaches using student explanation text performed tag extraction from the response text [74–76]. Recently, GPT-4 was employed to automatically generate KCs for MCQs in Chemistry and E-Learning [73]. These methods often face challenges concerning the interpretability of the identified skills and the creation of knowledge

components that are meaningful for educators.

On the *classification* front, the goal is to assign existing KCs to problems by utilizing the information contained in the assessment text and in certain cases, other sources of information, such as KC descriptions. For instance, B. Sun et. al. [69] proposed a position-based attention model and keywords-based model to automatically associate questions with tags from a knowledge map for multiple choice questions (MCQs) in English course. Ramesh et. al. [49] proposed a semi-automatic method for tagging questions based on cognitive level, question type, content and difficulty level, using techniques such as n-grams keyword matching, semantic dictionary and domain ontology.

In this dissertation, we focus on the problem of *tagging of assessment items with skill labels*, assuming that the skill model has already been defined. The terms *skills*, *concepts* and *knowledge components* are mostly used as synonymous in this part of the thesis. In the next section, we discuss automatic tagging of mathematical questions, which is our problem of interest.

7.2 AUTOMATIC TAGGING OF MATHEMATICAL QUESTIONS

Much of the work [77–85] on automatic tagging techniques for math questions are supervised by nature. These models are designed to predict skill tags for items, after being trained on problems with similar skill tags and also by employing additional information such as KC descriptions. There have been limited attempts to utilize text mining techniques to produce skill tags/KCs for math questions, with no training or prediction involved. Some of the recent works have also employed large language models (LLMs) to perform tagging of math questions [86]. In the next section, we describe the prior research work on automatic tag generation for mathematical questions and later discuss the major challenges associated with this problem.

7.2.1 Related work on tagging of mathematical questions

The methods for supervised tagging of math questions can be grouped based on the type of input features used by the model to perform tagging. The first group involves techniques which perform tagging using only the mathematical expressions/formulae [77] and the question text or images provided along with it [78–82]. The machine learning models used in these studies include Support Vector Machine and Nearest

Neighbor [79], Multi-layer Perceptron [80], BiLSTM with label-semantic attention [78], deep learning models such as Convolutional Neural Networks (CNN) and BERT [77, 81]. Tian et al. [82] proposed an ensemble model for tagging Japanese math exercises by combining a word embedding based method and a keyphrase extraction technique. The second category of approaches incorporates additional input features, either in conjunction with or independent of the problem description and other associated attributes. These include tagging using skip-gram based embeddings based on only the problem sequences (and not the question text) [83], and a BERT model trained using KC descriptions and instructional video titles along with problem descriptions [84]. Li et al. proposed a novel combination of machine learning techniques to help automate tagging open educational resources including text, image figures, and videos, with math skills from an existing taxonomy as well as those from any newly introduced taxonomy [85].

Although these techniques have demonstrated promising results, the necessity of labeled samples during the training process is a significant limitation. Often obtaining labeled data is expensive or even unrealistic in some cases. The text mining based unsupervised tagging approaches studied in literature, discussed in detail in a later part of this section, are mostly semi-automatic by nature [74–76]. These are *tag generation* techniques that involves some kind of human effort to arrive at the final tags.

Large language models can help overcome the challenges in collecting large-scale, expertise-annotated datasets as required by supervised tagging methods. For instance, Li et al. studied the zero/few-shot learning capability of a variety of representative LLMs in tagging elementary math questions with 12 knowledge concepts [86] and the experiments yielded encouraging results.

However, despite the significant advancements in artificial intelligence, automatic tagging methods have yet to reach the same level of performance as humans. A study [87] that explored the effect of Human+AI collaboration on the task of tagging educational content with math skills from the US common core taxonomy, reported that Human+AI group saved almost half the time as compared to the Human only control group in performing skill tagging. But in terms of accuracy, the Human+AI group performed worse than the Human-only group, but better than the AI model. This illustrates that there is still scope for enhancement of automatic tagging methods for math problems. In the following part, we will discuss the key challenges associated with this problem.

7.2.2 Challenges in automatic tagging of math questions

Generating tags for mathematical questions is quite demanding as the question text is mostly succinct and not rich enough to support tagging. The answers to such questions are mostly numerical values, which do not add any value to tag generation. Also, solving such questions usually involve application of multiple skills/concepts which might not be entirely captured by the question text. For instance, consider the question in Figure 7.2. We can deduce from the question text that in order to answer this question, a student must be familiar with concepts like *outer product* and *nullity*. However solving this problem also requires knowledge of other concepts such as *rank* and *rank-nullity theorem*, which is not evident from the question text. Therefore, for automatic tagging systems to effectively tag mathematical questions, the various steps taken by the student to arrive at the solution must be accurately mapped to the corresponding skills or knowledge components.

7.2.3 Integrating human knowledge into automatic tagging

Crowdsourcing is a prominent model for distributed problem-solving and is well suited for tasks in which human judgement is critical. Learnersourcing [88] is a form of crowdsourcing with learners as crowd, where input from a group of learners is used to help improve the educational experience of future learners. Learnersourcing differs from crowdsourcing in that it utilizes a specific group of motivated learners who are actively involved in their own learning process, as opposed to a generic crowd.

Learnersourcing has been a popular model for the development and refinement of educational content [88–90]. For example, Williams et al. [90] created a system called AXIS, which had learners generating explanations for arithmetic word problems, which were then used to assist future learners. It was observed that the explanations generated by the learners were comparable in quality to those produced by expert instructors.

Recent studies have also employed learnersourcing for associating assessment items with knowledge components (KCs) [74–76], where each KC is “an acquired unit of cognitive function or structure that can be inferred from performance on a set of related

The nullity of the outer product between the normal of hyperplanes $x + y - z = 0$ and $2x - y - 3z = 0$ in \mathbb{C}^3 is _____.

Figure 7.2: A sample question from a graduate level linear algebra course.

tasks” [91]. Moore et al. [74] studied whether KCs identified by domain experts could be generated through crowd-sourcing by asking participants to concisely list three skills required to solve a problem in mathematics and English writing. The participants’ responses were coded and compared to the KCs identified by an expert. It was found that roughly 33% of the crowdsourced KCs directly matched those generated by domain experts for the same problem. While the participants for this study were selected through the Amazon Mechanical Turk (AMT) platform, a similar study was conducted with learners from two online courses in chemistry and programming [76]. In this case, it was found that 12% of the skill tags generated by students matched expert generated KCs and more than half were not usable. Both studies required considerable human effort to code the participant responses and conduct the analysis.

Moore et al. [75] also studied the efficacy of using topic modeling approaches for identifying relevant KCs from student explanations for math and English writing questions. It was observed that interpreting the topics as KCs was difficult and none of the topics were found to be directly related to any of the expert generated KCs, which made it unsuitable for automatic tag generation.

While all the above studies focused on using crowdsourced explanations/responses to address the problem of *generation of skill tags for assessment items*, we study the *classification problem* of mapping assessment items to a set of pre-defined skill tags using learnersourced explanations on problem solving. Similar to Moore et al. [75], we investigate the viability of using natural language processing (NLP) techniques on the explanation text to perform the tagging.

The *key idea* in this work is that, when explicitly asked to explain their response to a given mathematical problem/question, students who have the right approach to solving a problem would mention the right concepts related to the question in their explanation. In the next section, we describe the specific problem of interest.

7.3 PROBLEM DEFINITION

In this dissertation we consider the problem of automatic tagging of *math word problems* with labels of *concepts* from a pre-defined *skill model* using student explanations on solving the problem. The domain we consider is a graduate level course for undergraduate and masters students.

The *math word problems* considered in this study are mathematical exercises presented in the form of written or verbal descriptions. These problems require individuals to decipher the information provided, identify and apply the relevant mathematical concepts to solve the given problem, rather than use straightforward math equations, formulas or numeric computations/manipulations. Solving such problems typically involves critical thinking, problem-solving ability, and the application of relevant mathematical concepts to arrive at the answer. A sample problem is given in Figure 7.2.

A skill model defined for the domain contains the *concepts* used to tag the *math word problems*, known as *concept labels*. Each *math word problem* gets tagged with *one or more concept labels*. Thus the problem can be formally defined as follows.

Given a set of *math word problems* \mathcal{Q} and a set of *concept-labels* $\mathcal{C} = \{C_1, C_2, \dots, C_n\}$, for each $q \in \mathcal{Q}$, find the label subsets or label vectors $y_q = (y_{q1}, \dots, y_{qn}) \in \{0, 1\}^n$ such that each label attribute y_{qi} corresponds to the absence (0) or presence (1) of concept label C_i for the problem q based on the set of student explanations \mathcal{E}_q for solving q . Here \mathcal{E}_q comprises the collection of explanatory texts written by students detailing the steps they employed to solve the question q .

7.4 EXPERIMENT SETUP

In this section, we describe the experimental setup used to collect data from the students. The study was conducted during the *Linear Algebra for Engineers* course in the *Fall 2020* semester at *Institute of Technology Palakkad*. The course was offered in online mode due to COVID-19 [92] and was credited by 30 masters students.

REGULAR QUIZZES As a part of continuous evaluation, five weekly/regular quizzes Q_1, Q_2, \dots, Q_5 were held through Moodle. Each quiz consisted of about 5-7 questions based on the topics covered in each week. The quizzes consisted of a mix of

$$\text{Let } M = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 1 & -2 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 \\ 1 & 0 & 4 & -1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

The eigen values of $(M^2)^T$ in ascending order are , , , , .

-1	1	6	4	-4	3	0	7	-3	5	8	2	9	-2
----	---	---	---	----	---	---	---	----	---	---	---	---	----

Figure 7.3: A question asked in one of the regular weekly quizzes of the linear algebra course.

Table 7.1: Concept-labels included in the master tag-list.

orthonormal/orthogonal matrix, trace, transpose, frobenius norm, orthogonal projection, rank, range space, idempotent matrix, dimension, symmetry of inner product, linearity of inner product, positive definiteness of inner product, subspace addition, subspace intersection, linear independence, outer product, rank nullity theorem, null space, linear map composition, nullity, eigenvalues triangular matrix, eigenvalues matrix powers, eigenvalues matrix transpose, eigenvalue, eigenvector, invertibility, function space, derivative, inner product, linear transformation, nilpotent matrix, span

multiple-choice and numerical questions that were auto-evaluated. An example question has been provided in Figure 7.3.

MASTER TAG-LIST A *master tag-list* was constructed based on the important concepts covered in the course, as shown in Table 7.1. A total of 32 concept-tags were included in the *master tag-list* and these tags were to be used for labeling the questions.

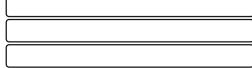
SUPPLEMENTARY QUIZZES Each of the regular quiz Q_1, Q_2, \dots, Q_5 were accompanied by a supplementary quiz SQ_1, SQ_2, \dots, SQ_5 . In each supplementary quiz SQ_i , the students were asked to perform additional activities for two selected questions included in Q_i .

EXPERT-DEFINED CONCEPT LABELS The concepts required to solve each of the regular quiz questions included in the supplementary quiz were identified by the course instructor from the *master tag-list*. These are referred to as the *expert-defined concept labels*. The 10 questions used in this study along with the *expert-defined concept labels* are given in Appendix C.1

For each question selected from the regular quiz, the following two activities were included in the supplementary quiz.

1. *Manual tagging* using a drag-and-drop activity where the student selects the labels of the concepts which they think are required to solve the given problem from a subset of the tags from the *master tag-list* (Figure 7.4). The drag-and-drop activity was scored based on the level of agreement of the labels selected by the student with those decided by the instructor.
2. *Explanation of problem solving approach* where the student justifies the choices made in the earlier part by explaining how they solved the problem (Figure 7.5).

Let $\mathbf{M} = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 1 & -2 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 \\ 1 & 0 & 4 & -1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}$. Find the eigenvalues of $(\mathbf{M}^2)^T$.



Transpose Eigenvector Eigenvalues-Triangular-Matrix
 Eigenvalue Similarity None Eigenvalues-Matrix-Powers
 Triangular Matrix Upper Triangular Matrix
 Lower Triangular Matrix Eigenvalues-Matrix-Transpose

Figure 7.4: A drag-and-drop activity used in one of the secondary quizzes.

Separate time and marks were allocated for supplementary quiz to motivate the students to attempt the quiz earnestly. Student responses for a total of 10 questions were thus collected. The responses to these questions were evaluated by the teaching assistant for the course. The explanations were also intended for the students to learn by reflecting on their problem solving approach. The majority of students who participated in the feedback survey at the end of the course stated that they found the supplementary quizzes useful and that it helped them in reflecting on their conceptual understanding.

