Proximity-Based Educational Recommendations: A Multi-Objective Framework*

CHUNPAI WANG, University at Albany, State University of New York, USA
SHAGHAYEGH SAHEBI, University at Albany, State University of New York, USA
PETER BRUSILOVSKY, University of Pittsburgh, USA

Personalized learning and educational recommender systems are integral parts of modern online education systems. In this context, the problem of recommending the best learning material to students is a perfect example of sequential multi-objective recommendation. Learning material recommenders need to optimize for and balance between multiple goals, such as adapting to student ability, adjusting the learning material difficulty, increasing student knowledge, and serving student interest, at every step of the student learning sequence. However, the obscurity and incompatibility of these objectives pose additional challenges for learning material recommenders. To address these challenges, we propose Proximity-based Educational Recommendation (PEAR), a recommendation framework that suggests a ranked list of problems by approximating and balancing between problem difficulty and student ability. To achieve an accurate approximation of these objectives, PEAR can integrate with any state-of-the-art student and domain knowledge model. As an example of such student and domain knowledge model, we introduce Deep Q-matrix based Knowledge Tracing model (DQKT), and integrate PEAR with it. Rather than static recommendations, this framework dynamically suggests new problems at each step by tracking student knowledge level over time. We use an offline evaluation framework, Robust Evaluation Matrix (REM), to compare PEAR with various baseline recommendation policies under three different student simulators and demonstrate the effectiveness of our proposed model. We experiment with different student trajectory lengths and show that while PEAR can perform better than the baseline policies with fewer data, it is also robust with longer sequence lengths.

CCS Concepts: • **Information systems** → **Personalization**; • **Applied computing** → **E-learning**.

Additional Key Words and Phrases: multi-objective, educational recommender system, knowledge modeling

1 INTRODUCTION

As online educational systems, such as Massive Open Online Courses (MOOCs), become more prevalent, there is a growing need for them to accommodate individual differences between students. To handle the abundance of data and to guide students efficiently, these systems need automatic tools, such as educational recommender systems, for guiding students. Educational recommender systems aim to suggest the best learning material tailored to each student, in contrast to the "one-size-fits-all" experience in traditional classrooms.

With the recent developments in machine learning techniques, researchers have applied new recommender system techniques in the education context to provide a personalized e-learning experience [4, 14, 15, 25, 26]. However, these educational recommender systems tend to ignore the multi-objectivity of the educational applications and only focus on one objective at a time. For example, they aim to recommend a problem that is highly likely for the student to solve [14], has a tolerable difficulty level for the target student [25], or increases the student's knowledge as much as possible [2, 26]. While these objectives are important aspects of educational recommendations, we argue that addressing them separately is not adequate for achieving an optimal learning experience, and may even harm student learning. For instance, only recommending problems that the students can solve would lead to suggesting problems with concepts that the student is already knowledgeable about. As a result, the students will not practice the concepts that they are uncertain about, nor they will be exposed to new concepts. Such a recommender system may not challenge the students enough to increase their knowledge, and can bore the students. Conversely, only optimizing for increasing student knowledge

*Copyright 2022 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
Presented at the MORS workshop held in conjunction with the 16th ACM Conference on Recommender Systems (RecSys), 2022, in Seattle, USA.

would result in recommending problems with new concepts for students. Not only this can lead to recommending problems that are too advanced or difficult for the student to comprehend, but it may also discourage the students from learning. An ideal problem recommender system should be able to address these objectives simultaneously, and balance between students' ability to solve the suggested problem and the expected knowledge gain as a result of solving it.

To achieve such a balance, the ideal problem recommender system should be able to quantify students' current state of knowledge and the concepts that are provided in each problem. However, the unobservability of these variables imposes an additional challenge for educational recommenders. While students' past performance (scores or grades) in their attempted problems are observed in online education systems, their knowledge in the underlying concepts of these problems is hidden. Similarly, the underlying concepts provided in these problems can be latent and need to be discovered. Accordingly, an educational recommender system should build upon a reliable *student knowledge model* to track the students' latent knowledge and a *domain knowledge model* to represent the problems' latent concepts.