7.5 MANUAL TAGGING

Based on the student responses to the drag-and-drop activity, we attempted to perform manual tagging by assigning the most frequently chosen concept labels to the question. Figure 7.6 shows four instances where this technique did not work well. Each bar chart displays the number of selections of each concept tag in the drag-and-drop activity for the questions in Table 7.2 by students who answered the question correctly in the regular quiz. The bar chart labels representing the expert-defined tags for each

Justify your above selection in a few lines. Clearly explain how the selected concepts can be used to solve the given problem.

Figure 7.5: A sample question from one of the supplementary quizzes for the student to explain how they solved the quiz question.

question are highlighted in red.

It can be observed from the plots that except for one or two expert-defined concepts, none of the other tags received a strong mandate that distinguishes them from the non-relevant labels. Note that during the drag-and-drop activity, the students were presented with around 10 concept tags to choose from, which also contained the expert-defined labels. Therefore the activity was easier as compared to an actual tagging procedure where the tags are to be selected from the list of all available concept labels. This perhaps points to a flaw of manual tagging where multiple taggers may not have a consensus regarding the concept-tags associated with a question. An explanation to this phenomenon could be that some students might have missed out a few concepts while connecting the problem solving logic with the concept labels, or they could have inadvertently chosen some wrong labels due to the similarity between the concepts. In any case, this surprisingly poor performance of manual tagging might be of interest of future study from a behavioral perspective.

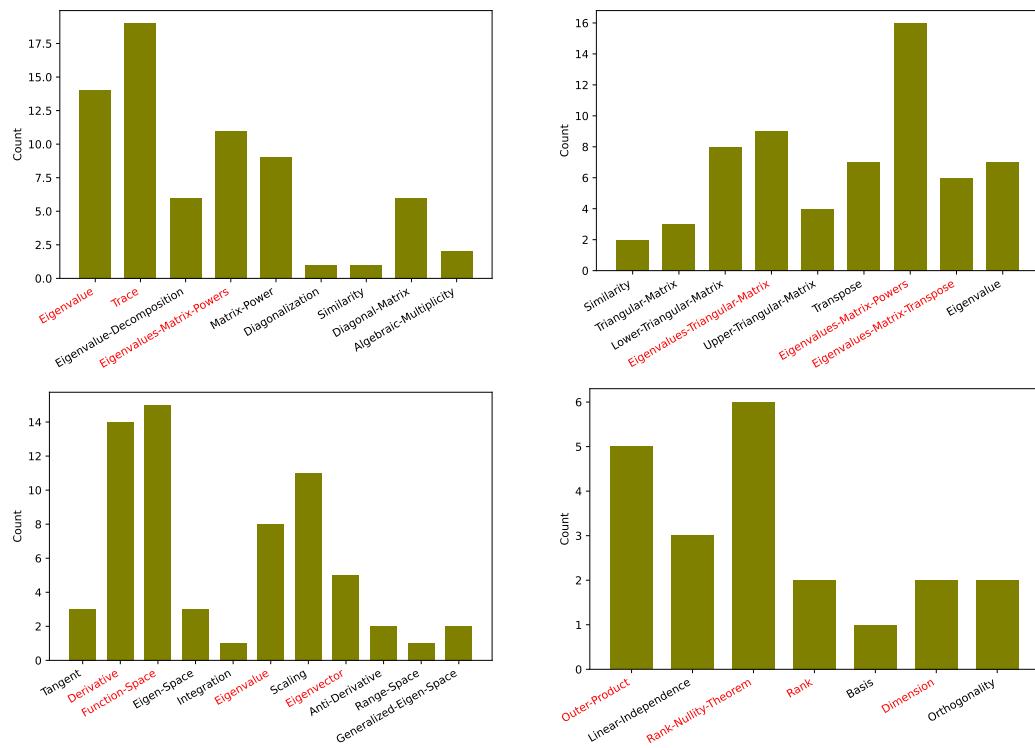


Figure 7.6: Bar plots of concept labels selected by students who correctly solved the problems in Table 7.2. The bar chart labels representing the expert-defined tags for each question are highlighted in red.

Table 7.2: Questions associated with the bar plots in Figure 7.6.

Q1	If the eigen values of \mathbf{M} are $-2, 0, 1, 1, 3, 4$ then the trace of $(3\mathbf{I} - 2\mathbf{M})^2$ is
Q2	Let $\mathbf{M} = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 1 & -2 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 \\ 1 & 0 & 4 & -1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}$ The eigen values of $(\mathbf{M}^2)^T$ in ascending order are:
Q3	The differentiation of $f(x)$ in function space w.r.t x is a linear transformation in x , which of the following is an eigen function of this linear transformation?
Q4	The nullity of the outer product between the normal of the hyperplanes $x + y - z = 0$ and $2x - y - 3z = 0$ in \mathbb{C}^3 is:

7.6 SUMMARY

Question tags are utilized by educational technology systems to organize content as well as to perform ability analysis of students, which helps with intelligent question recommendation. Question tagging demands domain expertise and is labour intensive and time consuming. Automated approaches have emerged as valuable tools which can scale the tagging process efficiently. However, automatic tagging of mathematical questions is a challenging task. Most of the machine learning based automatic tagging methods for math problems utilize only the question text and any accompanying mathematical expressions/formula or images to perform tagging. This might not always work well because (i) the questions are usually concise, and (ii) the answers are mostly numerical values. Efficient tagging of math questions requires mapping the problem-solving approach to the relevant skills or knowledge components.

Prior research has investigated the use of learnersourcing to generate tags for mathematical problems. In this thesis, we examine if this idea can be adopted to perform tagging of math questions with concept-labels from a pre-defined skill model. We conducted an experiment in a graduate-level linear algebra course to test our idea, where we asked the students to explain their problem-solving strategy for a selected set of questions from their weekly quizzes. The students were also instructed to perform manual tagging by choosing the labels of the concepts they deemed necessary to solve the given problem from a limited set of tags derived from the skill model. An analysis of the frequency distribution of the tag selections revealed that, with the exception of one or two expert-defined concepts, none of the other tags received a firm mandate that distinguished them from the non-relevant labels. This shows that manual tagging does not always work well for mathematical questions.

8

UNSUPERVISED SKILL TAGGING (UST)

It was observed from the experiment results in the previous chapter that manual tagging did not work well in several instances. This could be due to: (i) the students failing to recall all the necessary concepts used to solve a problem, or (ii) the students mistakenly choosing wrong tags due to the similarity between the concept labels. Therefore, we next study if student explanations on solving a problem can be utilized to improve the tagging process for mathematical questions. The key idea in this work is that students who possess the correct problem-solving strategy would correctly mention the relevant concepts when specifically asked to explain their response to a given mathematical problem.

In the preceding chapter, we described the collection of student explanatory texts on problem-solving within a graduate-level linear algebra course. In this chapter, we first describe our analysis of the set of student explanation text, where we show that there is a considerable similarity between the most frequent n-grams in the explanation text set and the expert-defined concept tags (Section 8.1) and then describe how they can be aligned using string-matching techniques (Section 8.2). We then present our proposed method called *Unsupervised Skill Tagging (UST)* to associate mathematical questions with concept tags based on student explanations (Section 8.3). Additionally, we analyze the efficacy of LLMs in tagging the questions (Section 8.4) and compare the performance of all the techniques on a test set (Section 8.5).

8.1 ANALYZING THE SET OF STUDENT EXPLANATION TEXT

The dataset we collected includes textual explanations from 30 masters students detailing their problem-solving strategies for 10 questions in a graduate-level linear algebra course. For each question, an average of 23 explanations were gathered (minimum 15 and maximum 28). In this section, we describe the analysis of the set of student explanation text using natural language processing techniques, to determine if the text set contains features with sufficient predictive power for determining the correct tags for the question. To understand the nature of the student explanations that were gathered, refer to Table 8.1, which presents sample explanations from four students for solving one of the 10 problems. We will now outline the processing steps applied to

Table 8.1: Student explanations on solving one of the problems.

Question	Let A be the matrix for the orthogonal projection of vectors in \mathbb{R}^3 onto the line spanned by the vector $[-1 0 1]$. Then what is $\text{rank}(A^7)$?
Expln 1	It is projecting \mathbb{R}^3 to a line, so $\text{rank}(A) = 1$. And the projection is orthogonal, which means it is just a rotation, not changing magnitude of vector, so the vector in the line will be projected there itself. $\text{rank}(A^7) = 1$
Expln 2	A will be a idempotent matrix and have full rank since A will be identity
Expln 3	The projection matrix is idempotent $A^2 = A$, Hence $A^7 = A$. The rank of the projection matrix is the same as that of input space, in our case, the line spanned by $[-1 0 1]$. Projection does not change the span of the input space, and the matrix is idempotent.
Expln 4	Projection matrices are idempotent. Here the matrix is projecting onto a vector space basis having dimension 2. Hence rank of A will be 2 which is the dimension of its range space. Since projection matrices are idempotent $A^7 = A$ itself and hence $\text{rank}(A^7) = \text{rank}(A) = 2$.

the text data set.

Pre-processing: The set of student explanation text was cleaned by performing the basic text preprocessing steps such as removing punctuation, extra spaces and digits, conversion to lower case and removal of stop words. Tokenization and lemmatization were then performed. The text preprocessing steps were performed using NLTK [93].

Extracting the most frequent unigrams and bigrams: For each question, the most frequent unigrams and bigrams (referenced as *n-grams* in the rest of the chapter) from all the student explanations were extracted. The 10 most frequent n-grams for 3 questions are provided in Table 8.2. See Appendix C.2 for the list of frequent n-grams for all the 10 questions.

It can be observed from the table that the most frequent n-grams in the explanation text have significant similarity with the expert-defined labels of the question. This suggests that we may determine the appropriate tags for a question by linking the most commonly occurring unigrams in the explanation text with the existing concept labels in the skill model. In the next section, we explain the proposed method to match the most frequent n-grams in the text with the concept labels in the *master tag-list*.

8.2 MATCHING FREQUENT N-GRAMS WITH CONCEPT LABELS

The key idea is to match the frequently occurring n-grams in the explanation text set of the question with the concept labels in the *master tag-list* using string-matching techniques. For each n-gram, matching is performed against each of the labels in the

Table 8.2: The 10 most frequent n-grams in the student explanations for 3 questions along with the *expert-defined concept labels*.

Question	10 most frequent uni/bi-grams	expert-defined tags
Let A be the matrix for the orthogonal projection of vectors in \mathbb{R}^3 onto the line spanned by the vector $[-101]$. Then what is $\text{rank}(A^7)$?	matrix, projection, rank, projection matrix, vector, orthogonal, idempotent, orthogonal projection, span, space	Projection, Rank, Range-Space, Idempotent-Matrix, Dimension
Let v_1, v_2, v_3, v_4 be the basis of a vectorspace $S_1 = \text{span}(v_1 + v_3, v_4)$ and $S = \text{span}(v_3, v_2 + v_4)$. The dimension of $S_1 + S$ is:	vector, subspace, dims, dimension, intersection, basis, dims dims, span, independent, vector space	Subspace Addition, Subspace Intersection, Dimension, Linear Independence
The eigenvalues of a matrix M are $0, 1, 1, 2, 3, 5, 5$. Is M invertible?	invertible, matrix, eigen, value, eigen value, zero, matrix invertible, eigenvalue, one, rank	Eigenvalue, Eigenvector, Null-Space, Invertibility

master tag-list and a score is assigned based on the degree of similarity. Finally, the labels having similarity score greater than a threshold are selected as the tags for the question. The string-matching based similarity computation is explained below.

A string s_1 is said to have an *exact match* in another string s_2 if the words in s_1 are present as exact continuous words in s_2 . For instance, 'norm' doesn't have an exact match in 'orthonormal' whereas 'nullity' has an exact match in 'rank nullity theorem'. This matching is captured in the "contains" function defined below.

$$\text{contains}(cLabel, fGram) = \begin{cases} \text{True} & \text{if there is an exact match of } fGram \text{ in } cLabel \\ \text{False} & \text{otherwise} \end{cases} \quad (8.1)$$

A concept-label "cLabel" in the *master tag-list* is matched with a frequent n-gram "fGram" if fGram has an exact match in cLabel or if the Levenshtein distance between fGram and cLabel, denoted as $\text{levD}(cLabel, fGram)$, is less than the threshold $\gamma_{\text{dist}} > 0$ (Equation (8.2)). Levenshtein distance between two strings is defined as the minimum number of single-character edits (insertions, deletions or substitutions) required to change one string into the other. For instance, the Levenshtein distance between 'test' and 'tent' is 1, while that between 'test' and 'ten' is 2.

$$\text{match}(fGram, cLabel) = \begin{cases} \text{True} & \text{if } \text{contains}(cLabel, fGram) \text{ is True or} \\ & \text{levD}(cLabel, fGram) < \gamma_{\text{dist}} \\ \text{False} & \text{otherwise} \end{cases} \quad (8.2)$$

The scoring function to perform the matching is defined as follows:

$$\text{score}(\text{fGram}, \text{cLabel}) = \begin{cases} \text{lexical-count}(\text{fGram}) & \text{if } \text{match}(\text{cLabel}, \text{fGram}) \text{ is True} \\ \text{lexical-count}(\text{cLabel}) \\ 0 & \text{otherwise} \end{cases} \quad (8.3)$$

$\text{lexical-count}(\text{term})$ is the number of words in term. For each question, a score is computed for every concept label, cLabel , in the *master tag-list* against each of the frequent n-grams, fGram , using the scoring function score .

$$\text{score}(\text{cLabel}) = \sum_{\text{fGram}} \text{score}(\text{fGram}, \text{cLabel}) \quad (8.4)$$

$$y_{\text{cLabel}} = \begin{cases} 1 & \text{if } \text{score}(\text{cLabel}) \geq \gamma_{\text{score}} \\ 0 & \text{otherwise} \end{cases} \quad (8.5)$$

The question is tagged with a concept label cLabel if y_{cLabel} equals 1 (Equation (8.5)), i.e., the concept labels in the master tag-list with a score greater than or equal to the $\gamma_{\text{score}} > 0$ are determined to be the tags associated with that question. The values of $\gamma_{\text{dist}} > 0$ and $\gamma_{\text{score}} > 0$ are user-defined and can be determined from data.

In the next section, we formalize this approach and propose *Unsupervised Skill Tagging* (UST), a technique for tagging mathematical questions by linking the most frequent unigrams in its explanation text set with an existing list of concept labels.

8.3 UNSUPERVISED SKILL TAGGING (UST)

The UST algorithm is based on the idea that explanations by students with the right problem-solving approach contains the correct concepts needed to solve the problem. The procedure has been illustrated in Algorithm 8.1.

The set of student explanations for the question to be tagged, are collected and cleaned by applying the standard pre-processing techniques such as the removal of non-alphabetic characters, converting to lowercase, discarding stop words, tokenization, and lemmatization. The processed set of explanation text, explnSet along with the tagLst , that contains the concept-labels for tagging, are passed to the UST() algorithm. The unigrams and bigrams in explnSet with frequency of occurrence at least thr are added to fGramSet . Each fGram in fGramSet is scored against every label in tagLst

Algorithm 8.1: UST

```

Input: explnSet, tagLst
nGramSet  $\leftarrow \{ \text{unigrams(explnSet)}, \text{bigrams(explnSet)} \}$ 
fGramSet  $\leftarrow \{ \text{ngram} \in \text{nGramSet} \mid \text{freq(ngram)} \geq \text{thr} \}$ 
score[lab] := 0  $\forall \text{lab} \in \text{tagLst}$ 
for fGram  $\in \text{fGramSet}$  do
    for cLabel  $\in \text{tagLst}$  do
        score[cLabel] := score[cLabel] + score(fGram, cLabel) (using (8.3))
    tagQn = { }
    for cLabel  $\in \text{tagLst}$  do
        if score[cLabel]  $\geq \gamma_{\text{score}}$  then
            tagQn  $\leftarrow \text{tagQn} \cup \text{cLabel}$ 
return tagQn

```

by computing $\text{score}(\text{fGram}, \text{cLabel})$ using Equation (8.3). Finally the concept-labels with a total score of at least γ_{score} are assigned as the tags for the question.

In Section 8.5, we describe the result of applying UST on a designated test dataset. We compare the performance of UST against *manual tagging* and tags generated using LLMs, as explained in the next section.

8.4 TAGGING USING LLMS

Similar to Li et al. [86], we attempted automatic tagging of the questions using *Zero shot chain-of-thought (CoT)* prompting. A zero-shot prompt directly instructs the model to perform a task without any additional examples or demonstrations. Large-scale, extensive training on diverse and vast datasets makes the LLMs capable of performing various tasks in a ‘zero-shot’ manner. Chain-of-thought (CoT) prompting, introduced by Wei et al. ([94]), has demonstrated improved performance on complicated tasks by enabling complex reasoning capabilities through intermediate reasoning steps.

Li et al. employed CoT prompting of the LLMs to determine whether a given knowledge-component matches the specified question in the form of an Yes/No response and also provide the reason for the judgement. However, in our work, we use CoT prompting to get the LLMs to determine the suitable tags for a question from the list of all tags in the *master tag-list*. The template of the prompt for tagging is given in Table 8.3. We used three popular LLMs for our tagging task - GPT-4o [95] (via chatbot

Table 8.3: Prompt given to LLMs to perform zero-shot CoT tagging.

<p>Q: You are an expert in linear algebra who performs tagging of linear algebra questions with labels of concepts that are used to solve them. Each question is associated with one or more tags exclusively from the following comma separated list: [<i>tags from master tag-list inserted here</i>] Assign the labels for the following question exclusively from the above list: [<i>the question text inserted here</i>]</p> <p>A: Let's think step by step.</p>
--

chatGPT), Gemini 1.0 [96] and Meta Llama 3 8B [97] (via <https://www.llama2.ai/>). The experiment results are discussed in the next section.

8.5 RESULTS

UST is an unsupervised tagging method. However, we make a train-test split so as to choose the values of the thresholds used for selecting the frequent n-grams and the scoring function. Hence the dataset was randomly split into a training-test set with the training set consisting of 6 questions and the test set with 4 questions.

8.5.1 UST and Manual Tagging

UST was used for tagging each question in the test set by including explanations of:

SETTING 1: all students

SETTING 2: students who scored non-zero marks (partially correct answers) for the question in regular quiz

SETTING 3: students who scored full marks for the question in regular quiz

For each of these settings, the values of the frequency threshold for frequent n-gram extraction, γ_{dist} and γ_{score} were selected using parameter tuning on the training set. The best parameter values discovered under all the three settings were the same: frequency threshold for n-gram discovery as one-tenth of the total number of student explanations for that question, $\gamma_{dist} = 1$ and $\gamma_{score} = 1$.

UST was applied on the test dataset with the chosen parameters. As each question is tagged with multiple concept labels, this is an instance of multi-label classification and therefore the precision, recall and F_1 -score of our method were computed for the test

Table 8.4: Evaluation metrics computed for the 3 settings under a) UST and b) Manual tagging.

Metric	UST			Manual Tagging		
	Setting 1	Setting 2	Setting 3	Setting 1	Setting 2	Setting 3
Precision	0.713	0.742	0.775	0.895	0.742	0.742
Recall	0.938	0.875	0.792	0.77	0.707	0.707
F1-score	0.810	0.803	0.783	0.827	0.724	0.724

set as follows:

$$\text{precision} = \frac{1}{n} \sum_{i=1}^n \frac{|y_i \cap y'_i|}{|y'_i|} \quad (8.6)$$

$$\text{recall} = \frac{1}{n} \sum_{i=1}^n \frac{|y_i \cap y'_i|}{|y_i|} \quad (8.7)$$

$$F_1\text{score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (8.8)$$

n: number of questions

y_i : expert defined concept labels for question i

y'_i : labels generated by UST for question i

The precision, recall and F_1 score for our method in each of the three cases is shown in Table 8.4. We have also compared the performance of UST against manual tagging where the question tags were chosen to be the ones selected by at least half of the students under each setting. See Appendix C.3 for the list of tags obtained through *UST* and *manual tagging*.

It can be observed that for both the methods, the highest F_1 score is obtained when the explanations of all the students were considered. In this setting, Manual tagging performs slightly better than UST in terms of the F_1 score. Note that for manual tagging a small subset of tags in the *master tag-list* were presented to the students which also included all of the expert-defined labels, which contributed to the higher precision value and subsequently an improved F_1 score. UST on the other hand utilized the entire *master tag-list* and produced a better recall than manual tagging.

Metric	GPT-4o	Gemini 1.0	Llama 3
Precision	0.75	0.75	0.546
Recall	0.708	0.375	0.562
F1-score	0.728	0.50	0.554

Table 8.5: Evaluation metrics computed for GPT-4o, Gemini 1.0 and Llama 3.

Note that while precision improves for UST from Setting 1 to 2 and 3, a drop in recall is simultaneously observed. This might be due to the students who correctly solved a problem listing mostly the right concept labels while missing out on a few other relevant ones. The students who have incorrectly solved a problem but with the right problem-solving strategy might still list out all the required concepts, while the ones with an incorrect strategy would list the irrelevant concepts in their explanations. Thus including these explanations causes UST to lose on precision but gain on recall.

8.5.2 LLM based tagging

The evaluation metrics for the tagging performed by the 3 LLMs are given in Table 8.5. It can be observed from the table that GPT-4o significantly out-performs the other two models in terms of Recall as well as F₁ score. CoT prompting gets GPT-4o to reflect on the problem-solving procedure, which enables the model to identify most of the required concepts for solving the question. Appendix C.4 shows the output produced by GPT-4o for one of the tagging prompts.

Both of the other two models lacked the ability to engage in significant reasoning about the problem-solving technique, as evidenced by their tagging performance. This highlights the dominance of GPT-4o in solving mathematical problems and tasks that require reasoning. The results demonstrate that GPT-4o is a promising tool for concept-tagging of math questions. See Appendix C.3 for the tags generated by the three LLMs for each of the 10 questions.

8.6 DISCUSSION

Unsupervised Skill Tagging (UST) was proposed based on the observation that the frequent unigrams and bigrams in the student explanations on solving a problem had significant similarity with the expert-defined concept-tags for that question. In spite of its simplicity, the scoring function based on Levenshtein distance and string matching,

demonstrated good performance on the test set. Due to the unavailability of other publicly accessible datasets containing student explanations in the field of advanced mathematics, we were unable to assess the extent to which UST may be applied to various other scenarios.

It is also noteworthy that, in comparison to all other approaches, UST had the highest recall. This can be attributed to the collection of student explanations that included all the pertinent concepts for problem-solving, and the UST algorithm accurately associating them with the appropriate concept labels in the master tag-list. To improve precision, one possible method is to enhance the selection of tags by UST by eliminating broader tags like ‘eigenvalue’ and ‘vector’ when more precise tags like ‘eigenvalue-matrix-powers’ or ‘span’ are included in the shortlist. This can be accomplished by utilizing a tool such as a concept map, that represents different types of relations between the concepts.

GPT-4o with zero shot chain-of-thought prompting yielded good tagging results, indicating its potential as a viable tool. Li et al. [86] used LLMs for tagging elementary math questions with 12 KCs under zero-shot as well as few-shot settings with CoT. It will be interesting to study how these models can be leveraged to perform tagging when the number of tags in the skill model is high, as in the case of large, complex learning systems.

8.7 CONCLUSION

In this part of our study, we examined the problem of automatic generation of concept/skill tags for assessment items in a higher education math course. Specifically, our study focused on the task of assigning pre-defined concept labels to math questions. In order to accurately identify all the pertinent concept labels for math questions, it is necessary to integrate the problem-solving technique into the tagging procedure. Therefore we investigated if learner-sourced explanations on problem solving can be used for designing efficient tagging techniques for advanced math problems. The key idea in this work is that students with the correct problem-solving approach would describe the necessary concepts to solve a problem when prompted to explain their problem-solving strategy. In this context, we conducted a study in a graduate level linear algebra course to understand if student explanations to solving mathematical problems can be employed to generate concept tags associated with those questions.

It was observed that the most frequent unigrams and bigrams in the processed explanation text provided by the students had good similarity with the important concepts used in solving the problem. We then developed a technique called *Unsupervised Skill Tagging (UST)* to tag questions with labels from a master tag-list, based on the most frequent n-grams in the set of explanations for each problem. UST demonstrated good performance on our dataset comprising the student explanations, which reinforces the idea that the student explanations indeed describe the right concepts required to solve a problem, which is exploited by *UST* to generate the correct question tags.