In this paper, we propose a new recommendation framework, Proximity-based Educational Recommendation (PEAR), that is based on properly designed student knowledge modeling and domain knowledge modeling. To suggest a ranked list of problems to students, PEAR strikes a balance between problem difficulty and student ability. It can integrate with any student and domain knowledge model that maps student knowledge and problem concepts in the same unobservable latent subspace and does not require concept annotations on learning materials from domain experts. We demonstrate an example of student knowledge modeling and domain knowledge modeling to be integrated with the PEAR framework. To this end, we introduce Deep Q-matrix based Knowledge Tracing (DQKT), which is a variation of the DKVMN knowledge model [35] and unifies the student knowledge and problem concept subspaces. To evaluate PEAR, we use an offline evaluation framework, Robust Evaluation Matrix (REM) [9, 30], which is specifically designed for educational sequencing and experiments with different student trajectory simulators that are trained on real-world data. Using REM, we compare PEAR with eleven baseline recommendation policies under three different student simulators with a real-world dataset collected from the Mastery Grids [17] intelligent tutoring system. Our experiments demonstrate PEAR's effectiveness in comparison with the baseline policies in all student simulators. We also experiment with different student trajectory lengths and show that while PEAR can perform better than the baseline policies with fewer data, it is also robust with longer sequence lengths.

2 RELATED WORK

Our proposed method is related to past existing work on: 1) instructional sequencing or personalized learning; and 2) knowledge tracing or knowledge modeling; In this section, we discuss these two key related works. In addition, we also discuss the state-of-the-art model-based offline evaluation of instructional sequencing called Robust Evaluation Matrix (REM), which is used in our experiment to demonstrate the performance of our proposed recommendation framework.

Student and Domain Knowledge Modeling. *Student knowledge modeling* (SKM) or knowledge tracing (KT) is a task of modeling student knowledge as they interact with coursework over time. One follow-up task of knowledge modeling is to predict the student's future performance, and another is to provide the personalized learning experience such as learning materials based on the student's knowledge. Student knowledge modeling has a long history in educational data mining. For example, Bayesian Knowledge Tracing (BKT)[8] is a pioneer student knowledge modeling approach that is based on hidden Markov models and is famous for its interpretability. It models students' knowledge as a latent variable and maintains an estimate of the probability that the student has mastered a particular set of skills or *knowledge components* (KCs). It can also explicitly model students' knowledge growth for different KCs, which is important for personalized learning. However, classic BKT requires each traced problem-solving step or activity to be associated by

a single KC. Such requirement is hard to enforce in advanced courses that include complex learning materials with multiple involved knowledge components. Moreover, BKT uses the association between activities and KCs an input to the model. In the past, these associations were defined by domain experts. However, this is known as a very expensive and error-prone task.

To resolve this issue, automated approaches to *domain knowledge modeling* were developed and explored. These approaches aim to automatically quantify the associations between learning materials and the knowledge components practiced by working with these materials. The knowledge components themselves can be predefined or also discovered automatically as a set of latent variables. Initially, automated domain knowledge modeling focused on improving the predefined associations between learning material and KC defined by experts in Q-matrices [1, 27]. A typical Q-matrix is defined as a binary matrix $Q \in \{0, 1\}^{\mathcal{N} \times C}$ where $\mathcal{N}$ denotes the number of questions and C denotes the number of knowledge components or concepts. Each row is a representation of the mapping of KCs to the corresponding learning material. Later on, researchers focused on automatic discovery of these associations with non-binary and probabilistic variations for Q-matrices [16, 24, 31].

Recently, advanced student and domain knowledge modeling approaches were developed to simultaneously address these two problems [36]. Additionally, with the advance of deep learning techniques on predictive models, deep knowledge tracing models such as DKT[23], DKVMN [35], DMKT [32], SAKT [20], AKT [11], and SAINT [6] have been introduced. Although these deep-learning based KT methods are showing superior performance in predicting student scores compared to their traditional peers, they are usually criticized for their poor interpretability.