CONCLUSION AND FUTURE DIRECTIONS

In this thesis, we proposed novel machine learning based models and techniques for mastering a curriculum using adaptive sequencing of *activities* and *content*. We considered tutoring under two settings. The first one is the *learning by doing* setting where concepts are learned by doing activities or solving problems related to the concept. Here the challenge for the tutoring algorithm is to present at each time step, activities based on concepts that the student learns at a faster pace. In order to prevent excessive or insufficient practice of a concept, the algorithm has to also determine when a student has *truly mastered* a concept. This facilitates improving the student's learning outcomes and simultaneously sustains their motivation. In the first part of this thesis, we study this problem, also referred to as *personalized concept-sequencing*, and proposed *CurriculumTutor*, a technique that combines KL-UCB, an upper confidence bound Bernoulli bandit algorithm, with CUSUM, a log-likelihood based statistical test for change-point detection. Here activity selection at each time instant is performed by the bandit model where the arms correspond to the set of concepts in the ZPD and the rewards are computed based on student responses to activities related to the concepts. The change-point detection algorithm separately tracks the mastery of each concept. Simulation results show that *CurriculumTutor* significantly out-performs prior approaches in the benchmark domain of two operand addition up to a maximum of four digits.

The second tutoring setting that we studied involves learning through engagement with different learning resources such as videos, text-book sections and illustrations. In this context, the second part of the thesis studied the problem of generating a *personalized optimal content sequence* to be presented to a student in order to master a concept within a given time horizon. In this thesis, we modeled this problem as an episodic, fixed time horizon Markov Decision Process (MDP) based on generalized linear bandits. We have also proposed *OPT-CSEQ*, a reinforcement learning based planning technique that uses an upper confidence bound bandit algorithm, to learn an optimal personalized sequence of content. We have demonstrated the effectiveness of *OPT-CSEQ* via numerical experiments and have also presented the regret analysis of the same.

In the final part of the thesis, we studied the problem of tagging math questions with concept labels, enabling their utilization by tutoring algorithms for practice or assessments. We claim that generating accurate concept tags would require the tagging algorithm to also consider the problem solving logic into the tagging process. In this context, we conducted a study in a graduate level linear algebra course to examine if students with the correct problem solving approach mentions the right concepts required to solve the problem, when asked to explain their problem solving strategy. Based on our observations, we have proposed a method called *Unsupervised Skill Tagging (UST)*, to identify the concept-tags associated with a question item by processing the student explanations. *UST* extracts frequent unigrams and bigrams from the text and uses a scoring function based on exact word matching and Levenshtein distance to match them with a master list of concept tags.

In the next section, we describe the potential future directions for research for each of the problems that we have studied.

9.1 FUTURE DIRECTIONS

9.1.1 *Adaptive sequencing of concepts*

In Chapters 3 and 4, we proposed a framework for sequencing of concepts where we assumed that (i) a concept once mastered is never forgotten (ii) each activity/problem is associated with exactly one concept, and (iii) all activities/problems are of the same difficulty level. However these assumptions are very rudimentary and not always satisfied in reality. Therefore future work on this problem involves extending the proposed framework to incorporate additional elements such as accounting for forgetting of a learned concept, considering partial credit scores for problems and accommodating activities associated with more than once concept and with varying levels of difficulty. An important direction for future research involves testing and validating CurriculumTutor in real educational settings as described in Section 4.5.1. Potential extensions to this framework include creating an exercise recommendation system that assists students in preparing for competitive tests or exams that adhere to a specific curriculum. The system must offer students a progressive series of questions that systematically increase in difficulty and encompass multiple interrelated concepts, thereby effectively preparing them for the test.

It will also be interesting to study if activity selection can be modeled as an *improving multi-armed bandit problem* [98] where the reward obtained from an arm increases with the number of times it is pulled.

9.1.2 Optimal adaptive sequencing of content

In Chapter 6, we demonstrated the effectiveness of OPT-CSEQ via numerical experiments. A future course of study involves deploying and testing the algorithm with real students as discussed in Section 6.5.1. Another avenue for future work includes extending the technique for the scenario where multiple related concepts have to be simultaneously mastered by the student.

9.1.3 Concept tagging of mathematical questions

In Chapter 8, we studied the performance of *Unsupervised Skill Tagging (UST)* on linear algebra problems from a graduate level course. One potential direction for future research is to assess the efficacy of *UST* in (i) other advanced mathematical subjects, such as probability and calculus, and (ii) classes with more number of students.

UST employs a simple scoring function based on string matching to associate the most commonly occurring unigrams and bigrams in the explanation text with the labels in the master tag list. In spite of its simplicity, the scoring function demonstrates good performance on the test set. It would be interesting to study the tagging performance of *UST* when other techniques such as string alignment is used to compute the similarity scores. Another avenue of future work is to study if the precision of *UST* can be improved by incorporating additional information such as a concept map that models the relations between various concepts. This can help to further refine the tags selected by *UST* based on the scoring function, by eliminating broader tags when more specific tags are present in the list.

In our experiments on LLM based tagging, we observed that models such as GPT-4o with zero shot chain-of-thought prompting produced good results on our dataset. As the models continue to evolve, they become more adept at a variety of tasks, including solving mathematical problems and reasoning. Hence it will be interesting to study how large models can be employed to develop efficient tagging techniques for mathematical questions.

Part IV

APPENDIX

A

DETAILS OF EXPERIMENTS IN PERSONALIZED CONCEPT SEQUENCING

A.1 PARAMETERS FOR THE STUDENT MODELS

The parameters for the various student models for each knowledge component kc in our experiment have been given in Table A.1.

Table A.1: Parameters for BKT, LFA and PFA student models used in our experiment.

kc	p_{T_{low}}	p_{T_{high}}	p_g	p_s	β_{kc} for LFA	β_{kc} for PFA	η_{kc} for PFA
1	0.3	0.3	0.2	0.05	0.2	0.2	0.05
2	0.01	0.3	0.2	0.05	0.2	0.2	0.05
3	0.01	0.3	0.1	0.05	0.15	0.15	0.03
4	0.01	0.3	0.1	0.05	0.15	0.15	0.03
5	0.01	0.2	0.1	0.075	0.15	0.15	0.03
6	0.01	0.2	0.1	0.075	0.15	0.15	0.03
7	0.01	0.3	0.05	0.075	0.1	0.1	0.02
8	0.01	0.1	0.05	0.075	0.05	0.1	0.02
9	0.01	0.15	0.05	0.1	0.1	0.1	0.02
10	0.01	0.15	0.05	0.1	0.1	0.1	0.02
11	0.01	0.1	0.02	0.1	0.05	0.05	0.01
12	0.01	0.2	0.02	0.1	0.05	0.05	0.01
13	0.01	0.1	0.02	0.15	0.05	0.05	0.01
14	0.01	0.1	0.02	0.15	0.05	0.05	0.01

Other parameters

- LFA: $\alpha = 0.7, \gamma = 1.2$
- PFA: $\alpha = 1.3, \gamma = 1.5$

B

EXPERIMENTAL DETAILS FOR CONTENT SEQUENCING

B.1 SETTING I (WITHOUT CONTEXT)

The Ω and other parameters used for the experiment in **Setting 1 (without context)**, the results of which are presented in the main paper, are given below.

$$\Omega = \begin{pmatrix} 1.0 & 0.8 & 0.7 & 0.3 & 0.4 & 0.2 & 0.1 & 0.2 & 0.1 & 1.5 \\ 0.1 & 0.2 & 0.2 & 1.0 & 0.7 & 0.3 & 0.3 & 0.4 & 0.4 & 0.4 \\ 0.3 & 0.1 & 0.2 & 0.7 & 0.6 & 0.2 & 0.2 & 0.3 & 0.3 & 0.3 \\ 0.2 & 0.3 & 0.1 & 0.5 & 0.7 & 0.2 & 0.2 & 0.3 & 0.3 & 0.3 \\ 0.2 & 0.2 & 0.2 & 0.1 & 0.3 & 0.7 & 0.6 & 1.0 & 0.7 & 0.4 \\ 0.3 & 0.3 & 0.3 & 0.2 & 0.1 & 0.7 & 0.8 & 0.7 & 0.6 & 0.3 \\ 0.2 & 0.3 & 0.2 & 0.2 & 0.3 & 0.1 & 0.4 & 0.5 & 0.6 & 1.0 \\ 0.2 & 0.3 & 0.2 & 0.4 & 0.3 & 0.3 & 0.1 & 0.6 & 0.7 & 0.8 \\ 0.2 & 0.2 & 0.3 & 0.4 & 0.2 & 1.0 & 0.7 & 0.1 & 0.4 & 0.7 \\ 0.3 & 0.2 & 0.2 & 0.3 & 0.2 & 0.8 & 0.6 & 0.3 & 0.1 & 0.8 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.1 \end{pmatrix}$$

Parameters used: $k = 10$, $H = 5$, $T = 2000$, $\kappa = 0.9$, $\alpha = 2.5$, $Z = 0.025$, $\gamma = 3$, $D = 3.5$, $C = 4$, $T_0 = 150$

B.2 ADDITIONAL EXPERIMENTS FOR SETTING I

We also performed additional experiments for $k = 10$ contents and horizon $H = 5$ in **Setting 1** with different values of Ω . The results, averaged over 5 runs, are provided below. Note that in the line charts (Figures B.2, B.4 and B.6), the values have been smoothed using a window of size 150. In all these experiments, *OPT-CSEQ* was found to outperform both Greedy and Random policies (see Figures B.1-B.6).

Run 1

$$\Omega = \begin{pmatrix} 1.0 & 0.3 & 0.2 & 1.2 & 0.4 & 0.5 & 1.0 & 0.3 & 0.2 & 0.4 \\ 0.1 & 0.5 & 0.6 & 0.3 & 0.8 & 0.7 & 0.3 & 0.4 & 0.6 & 0.2 \\ 0.5 & 0.1 & 0.6 & 0.5 & 0.4 & 0.3 & 0.6 & 0.7 & 0.7 & 0.8 \\ 0.6 & 1.0 & 0.1 & 0.3 & 0.5 & 0.4 & 0.2 & 0.6 & 0.4 & 0.7 \\ 0.3 & 0.2 & 0.3 & 0.1 & 0.2 & 0.5 & 0.5 & 0.2 & 0.4 & 0.3 \\ 0.4 & 0.6 & 1.0 & 0.7 & 0.1 & 0.3 & 0.4 & 0.5 & 0.8 & 0.4 \\ 0.3 & 0.5 & 0.4 & 0.5 & 0.6 & 0.1 & 0.6 & 0.7 & 0.3 & 0.2 \\ 0.4 & 0.5 & 0.3 & 0.4 & 0.7 & 0.5 & 0.1 & 0.6 & 1.0 & 0.8 \\ 0.8 & 0.3 & 0.6 & 0.7 & 0.5 & 0.3 & 0.4 & 0.1 & 0.2 & 0.4 \\ 0.2 & 0.5 & 0.3 & 0.6 & 1.0 & 0.7 & 0.5 & 0.8 & 0.1 & 0.3 \\ 0.3 & 0.4 & 0.6 & 0.5 & 0.2 & 0.4 & 0.3 & 0.6 & 0.7 & 0.1 \end{pmatrix}$$

Parameters used: $k = 10$, $H = 5$, $T = 2000$, $\kappa = 0.9$, $\alpha = 2.5$, $Z = 0.025$, $\gamma = 3$, $D = 3.5$, $C = 4$, $T_0 = 150$

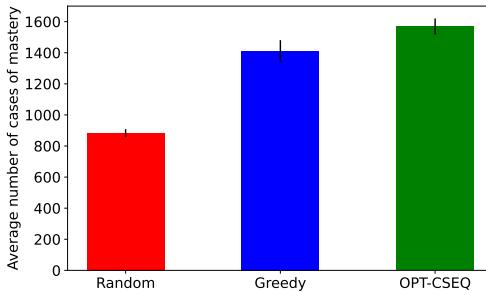


Figure B.1: **Setting I (Without context)**: Comparing the average number of cases where mastery was attained in Run 1.

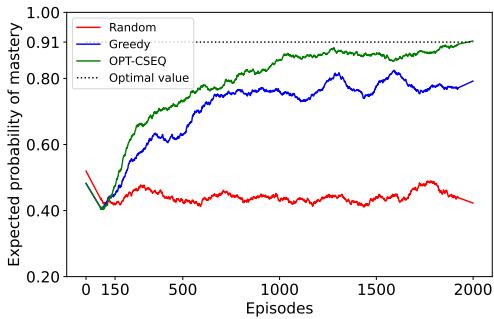


Figure B.2: **Setting I (Without context)**: The average probability of mastery for the policies computed at each episode in Run 1.

Run 2

$$\Omega = \begin{pmatrix} 0.5 & 1.0 & 0.4 & 0.6 & 1.5 & 0.5 & 0.6 & 1.0 & 0.3 & 0.4 \\ 0.1 & 0.5 & 0.3 & 0.2 & 0.4 & 1.5 & 0.6 & 0.9 & 0.5 & 1.0 \\ 0.6 & 0.1 & 0.3 & 0.4 & 0.7 & 0.2 & 0.5 & 0.6 & 0.2 & 0.3 \\ 0.3 & 0.4 & 0.1 & 0.6 & 0.2 & 0.7 & 0.5 & 0.8 & 0.4 & 0.8 \\ 0.5 & 0.7 & 0.6 & 0.1 & 0.4 & 0.3 & 0.8 & 0.3 & 0.7 & 0.9 \\ 0.4 & 0.6 & 0.5 & 0.2 & 0.1 & 0.4 & 0.3 & 0.7 & 1.0 & 0.8 \\ 0.7 & 0.4 & 1.5 & 0.8 & 0.9 & 0.1 & 0.5 & 0.6 & 0.7 & 0.6 \\ 0.8 & 0.5 & 0.4 & 0.7 & 0.6 & 0.7 & 0.1 & 0.3 & 0.5 & 0.9 \\ 0.5 & 1.25 & 0.7 & 0.5 & 0.3 & 0.4 & 0.8 & 0.1 & 0.6 & 1.0 \\ 1.5 & 0.4 & 0.6 & 0.8 & 0.5 & 0.2 & 0.7 & 0.9 & 0.1 & 0.7 \\ 0.6 & 0.3 & 0.5 & 0.7 & 0.9 & 0.8 & 0.3 & 0.4 & 0.8 & 0.1 \end{pmatrix}$$

Parameters used: $k = 10$, $H = 5$, $T = 2000$, $\kappa = 0.9$, $\alpha = 3.2$, $Z = 0.025$, $\gamma = 3$, $D = 5$, $C = 4$, $T_0 = 150$

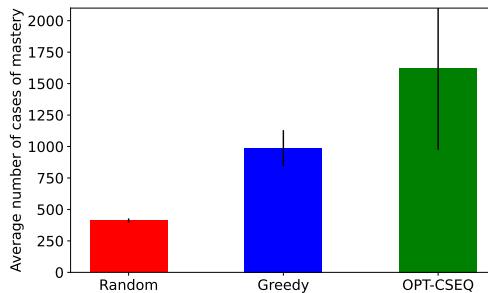


Figure B.3: **Setting I (Without context)**: Comparing the average number of cases where mastery was attained in Run 2.

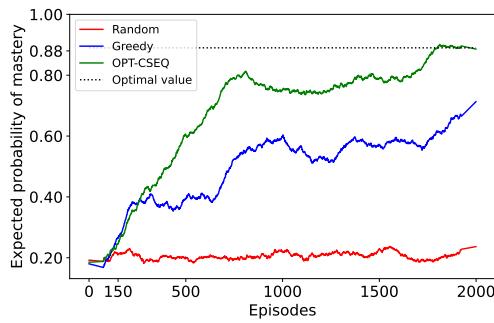


Figure B.4: **Setting I (Without context)**: The average probability of mastery for the policies computed at each episode in Run 2.

Run 3

$$\Omega = \begin{pmatrix} 0.3 & 0.5 & 0.6 & 1.2 & 0.4 & 0.3 & 0.7 & 0.6 & 0.5 & 1.0 \\ 0.1 & 0.3 & 0.5 & 0.4 & 0.8 & 0.6 & 0.5 & 0.4 & 0.6 & 0.7 \\ 0.5 & 0.1 & 0.3 & 0.6 & 0.4 & 0.7 & 0.6 & 0.7 & 0.5 & 0.4 \\ 0.3 & 0.4 & 0.1 & 0.5 & 0.2 & 0.5 & 0.4 & 0.3 & 0.6 & 0.4 \\ 0.4 & 0.7 & 0.5 & 0.1 & 0.4 & 0.3 & 0.2 & 0.6 & 0.5 & 0.3 \\ 0.4 & 0.6 & 0.3 & 0.5 & 0.1 & 0.7 & 0.5 & 0.4 & 0.2 & 0.3 \\ 0.3 & 0.5 & 0.4 & 0.6 & 0.7 & 0.1 & 0.3 & 0.5 & 0.4 & 0.6 \\ 0.6 & 0.4 & 0.7 & 0.3 & 0.5 & 1.0 & 0.1 & 0.3 & 0.4 & 0.6 \\ 0.5 & 0.3 & 0.6 & 0.4 & 0.7 & 0.2 & 1.0 & 0.1 & 0.4 & 0.5 \\ 0.5 & 0.6 & 1.2 & 0.4 & 0.3 & 0.8 & 0.7 & 1.0 & 0.1 & 0.2 \\ 0.4 & 0.5 & 0.6 & 0.4 & 0.4 & 0.2 & 0.5 & 0.3 & 1.0 & 0.1 \end{pmatrix}$$

Parameters used: $k = 10$, $H = 5$, $T = 2000$, $\kappa = 0.9$, $\alpha = 2.5$, $Z = 0.025$, $\gamma = 3$, $D = 3.5$, $C = 4$, $T_0 = 150$

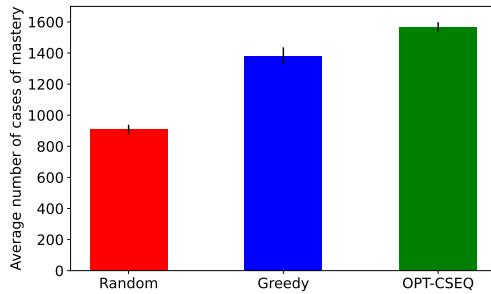


Figure B.5: **Setting I (Without context)**: Comparing the average number of cases where mastery was attained in Run 3.

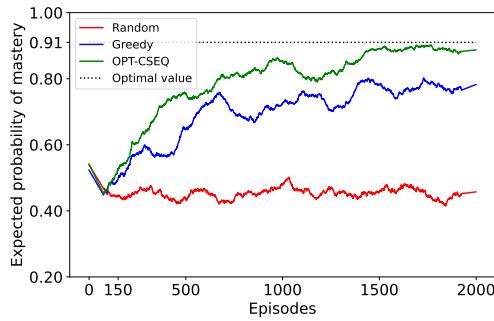


Figure B.6: **Setting I (Without context)**: The average probability of mastery for the policies computed at each episode in Run 3.

B.3 SETTING II (WITH CONTEXT)

The θ^* and other parameters used for the experiment in **Setting II (with context)**, the results of which are presented in the main paper, are given below. Note that Ω is fixed once the context is fixed, and the capability of OPT-CSEQ to find the optimal policy with respect to a given Ω has been demonstrated in **Setting I (without context)**. The experiment in **Setting II** is meant to demonstrate the working of the algorithm with 4 different student contexts.

$$\theta^*(\cdot, \cdot, 1) = \begin{pmatrix} 1.0 & 0.5 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.1 & 0.2 & 1.0 & 0.6 & 0.3 & 0.2 \\ 0.2 & 0.1 & 0.6 & 0.5 & 0.2 & 0.3 \\ 0.2 & 0.4 & 0.1 & 0.3 & 1.0 & 0.5 \\ 0.3 & 0.3 & 0.2 & 0.1 & 0.6 & 0.6 \\ 0.4 & 0.3 & 0.6 & 0.7 & 0.1 & 0.3 \\ 0.3 & 0.4 & 0.7 & 0.6 & 0.3 & 0.1 \end{pmatrix}$$

$$\theta^*(\cdot, \cdot, 2) = \begin{pmatrix} 0.5 & 1.0 & 0.3 & 0.3 & 0.2 & 0.2 \\ 0.1 & 0.3 & 0.4 & 0.6 & 0.2 & 0.3 \\ 0.2 & 0.1 & 0.5 & 1.0 & 0.3 & 0.2 \\ 0.2 & 0.3 & 0.1 & 0.3 & 0.5 & 0.6 \\ 0.3 & 0.4 & 0.4 & 0.1 & 0.6 & 1.0 \\ 0.4 & 0.3 & 0.5 & 0.6 & 0.1 & 0.3 \\ 0.3 & 0.4 & 0.6 & 0.5 & 0.3 & 0.1 \end{pmatrix}$$

$$\theta^*(\cdot, \cdot, 3) = \begin{pmatrix} 0.5 & 1.0 & 0.3 & 0.3 & 0.2 & 0.2 \\ 0.1 & 0.3 & 0.4 & 0.6 & 0.2 & 0.3 \\ 0.2 & 0.1 & 1.0 & 0.5 & 0.3 & 0.2 \\ 0.2 & 0.3 & 0.1 & 0.3 & 0.6 & 1.0 \\ 0.3 & 0.4 & 0.4 & 0.1 & 0.5 & 0.7 \\ 0.3 & 0.4 & 0.5 & 0.7 & 0.1 & 0.4 \\ 0.4 & 0.3 & 0.7 & 0.6 & 0.4 & 0.1 \end{pmatrix}$$

Parameters used: $m = 3, k = 6, H = 3, T = 2500, \kappa = 0.9, \alpha = 1.5, Z = 0.025, \gamma = 3, D = 3.5, C = 4, T_0 = 150$

B.3.1 Ω for various student contexts

Context: [1,1,0]

$$\Omega = \begin{bmatrix} 1.5 & 1.5 & 0.5 & 0.5 & 0.4 & 0.4 \\ 0.2 & 0.5 & 1.4 & 1.2 & 0.5 & 0.5 \\ 0.4 & 0.2 & 1.1 & 1.5 & 0.5 & 0.5 \\ 0.4 & 0.7 & 0.2 & 0.6 & 1.5 & 1.1 \\ 0.6 & 0.7 & 0.6 & 0.2 & 1.2 & 1.6 \\ 0.8 & 0.6 & 1.1 & 1.3 & 0.2 & 0.6 \\ 0.6 & 0.8 & 1.3 & 1.1 & 0.6 & 0.2 \end{bmatrix}$$

Context: [1,0,1]

$$\Omega = \begin{bmatrix} 1.5 & 1.5 & 0.5 & 0.5 & 0.4 & 0.4 \\ 0.2 & 0.5 & 1.4 & 1.2 & 0.5 & 0.5 \\ 0.4 & 0.2 & 1.6 & 1.0 & 0.5 & 0.5 \\ 0.4 & 0.7 & 0.2 & 0.6 & 1.6 & 1.5 \\ 0.6 & 0.7 & 0.6 & 0.2 & 1.1 & 1.3 \\ 0.7 & 0.7 & 1.1 & 1.4 & 0.2 & 0.7 \\ 0.7 & 0.7 & 1.4 & 1.2 & 0.7 & 0.2 \end{bmatrix}$$

Context: [0,1,1]

$$\Omega = \begin{bmatrix} 1.0 & 2.0 & 0.6 & 0.6 & 0.4 & 0.4 \\ 0.2 & 0.6 & 0.8 & 1.2 & 0.4 & 0.6 \\ 0.4 & 0.2 & 1.5 & 1.5 & 0.6 & 0.4 \\ 0.4 & 0.6 & 0.2 & 0.6 & 1.1 & 1.6 \\ 0.6 & 0.8 & 0.8 & 0.2 & 1.1 & 1.7 \\ 0.7 & 0.7 & 1.0 & 1.3 & 0.2 & 0.7 \\ 0.7 & 0.7 & 1.3 & 1.1 & 0.7 & 0.2 \end{bmatrix}$$

Context: [1,1,1]

$$\Omega = \begin{bmatrix} 2.0 & 2.5 & 0.8 & 0.8 & 0.6 & 0.6 \\ 0.3 & 0.8 & 1.8 & 1.8 & 0.7 & 0.8 \\ 0.6 & 0.3 & 2.1 & 2.0 & 0.8 & 0.7 \\ 0.6 & 1.0 & 0.3 & 0.9 & 2.1 & 2.1 \\ 0.9 & 1.1 & 1.0 & 0.3 & 1.7 & 2.3 \\ 1.1 & 1.0 & 1.6 & 2.0 & 0.3 & 1.0 \\ 1.0 & 1.1 & 2.0 & 1.7 & 1.0 & 0.3 \end{bmatrix}$$

C

CONCEPT TAGGING EXPERIMENTAL MATERIALS

C.1 QUIZ QUESTIONS USED IN OUR STUDY

The 10 questions used in our study are listed in Table C.1.

Table C.1: Quiz questions used in the supplementary quizzes along with the *expert-defined concept labels* for each question.