Personalized Learning. Many researchers have been trying to develop intelligent tutoring system to optimize the sequencing of instructional activities and provide the personalized learning experience to students since the 1960s [10]. Generally, most personalized systems are designed by considering learners' cognitive states, behaviors, or preferences.

One important goal of personalized learning is to improve students' learning gain. Some methods aim to improve students' knowledge gain by considering the students' cognitive state or knowledge state using knowledge modeling. For example, Pardos et al. [21] proposed a Bayesian method using similar permutation analysis techniques and knowledge tracing to determine which orderings of questions produce the most learning. Pelanek et al. [22] leveraged knowledge tracing to model students' skill acquisition over time and sequence questions to students based on their mastery level and predicted performance. One limitation of these knowledge tracing models is the assumption of a single knowledge concept for each learning content. Item Response Theory (IRT) models have been utilized in many adaptive testing systems to predict students' dichotomous response[3]. Chen et al. [4] proposed a personalized e-learning system based on Item Response Theory (PEL-IRT) which estimates learner ability and problem difficulty and accordingly suggests the course material that includes the maximum information for the student. Baylari et al. [2] also proposed a personalized multi-agent e-learning system based on item response theory (IRT) and artificial neural network (ANN) which presents adaptive tests (based on IRT) and personalized recommendations (based on ANN). However, the uni-dimensional IRT also assumes a single concept for each learning content and fixed user ability over time, and multi-dimensional IRT model is rarely investigated for personalized learning action selection due to the inefficiency on large scale data.

Another class of personalized learning methods rely on the reinforcement learning to maximize the studnets' learning gain. Multi-armed bandits framework is a popular reinforcement learning model to select actions sequentially that achieve students' long-term rewards [13, 18, 28, 33, 34]. Segal et al. [25] proposed to combine the difficulty ranking with multi-armed bandits to personalize educational content to students in order to maximize their learning gains over time. This model integrates offline learning from students' past interactions with online mechanisms, which requires the student simulation model to evaluate the performance. Lan et al. [14] also utilized the SPARFA framework

to first estimate each student's knowledge profile from their binary-valued graded responses to questions in their previous assessments, then employ these knowledge profiles as contexts in the contextual (multi-armed) bandits framework to recommend the most suitable contents. Lau et al. employed the unbiased offline evaluation by assuming random recommendation policy on logged system. Clement et al. [7] also proposed to integrate the expert domain knowledge on educational contents into the multi-armed bandits for more powerful personalized learning action selection. Chi et al. [5] leveraged the model-based reinforcement learning on a natural language tutoring system for inducing effective pedagogical strategies with empirical evaluations of the induced strategies. However, multi-armed bandits and reinforcement learning based sequential recommendation system suffer from an ineffective reward design and offline evaluation on sparse logged data.

3 PEAR: PROXIMITY-BASED EDUCATIONAL RECOMMENDATION

3.1 Problem Formulation

We assume an online education system, in which M students attempt N problems in a course. Although the system may provide a schedule for the topics presented in class, the students are free to attempt the problems in any order, for any number of times. Our goal is to recommend the best next problem to each student, personalized for the student's knowledge state at each attempt, to guide student knowledge acquisition, while considering students' individual abilities. We aim to solve this problem only using student attempt sequences on problems, without requiring extra information such as problem content or difficulty. We represent student u's score in problem q at the attempt or time step t as $x^t_{u,q}$. This score could be a binary value that represents the correctness of student attempts or a continuous normalized score in $[0, 1]$. All observed students' scores on all of their attempted problems are kept in a set Ω_{obs}.

3.2 PEAR Framework

We propose *P*roximity-based *E*duc*A*tional *R*ecommendation Framework (*PEAR*) that is established based on student and domain knowledge representations. Notably, PEAR aims to present students with problems that (1) are neither too easy nor too difficult, but (2) are slightly beyond or proximal to their current knowledge states, according to their performance on previous problems. These objectives are inspired by the concept of zone of proximal development [29] from Educational Psychology research, that refers to the space between what a student can do without assistance and what they can do with guidance. It suggests that the students learn from practicing the problems that are proximal to their abilities or the skills that are close to mastering.