Q1	<p>Let $\mathbf{M} = \begin{bmatrix} 0.5 & -0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & -0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & -0.5 \\ -0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}$</p> <p>Then find $\ \mathbf{M}^{-1}\ ^2$ and $\text{trace}(\mathbf{M}^{-1})$</p>	<p>Orthonormal/Orthogonal Matrix, Trace, Transpose, Frobenius-Norm</p>
Q2	<p>Let \mathbf{A} be the matrix for the orthogonal projection of vectors in \mathbb{R}^3 onto the line spanned by the vector $[-101]$. Then what is $\text{rank}(\mathbf{A}^7)$?</p>	<p>Orthogonal-Projection, Rank, Range-Space, Idempotent-Matrix, Dimension</p>
Q3	<p>For $v, w \in \mathbb{R}^2$, $\langle v, w \rangle = v_1(v_2 + 2w_1) + w_2v_1$ is a valid inner product?</p>	<p>Symmetry of Inner Product, Linearity of Inner Product, Positive Definiteness of inner product</p>
Q4	<p>Let v_1, v_2, v_3, v_4 be the basis of a vectorspace $S_1 = \text{span}(v_1 + v_3, v_4)$ and $S = \text{span}(v_3, v_2 + v_4)$. The dimension of $S_1 + S$ is:</p>	<p>Subspace Addition, Subspace Intersection, Dimension, Linear Independence</p>
Q5	<p>The nullity of the outer product between the normal of the hyperplanes $x + y - z = 0$ and $2x - y - 3z = 0$ in \mathbb{C}^3 is:</p>	<p>Outer-Product, Rank, Dimension, Rank-Nullity-Theorem</p>
Q6	<p>Let \mathbf{M} be a rank-3 matrix such that $\text{R}(\mathbf{M}) = \text{N}(\mathbf{M})$, then $\text{rank}(\mathbf{M}^7)$ is _____ and the dimensions of the input and output vectorspaces of this linear map are _____ and _____.</p>	<p>Range-Space, Null-Space, Rank, Linear-Map-Composition, Rank-Nullity-Theorem, Nullity</p>
Q7	<p>Let $\mathbf{M} = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 1 & -2 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 \\ 1 & 0 & 4 & -1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}$</p> <p>The eigen values of $(\mathbf{M}^2)^T$ in ascending order are:</p>	<p>Eigenvalues-Triangular-Matrix, Eigenvalues-Matrix-Powers, Eigenvalues-Matrix-Transpose</p>
Q8	<p>The eigenvalues of a matrix \mathbf{M} are $0, 1, 1, 2, 3, 5, 5$. Is \mathbf{M} invertible?</p>	<p>Eigenvalue, Eigenvector, Null-Space, Invertibility</p>
Q9	<p>If the eigen values of \mathbf{M} are $-2, 0, 1, 1, 3, 4$ then the trace of $(3\mathbf{I} - 2\mathbf{M})^2$ is</p>	<p>Eigenvalue, Eigenvalues-Matrix-Powers, Trace</p>
Q10	<p>The differentiation of $f(x)$ in function space w.r.t x is a linear transformation in x, which of the following is an eigen function of this linear transformation?</p>	<p>Eigenvalue, Function-Space, Eigenvector, Derivative</p>

On performing random split, the questions Q₁, Q₂, Q₃, Q₇, Q₈, Q₉ were included in the training set while Q₄, Q₅, Q₆, Q₁₀ were part of the test set.

C.2 FREQUENT N-GRAMS FROM EXPLANATION TEXT OF ALL QUESTIONS

The 15 most frequent n-grams extracted from the cleaned student explanation text for each question have been provided in Table C.2, along with the *expert-defined concept labels* for that question. It can be observed that there is some degree of similarity between the frequent n-grams and the expert-defined tags.

Table C.2: The 15 most frequent n-grams in the student explanations along with the *expert-defined concept labels* of each question.

Q ₁	matrix, trace, inverse, norm, transpose, given, frobenius, given matrix, find, orthonormal, frobenius norm, sum, inverse matrix, hence, element	Orthonormal/Orthogonal Matrix, Trace, Transpose, Frobenius-Norm
Q ₂	matrix, projection, rank, projection matrix, vector, orthogonal, idempotent, orthogonal projection, span, space, line, idempotent matrix, hence, given, find	Orthogonal-Projection, Rank, Range-Space, Idempotent-Matrix, Dimension
Q ₃	product, inner, inner product, positive, valid, valid inner, check, property, positive definiteness, linearity, definiteness, vw, symmetry, satisfy, condition	Symmetry of Inner Product, Linearity of Inner Product, Positive Definiteness of inner product
Q ₄	vector, subspace, dims, dimension, intersection, basis, dims dims, span, independent, vector space, sum, space, set, addition, vector span	Subspace Addition, Subspace Intersection, Dimension, Linear Independence
Q ₅	nullity, product, rank, outer product, outer, vector, matrix, theorem, rank nullity, nullity theorem, find, space, normal, dimension, linear	Outer-Product, Rank, Dimension, Rank-Nullity-Theorem
Q ₆	rank, space, nullity, dimension, rank nullity, matrix, vector, input, theorem, range, range space, nullity theorem, null space, null, equal	Range-Space, Null-Space, Rank, Linear-Map-Composition, Rank-Nullity-Theorem, Nullity
Q ₇	matrix, value, eigen, eigen value, triangular, diagonal, triangular matrix, element, diagonal element, square, transpose, lower, given, eigenvalue, upper	Eigenvalues-Triangular-Matrix, Eigenvalues-Matrix-Powers, Eigenvalues-Matrix-Transpose
Q ₈	invertible, matrix, eigen, value, eigen value, zero, matrix invertible, eigen-value, one, rank, value zero, since, non, space, one eigen	Eigenvalue, Eigenvector, Null-Space, Invertibility
Q ₉	value, eigen, eigen value, matrix, trace, sum, square, sum eigen, im, given, eigenvalue, diagonal, lambda, trace sum, since	Eigenvalue, Eigenvalues-Matrix-Powers, Trace
Q ₁₀	eigen, function, derivative, value, space, vector, fx, eigen value, scaled, function space, eigen function, scaling, linear, transformation, operator	Eigenvalue, Function-Space, Eigenvector, Derivative

C.3 CONCEPT LABELS OBTAINED THROUGH DIFFERENT TECHNIQUES

The concept labels obtained for each question through *manual tagging* and *UST* are provided in Table C.3, and the tags obtained from GPT-4o, Gemini and Llama 3 are given in Table C.4. In *manual tagging*, the tags were chosen to be the ones selected by at least half of the students who attempted the question in the *regular quiz*.

Table C.3: Comparing the *expert defined concept labels* with those obtained from *manual tagging* and *UST*.

Question	Expert-defined tags	Manual Tagging	UST
Q1	Orthonormal/Orthogonal Matrix, Trace, Transpose, Frobenius-Norm	Frobenius-Norm, Invertibility, Orthonormal/Orthogonal Matrix, Trace	Orthonormal/Orthogonal Matrix, Trace, Transpose, Frobenius-Norm, Eigenvalues-Matrix-Transpose
Q2	Orthogonal-Projection, Rank, Range-Space, Idempotent-Matrix, Dimension	Rank, Span, Orthogonal Projection	Orthonormal/Orthogonal Matrix, Orthogonal Projection, Rank, Range Space, Idempotent-Matrix, Dimension, Span
Q3	Symmetry of Inner Product, Linearity of Inner Product, Positive Definiteness of Inner Product	Linearity of Inner Product, Positive Definiteness of Inner Product	Symmetry of Inner Product, Linearity of Inner Product, Positive Definiteness of Inner Product, Inner Product
Q4	Subspace Addition, Subspace Intersection, Dimension, Linear Independence	Subspace Addition, Subspace Intersection, Basis, Dimension	Dimension, Subspace Addition, Subspace Intersection, Span
Q5	Outer-Product, Rank, Dimension, Rank-Nullity-Theorem	Outer Product, Rank, Rank-Nullity-Theorem	Orthonormal/Orthogonal Matrix, Rank, Dimension, Outer Product, Rank-Nullity-Theorem, Null-Space, Linear-Map-Composition, Nullity, Linear Transformation
Q6	Range-Space, Rank-Nullity-Theorem, Null-Space, Linear-Map-Composition, Rank, Nullity	Range-Space, Null-Space, Rank, Rank-Nullity-Theorem, Linear-Map-Composition, Dimension	Rank, Range Space, Dimension, Rank-Nullity-Theorem, Null-Space, Linear-Map-Composition, Nullity
Q7	Eigenvalues-Triangular-Matrix, Eigenvalues-Matrix-Powers, Eigenvalues-Matrix-Transpose	Eigenvalues-Matrix-Powers	Eigenvalues-Triangular-Matrix, Transpose, Eigenvalue, Eigenvalues-Matrix-Transpose
Q8	Eigenvalue, Eigenvector, Null-Space, Invertibility	Eigenvalue, Invertibility	Rank, Eigenvalue, Invertibility, Range Space, Null Space, Eigenvector
Q9	Eigenvalue, Eigenvalues-Matrix-Powers, Trace	Eigenvalue, Eigenvalues-Matrix-Powers, Trace	Trace, Eigenvalue
Q10	Eigenvalue, Eigenvector, Function-space, Derivative	Derivative, Function-Space, Eigenvalue	Eigenvalue, Linear Transformation, Eigenvector, Function-Space, Derivative

Table C.4: Comparing the *expert defined concept labels* with those obtained from *GPT-4o*, *Gemini* and *llama 3*.

Qn	Expert-defined tags	GPT-4o	Gemini	Llama 3
Q1	Orthonormal/Orthogonal Matrix, Trace, Transpose, Frobenius-Norm	Invertibility, Frobenius-Norm, Trace	Invertibility, Frobenius-Norm, Trace	Invertibility, Eigenvalues-Matrix-Inverse, Trace, Frobenius-Norm
Q2	Orthogonal-Projection, Rank, Range-Space, Idempotent-Matrix, Dimension	Orthogonal-Projection, Rank, Idempotent-Matrix	Orthogonal-Projection, Rank	Orthogonal-Projection, Span, Rank
Q3	Symmetry of Inner Product, Linearity of Inner Product, Positive Definiteness of Inner Product	Linearity of Inner Product, Symmetry of Inner Product, Positive Definiteness of Inner Product	Linearity of Inner Product, Inner Product, Symmetry of Inner Product	Inner Product, Linearity of Inner Product, Symmetry of Inner Product
Q4	Subspace Addition, Subspace Intersection, Dimension, Linear Independence	Span, Dimension, Subspace Addition, Linear Independence	Span, Subspace Addition	Span, Subspace Addition, Dimension, Linear Independence
Q5	Outer-Product, Rank, Dimension, Rank-Nullity-Theorem	Outer-Product, Nullity, Rank-Nullity-Theorem	Outer Product, Nullity	Outer-Product, Null-Space, Span
Q6	Range-Space, Rank-Nullity-Theorem, Null-Space, Linear-Map-Composition, Rank, Nullity	Rank, Range-Space, Nullity, Rank-Nullity-Theorem, Linear-Map-Composition, Dimension	Rank, Nullity, Rank-Nullity-Theorem	Rank, Nullity, Range Space, Linear Transformation, Span
Q7	Eigenvalues-Triangular-Matrix, Eigenvalues-Matrix-Powers, Eigenvalues-Matrix-Transpose	Eigenvalues-Matrix-Powers, Eigenvalues-Matrix-Transpose	Transpose, Eigenvalue, Eigenvalues-Matrix-Powers	Eigenvalue, Transpose, Matrix Powers
Q8	Eigenvalue, Eigenvector, Null-Space, Invertibility	Eigenvalue, Invertibility	Invertibility	Eigenvalue, Invertibility
Q9	Eigenvalue, Eigenvalues-Matrix-Powers, Trace	Eigenvalues-Matrix-Powers, Trace	Trace, Eigenvalue	Eigenvalue, Trace, matrix powers
Q10	Eigenvalue, Eigenvector, Function-space, Derivative	Derivative, Function-Space, Linear Transformation, Eigenvalue	Function-Space, Derivative	Linear Transformation, Eigenvalue, Eigenfunction, Derivative, Function-Space, Inner Product

C.4 RESULT PRODUCED BY GPT-4O FOR ONE OF THE TAGGING PROMPTS

One of the tagging prompts to GPT-4o and the corresponding output are given in Figures C.1 and C.2 respectively.

Q: You are an expert in linear algebra who performs tagging of linear algebra questions with labels of concepts that are used to solve them. Each question is associated with one or more tags exclusively from the following comma separated list: orthonormal/orthogonal matrix, trace, transpose, frobenius norm, orthogonal projection, rank, range space, idempotent matrix, dimension, symmetry of inner product, linearity of inner product, positive definiteness of inner product, subspace addition, subspace intersection, linear independence, outer product, rank nullity theorem, null space, linear map composition, nullity, eigenvalues triangular matrix, eigenvalues matrix powers, eigenvalues matrix transpose, eigenvalue, eigenvector, invertibility, function space, derivative, inner product, linear transformation, nilpotent matrix, span. Assign the labels for the following question exclusively from the above list: Let M be a rank-3 matrix such that range space of M equals the nullity of M . Then find $\text{rank}(M^7)$ and the dimension of the input and output vectorspaces of this linear map.

A: Let's think step by step.

Figure C.1: One of the tagging prompts to GPT-4o.

From Figure C.2 it can be seen that the model is able to predict the right tags by reflecting on the problem-solving procedure. Even though the model has solved the second part of the question (computing $\text{rank}(M^7)$) incorrectly, the relevant question tags have been mostly identified correctly.



To solve this problem, we need to analyze the given conditions and use appropriate linear algebra concepts. The problem specifies that matrix M is of rank 3 and that its range space equals its nullity. Let's break it down:

Step-by-Step Solution:

1. Understand the Rank and Nullity Condition:

- The matrix M has a rank of 3.
- The rank-nullity theorem states that:

$$\text{rank}(M) + \text{nullity}(M) = n$$

where n is the number of columns of M .

2. Range Space Equals Nullity:

- Given that the range space of M equals the nullity of M , we have:

$$\text{rank}(M) = \text{nullity}(M)$$

Since the rank of M is 3, it follows that:

$$\text{nullity}(M) = 3$$

3. Determine the Dimension of M :

- Combining these, we find:

$$3(\text{rank}) + 3(\text{nullity}) = n$$

$$n = 6$$

- Therefore, M is a 6×6 matrix.

4. Rank of M^7 :

- If M is a rank-3 matrix, M^7 will also have the same rank as M because the rank of a power of a matrix is less than or equal to the rank of the original matrix. The rank cannot increase with powers, and the null space structure does not change under matrix powers. Therefore:

$$\text{rank}(M^7) = \text{rank}(M) = 3$$

5. Dimension of the Input and Output Vector Spaces:

- Since M is a 6×6 matrix, the dimension of both the input and output vector spaces is 6.

Summary:

- Rank of M^7 : 3
- Dimension of Input and Output Vector Spaces: 6

Tags:

- rank
- range space
- nullity
- rank nullity theorem
- linear map composition
- dimension

Figure C.2: GPT-4o output for the tagging prompt in Figure C.1.

BIBLIOGRAPHY

- [1] Francisco Niño-Rojas, Diana Lancheros-Cuesta, Martha Tatiana Pamela Jiménez-Valderrama, Gelys Mestre, and Sergio Gómez. "Systematic Review: Trends in Intelligent Tutoring Systems in Mathematics Teaching and Learning." In: *International Journal of Education in Mathematics, Science and Technology* 12.1 (2024), pp. 203–229.
- [2] Rodrigo Elias Francisco and Flávio de Oliveira Silva. "Intelligent Tutoring System for Computer Science Education and the Use of Artificial Intelligence: A Literature Review." In: *CSEDU* (1) (2022), pp. 338–345.
- [3] Patricia L Albacete and Kurt VanLehn. "The Conceptual Helper: An intelligent tutoring system for teaching fundamental physics concepts." In: *Intelligent Tutoring Systems: 5th International Conference, ITS 2000 Montréal, Canada, June 19–23, 2000 Proceedings* 5. Springer. 2000, pp. 564–573.
- [4] Martina A Rau, Joseph E Michaelis, and Natalie Fay. "Connection making between multiple graphical representations: A multi-methods approach for domain-specific grounding of an intelligent tutoring system for chemistry." In: *Computers & Education* 82 (2015), pp. 460–485.
- [5] Steven Ritter, John R Anderson, Kenneth R Koedinger, and Albert Corbett. "Cognitive Tutor: Applied research in mathematics education." In: *Psychonomic bulletin & review* 14.2 (2007), pp. 249–255.
- [6] Kenneth R Koedinger and Albert Corbett. "Cognitive tutors." In: *Smart machines in education* (2001), pp. 145–167.
- [7] Tom Murray. "An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art." In: *Authoring Tools for Advanced Technology Learning Environments: Toward Cost-Effective Adaptive, Interactive and Intelligent Educational Software* (2003), pp. 491–544.
- [8] Beverly Woolf and Pat Cunningham. "Building a community memory for intelligent tutoring systems." In: *Proceedings of the sixth National conference on Artificial intelligence-Volume 1.* 1987, pp. 82–87.
- [9] John R Anderson, Albert T Corbett, Kenneth R Koedinger, and Ray Pelletier. "Cognitive tutors: Lessons learned." In: *The journal of the learning sciences* 4.2 (1995), pp. 167–207.

- [10] Radek Pelánek. "Bayesian knowledge tracing, logistic models, and beyond: an overview of learner modeling techniques." In: *User modeling and user-adapted interaction* 27 (2017), pp. 313–350.
- [11] Ekaterina Kochmar, Dung Do Vu, Robert Belfer, Varun Gupta, Iulian Vlad Serban, and Joelle Pineau. "Automated data-driven generation of personalized pedagogical interventions in intelligent tutoring systems." In: *International Journal of Artificial Intelligence in Education* 32.2 (2022), pp. 323–349.
- [12] Tiffany Barnes and John Stamper. "Toward automatic hint generation for logic proof tutoring using historical student data." In: *International conference on intelligent tutoring systems*. Springer. 2008, pp. 373–382.
- [13] Mariam Mahdaoui, NOUH Said, My Seddiq Elkasmi Alaoui, and Mounir Sadiq. "Comparative study between automatic hint generation approaches in Intelligent Programming Tutors." In: *Procedia Computer Science* 198 (2022), pp. 391–396.
- [14] Shayan Doroudi, Vincent Aleven, and Emma Brunskill. "Where's the reward? a review of reinforcement learning for instructional sequencing." In: *International Journal of Artificial Intelligence in Education* 29 (2019), pp. 568–620.
- [15] Hassan Khosravi, Kirsty Kitto, and Joseph Jay Williams. "Ripple: A crowdsourced adaptive platform for recommendation of learning activities." In: *arXiv preprint arXiv:1910.05522* (2019).
- [16] Yossi Ben David, Avi Segal, and Yaakov Gal. "Sequencing educational content in classrooms using Bayesian knowledge tracing." In: *Proceedings of the sixth international conference on Learning Analytics & Knowledge*. 2016, pp. 354–363.
- [17] Chris Piech, Jonathan Spencer, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas Guibas, and Jascha Sohl-Dickstein. "Deep knowledge tracing." In: *arXiv preprint arXiv:1506.05908* (2015).
- [18] Avi Segal, Yossi Ben David, Joseph Jay Williams, Kobi Gal, and Yaar Shalom. "Combining difficulty ranking with multi-armed bandits to sequence educational content." In: *International conference on artificial intelligence in education*. Springer. 2018, pp. 317–321.
- [19] Benjamin Clement, Didier Roy, Pierre-Yves Oudeyer, and Manuel Lopes. "Multi-armed bandits for intelligent tutoring systems." In: *arXiv preprint arXiv:1310.3174* (2013).
- [20] Jonathan Gordon, Linhong Zhu, Aram Galstyan, Prem Natarajan, and Gully Burns. "Modeling concept dependencies in a scientific corpus." In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016, pp. 866–875.

- [21] Shuting Wang, Alexander Ororbia, Zhaohui Wu, Kyle Williams, Chen Liang, Bart Pursel, and C Lee Giles. "Using prerequisites to extract concept maps from textbooks." In: *Proceedings of the 25th ACM international conference on information and knowledge management*. 2016, pp. 317–326.
- [22] Yiming Yang, Hanxiao Liu, Jaime Carbonell, and Wanli Ma. "Concept Graph Learning from Educational Data." In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. WSDM '15. Shanghai, China: Association for Computing Machinery, 2015, 159–168. ISBN: 9781450333177.
- [23] Norzaidah Md Noh, Azlin Ahmad, Shamimi Ab Halim, and Azliza Mohd Ali. "Intelligent tutoring system using rule-based and case-based: a comparison." In: *Procedia-Social and Behavioral Sciences* 67 (2012), pp. 454–463.
- [24] Tong Mu, Karan Goel, and Emma Brunskill. "Program2Tutor: Combining Automatic Curriculum Generation with Multi-Armed Bandits for Intelligent Tutoring Systems." In: *Conference on Neural Information Processing Systems*. 2017.
- [25] Tong Mu, Shuhan Wang, Erik Andersen, and Emma Brunskill. "Combining adaptivity with progression ordering for intelligent tutoring systems." In: *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*. 2018, pp. 1–4.
- [26] Benjamin J Dringoli, Ksenia Kolosova, Thomas J Rademaker, Juliann Wray, Jeremie Choquette, and Michael Hilke. "Content Sequencing and its Impact on Student Learning in Electromagnetism: Theory and Experiment." In: *arXiv preprint arXiv:1910.00145* (2019).
- [27] Benjamin S Bloom, Max D Engelhart, Edward J Furst, Walker H Hill, David R Krathwohl, et al. *Taxonomy of educational objectives: The classification of educational goals. Handbook 1: Cognitive domain*. Longman New York, 1956.
- [28] Albert T Corbett and John R Anderson. "Knowledge tracing: Modeling the acquisition of procedural knowledge." In: *User modeling and user-adapted interaction* 4.4 (1994), pp. 253–278.
- [29] Benjamin Clement, Pierre-Yves Oudeyer, and Manuel Lopes. "A comparison of automatic teaching strategies for heterogeneous student populations." In: *EDM 16-9th international conference on educational data mining*. 2016.
- [30] Anna N Rafferty, Emma Brunskill, Thomas L Griffiths, and Patrick Shafto. "Faster teaching via pomdp planning." In: *Cognitive science* 40.6 (2016), pp. 1290–1332.
- [31] Jung In Lee and Emma Brunskill. "The Impact on Individualizing Student Models on Necessary Practice Opportunities." In: *International educational data mining society* (2012).

- [32] S Chaiklin. *The zone of proximal development in Vygotsky's analysis of learning and instruction. Vygotsky's Educational Theory in Cultural Context*, 1, 39–64. 2003.
- [33] Radek Pelánek and Jiří Řihák. "Experimental analysis of mastery learning criteria." In: *Proceedings of the 25th conference on user modeling, adaptation and personalization*. 2017, pp. 156–163.
- [34] Cassidy B McDougale, Sarah M Richling, Emily B Longino, and Soracha A O'Rourke. "Mastery criteria and maintenance: A descriptive analysis of applied research procedures." In: *Behavior Analysis in Practice* 13 (2020), pp. 402–410.
- [35] Kim Kelly, Yan Wang, Tamisha Thompson, and Neil Heffernan. "Defining mastery: Knowledge tracing versus n-consecutive correct responses." In: *Student Modeling From Different Aspects* (2016), p. 39.
- [36] Zacharoula Papamitsiou, Kshitij Sharma, and Michail N Giannakos. "On the Dependence Structure Between Learners' Response-time and Knowledge Mastery: If Not Linear, Then What?" In: *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*. 2020, pp. 201–210.
- [37] Philip I Pavlik Jr, Hao Cen, and Kenneth R Koedinger. "Performance Factors Analysis—A New Alternative to Knowledge Tracing." In: *Online Submission* (2009).
- [38] Erik Andersen, Sumit Gulwani, and Zoran Popovic. "A trace-based framework for analyzing and synthesizing educational progressions." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2013, pp. 773–782.
- [39] Michael V Yudelson, Kenneth R Koedinger, and Geoffrey J Gordon. "Individualized bayesian knowledge tracing models." In: *Artificial intelligence in education: 16th international conference, AIED 2013, Memphis, TN, USA, July 9–13, 2013. proceedings* 16. Springer. 2013, pp. 171–180.
- [40] Hao Cen, Kenneth Koedinger, and Brian Junker. "Learning factors analysis—a general method for cognitive model evaluation and improvement." In: *International Conference on Intelligent Tutoring Systems*. Springer. 2006, pp. 164–175.
- [41] Lilian Besson and Emilie Kaufmann. "The generalized likelihood ratio test meets klucb: an improved algorithm for piece-wise non-stationary bandits." In: *arXiv preprint arXiv:1902.01575* (2019).
- [42] Fang Liu, Joohyun Lee, and Ness Shroff. "A change-detection based framework for piecewise-stationary multi-armed bandit problem." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

- [43] Yang Cao, Zheng Wen, Branislav Kveton, and Yao Xie. "Nearly optimal adaptive procedure with change detection for piecewise-stationary bandit." In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 418–427.
- [44] Gary Lorden et al. "Procedures for reacting to a change in distribution." In: *The Annals of Mathematical Statistics* 42.6 (1971), pp. 1897–1908.
- [45] Alexander G Tartakovsky, Boris L Rozovskii, Rudolf B Blazek, and Hongjoong Kim. "A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods." In: *IEEE transactions on signal processing* 54.9 (2006), pp. 3372–3382.
- [46] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press.
- [47] Tao Li and Sam Sambasivam. "Question difficulty assessment in intelligent tutor systems for computer architecture." In: *Proceedings of ISECON 2003* (2003).
- [48] Radek Pelánek, Tomáš Effenberger, and Jaroslav Čechák. "Complexity and difficulty of items in learning systems." In: *International Journal of Artificial Intelligence in Education* 32.1 (2022), pp. 196–232.
- [49] Rekha Ramesh, Shitanshu Mishra, M Sasikumar, and Sridhar Iyer. "Semi-automatic generation of metadata for items in a question repository." In: *2014 IEEE Sixth International Conference on Technology for Education*. IEEE. 2014, pp. 222–228.
- [50] Shilpi Banerjee, NJ Rao, and Chandrashekhar Ramanathan. "Rubrics for assessment item difficulty in engineering courses." In: *2015 IEEE Frontiers in Education Conference (FIE)*. IEEE. 2015, pp. 1–8.
- [51] Jonathan Bassen, Bharathan Balaji, Michael Schaarschmidt, Candace Thille, Jay Painter, Dawn Zimmaro, Alex Games, Ethan Fast, and John C Mitchell. "Reinforcement learning for the adaptive scheduling of educational activities." In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–12.
- [52] Guy Shani, David Heckerman, Ronen I Brafman, and Craig Boutilier. "An MDP-based recommender system." In: *Journal of Machine Learning Research* 6.9 (2005).
- [53] Romain Warlop, Alessandro Lazaric, and Jérémie Mary. "Fighting boredom in recommender systems with linear reinforcement learning." In: *Advances in Neural Information Processing Systems* 31 (2018).

- [54] Sumana Basu, Marc-André Legault, Adriana Romero-Soriano, and Doina Precup. "On the challenges of using reinforcement learning in precision drug dosing: delay and prolongedness of action effects." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 12. 2023, pp. 14102–14109.
- [55] Cem Tekin, Jonas Braun, and Mihaela van der Schaar. "etutor: Online learning for personalized education." In: *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2015, pp. 5545–5549.
- [56] Andrew S Lan and Richard G Baraniuk. "A Contextual Bandits Framework for Personalized Learning Action Selection." In: *EDM*. 2016, pp. 424–429.
- [57] Indu Manickam, Andrew S Lan, and Richard G Baraniuk. "Contextual multi-armed bandit algorithms for personalized learning action selection." In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017, pp. 6344–6348.
- [58] Walter L. Leite, Samrat Roy, Nilanjana Chakraborty, George Michailidis, A Corinne Huggins-Manley, Sidney D'Mello, Mohamad Kazem Shirani Faradonbeh, Emily Jensen, Huan Kuang, and Zeyuan Jing. "A novel video recommendation system for algebra: An effectiveness evaluation study." In: *LAK22: 12th International Learning Analytics and Knowledge Conference*. 2022, pp. 294–303.
- [59] Nilanjana Chakraborty, Samrat Roy, Walter L Leite, Mohamad Kazem Shirani Faradonbeh, and George Michailidis. "The Effects of a Personalized Recommendation System on Students' High-Stakes Achievement Scores: A Field Experiment." In: *International Educational Data Mining Society* (2021).
- [60] Ana Iglesias, Paloma Martínez, Ricardo Aler, and Fernando Fernández. "Learning content sequencing in an educational environment according to student needs." In: *Algorithmic Learning Theory: 15th International Conference, ALT 2004, Padova, Italy, October 2-5, 2004. Proceedings* 15. Springer. 2004, pp. 454–463.
- [61] Xueying Tang, Yunxiao Chen, Xiaoou Li, Jingchen Liu, and Zhiliang Ying. "A reinforcement learning approach to personalized learning recommendation systems." In: *British Journal of Mathematical and Statistical Psychology* 72.1 (2019), pp. 108–135.
- [62] Steve Olusegun Bada and Steve Olusegun. "Constructivism learning theory: A paradigm for teaching and learning." In: *Journal of Research & Method in Education* 5.6 (2015), pp. 66–70.
- [63] George Katona. "The role of the order of presentation in learning." In: *The American Journal of Psychology* (1942), pp. 328–353.

- [64] Peter Auer and Ronald Ortner. "Logarithmic Online Regret Bounds for Undiscounted Reinforcement Learning." In: *Advances in Neural Information Processing Systems*. Ed. by B. Schölkopf, J. Platt, and T. Hoffman. Vol. 19. MIT Press, 2006. URL: https://proceedings.neurips.cc/paper_files/paper/2006/file/c1b70d965ca504aa751ddb62ad69c63f-Paper.pdf.
- [65] Chi Jin, Zeyuan Allen-Zhu, Sébastien Bubeck, and Michael I Jordan. "Is Q-learning provably efficient?" In: *Advances in neural information processing systems* 31 (2018).
- [66] Christoph Dann and Emma Brunskill. "Sample complexity of episodic fixed-horizon reinforcement learning." In: *Advances in Neural Information Processing Systems* 28 (2015).
- [67] Lihong Li, Yu Lu, and Dengyong Zhou. "Provably optimal algorithms for generalized linear contextual bandits." In: *International Conference on Machine Learning*. PMLR. 2017, pp. 2071–2080.
- [68] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. "Improved algorithms for linear stochastic bandits." In: *Advances in neural information processing systems* 24 (2011).
- [69] Bo Sun, Yunzong Zhu, Yongkang Xiao, Rong Xiao, and Yungang Wei. "Automatic question tagging with deep neural networks." In: *IEEE Transactions on Learning Technologies* 12.1 (2018), pp. 29–43.
- [70] Noboru Matsuda, Jesse Wood, Raj Srivastava, Machi Shimmei, Norman Bier, et al. "Latent Skill Mining and Labeling from Courseware Content." In: *Journal of Educational Data Mining* 14.2 (2022).
- [71] Radek Pelánek. "Managing items and knowledge components: domain modeling in practice." In: *Educational Technology Research and Development* 68.1 (2020), pp. 529–550.
- [72] Mitchell J Nathan, Kenneth R Koedinger, Martha W Alibali, et al. "Expert blind spot: When content knowledge eclipses pedagogical content knowledge." In: *Proceedings of the third international conference on cognitive science*. Vol. 644648. 2001, pp. 644–648.
- [73] Steven Moore, Robin Schmucker, Tom Mitchell, and John Stamper. "Automated Generation and Tagging of Knowledge Components from Multiple-Choice Questions." In: *arXiv preprint arXiv:2405.20526* (2024).
- [74] Steven Moore, Huy A Nguyen, and John Stamper. "Towards Crowdsourcing the Identification of Knowledge Components." In: *Proceedings of the Seventh ACM Conference on Learning@ Scale*. 2020, pp. 245–248.

- [75] Steven Moore, Huy A Nguyen, and John Stamper. "Evaluating Crowdsourcing and Topic Modeling in Generating Knowledge Components from Explanations." In: *International Conference on Artificial Intelligence in Education*. Springer. 2020, pp. 398–410.
- [76] Steven Moore, Huy A Nguyen, and John Stamper. "Leveraging Students to Generate Skill Tags that Inform Learning Analytics." In: (2022), pp. 791–798.
- [77] Sebastian Wankerl, Gerhard Götz, and Andreas Hotho. "f2tag—Can Tags be Predicted Using Formulas?" In: *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2020, pp. 565–571.
- [78] Xiaolu Wang, Ziqi Ding, and Liangyu Chen. "Automatic tagging of knowledge points for K12 math problems." In: *arXiv preprint arXiv:2208.09867* (2022).
- [79] Mario Karlovčec, Mariheida Córdova-Sánchez, and Zachary A Pardos. "Knowledge component suggestion for untagged content in an intelligent tutoring system." In: *Intelligent Tutoring Systems: 11th International Conference, ITS 2012, Chania, Crete, Greece, June 14-18, 2012. Proceedings 11*. Springer. 2012, pp. 195–200.
- [80] Thanaporn Patikorn, David Deisadze, Leo Grande, Ziyang Yu, and Neil Heffernan. "Generalizability of methods for imputing mathematical skills needed to solve problems from texts." In: *Artificial Intelligence in Education: 20th International Conference, AIED 2019, Chicago, IL, USA, June 25-29, 2019, Proceedings, Part I 20*. Springer. 2019, pp. 396–405.
- [81] Jia Tracy Shen, Michiharu Yamashita, Ethan Prihar, Neil Heffernan, Xintao Wu, Ben Graff, and Dongwon Lee. "Mathbert: A pre-trained language model for general nlp tasks in mathematics education." In: *arXiv preprint arXiv:2106.07340* (2021).
- [82] Zejie Tian, B Flanagan, Y Dai, and H Ogata. "Automated matching of exercises with knowledge components." In: *30th International Conference on Computers in Education Conference Proceedings*. 2022, pp. 24–32.
- [83] Zachary A Pardos and Anant Dadu. "Imputing KCs with representations of problem content and context." In: *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*. 2017, pp. 148–155.
- [84] Jia Tracy Shen, Michiharu Yamashita, Ethan Prihar, Neil Heffernan, Xintao Wu, Sean McGrew, and Dongwon Lee. "Classifying math knowledge components via task-adaptive pre-trained BERT." In: *International Conference on Artificial Intelligence in Education*. Springer. 2021, pp. 408–419.

- [85] Zhi Li, Zachary A Pardos, and Cheng Ren. "Aligning open educational resources to new taxonomies: How AI technologies can help and in which scenarios." In: *Computers & Education* 216 (2024), p. 105027.
- [86] Hang Li, Tianlong Xu, Jiliang Tang, and Qingsong Wen. "Automate Knowledge Concept Tagging on Math Questions with LLMs." In: *arXiv preprint arXiv:2403.17281* (2024).
- [87] Cheng Ren, Zachary Pardos, and Zhi Li. "Human-AI Collaboration Increases Skill Tagging Speed but Degrades Accuracy." In: *arXiv preprint arXiv:2403.02259* (2024).
- [88] Juho Kim et al. "Learnersourcing: improving learning with collective learner activity." PhD thesis. Massachusetts Institute of Technology, 2015.
- [89] Shayan Doroudi. "Integrating human and machine intelligence for enhanced curriculum design." PhD thesis. Carnegie Mellon University, 2019.
- [90] Joseph Jay Williams, Juho Kim, Anna Rafferty, Samuel Maldonado, Krzysztof Z Gajos, Walter S Lasecki, and Neil Heffernan. "Axis: Generating explanations at scale with learnersourcing and machine learning." In: *Proceedings of the third (2016) ACM conference on learning@ scale*. 2016, pp. 379–388.
- [91] Kenneth R Koedinger, Albert T Corbett, and Charles Perfetti. "The Knowledge-Learning-Instruction framework: Bridging the science-practice chasm to enhance robust student learning." In: *Cognitive science* 36.5 (2012), pp. 757–798.
- [92] Marco Ciotti, Massimo Ciccozzi, Alessandro Terrinoni, Wen-Can Jiang, Cheng-Bin Wang, and Sergio Bernardini. "The COVID-19 pandemic." In: *Critical reviews in clinical laboratory sciences* 57.6 (2020), pp. 365–388.
- [93] Edward Loper and Steven Bird. "NLTK: The Natural Language Toolkit." In: *CoRR cs.CL/0205028* (2002).
- [94] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. "Chain-of-thought prompting elicits reasoning in large language models." In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837.
- [95] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. "Gpt-4 technical report." In: *arXiv preprint arXiv:2303.08774* (2023).

- [96] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. “Gemini: a family of highly capable multimodal models.” In: *arXiv preprint arXiv:2312.11805* (2023).
- [97] AI@Meta. “Llama 3 Model Card.” In: (2024). URL: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.
- [98] Vishakha Patil, Vineet Nair, Ganesh Ghalme, and Arindam Khan. “Mitigating Disparity while Maximizing Reward: Tight Anytime Guarantee for Improving Bandits.” In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*. Main Track. Aug. 2023, pp. 4100–4108.