We note that students' knowledge of a problem is associated with their likelihood of correctly solving that problem. The more knowledgeable the students are in the problem concepts, the easier it will be for them to achieve a good score. As a result, the two abovementioned objectives can be incompatible at times and a balance should be kept between the two. PEAR aims to balance the difficulty of the suggested problems for students and the potential knowledge gain that a student with a personalized estimated ability will have as a result of solving the suggested problems.

3.2.1 Proximity-Based Recommendation. PEAR's goal is to suggest the best problems for students to learn at each attempt. To quantify each problem q's utility for the target student u at attempt t, we propose a proximity score $\text{prox}^t_{u,q}$ that can keep the balance between problem difficulty and student ability at that attempt. After each student-problem interaction, PEAR suggests a ranked list of problems based on this proximity score to provide a personalized problem sequence and improve students' overall knowledge gain in the class. Particularly, given student u's knowledge at time

step t ($\mathbf{k}_u^t$), and the concepts provided in problem q ($\mathbf{w}_q$), we propose to formulate $\text{prox}_{u,q}^t$ as:

$$\text{prox}_{u,q}^{t+1} = \begin{cases} (1 - \hat{x}_{u,q}^t) \cdot \text{sim}(\mathbf{k}_u^t, \mathbf{w}_q) & \text{if} \quad x_{u,q}^t \notin \Omega_{obs} \\ (1 - x_{u,q}^t) \cdot \text{sim}(\mathbf{k}_u^t, \mathbf{w}_q) & \text{if} \quad x_{u,q}^t \in \Omega_{obs} \end{cases} \tag{1}$$

where $\text{sim}(\mathbf{k}_u^t, \mathbf{w}_q)$ is a similarity measurement between the current student knowledge $\mathbf{k}_u^t$ and problem concept $\mathbf{w}_q$, and $x_{u,q}^t$ ($\hat{x}_{u,q}^t$) is the (estimated) student performance in the problem q. To have the most accurate representation of student performance, we use the observed performance $x_{u,q}^t$ if it exists (if the student has tried problem q in their last attempt). Otherwise, we estimate student performance, as in $\hat{x}_{u,q}^t$. In this formulation, the higher (lower) the similarity $\text{sim}(\mathbf{k}_u^t, \mathbf{w}_q)$, the more familiar (newer) the suggested problem's concepts are for the student. While with a higher $\hat{x}_{u,q}^t$, the student has a higher chance to correctly solve the suggested problem, or a lower chance to fail in solving the problem $(1 - \hat{x}_{u,q}^t)$. Since student knowledge in problem concepts $\text{sim}(\mathbf{k}_u^t, \mathbf{w}_q)$ is associated with a higher score in the problem (lower $(1 - x_{u,q}^t)$), multiplication of these two terms creates a balance between the knowledge increase objective and the problem difficulty for the student. As a result, by selecting problems with maximized $\text{prox}_{u,q}^t$, PEAR recommends problems with new concepts (knowledge increase) that the students are likely to solve (less difficulty), or problems with familiar concepts (knowledge proximity) that the students are less likely to solve (more difficulty).

Pseudocode of PEAR framework is provided in Algorithm 1. In line 2, PEAR first uses a student and domain knowledge model to provide an estimation of student performance ($\hat{x}_{u,q}^t$), students' current knowledge ($\mathbf{k}_u^t$), and problem concepts ($\mathbf{w}_q$). Then, it calculates the proximity score $\text{prox}_{u,q}^t$ for each problem (lines 3-5). It sorts and suggests the top problems to be suggested to the target student (lines 6 and 7). In the next section, we provide the requirements for the student and domain knowledge models that can be integrated with PEAR and introduce an example of such models.

Algorithm 1: PEAR Framework

Input: Observed students' interaction records Ω_{obs}, including training students historical interaction records and target student u's first historical interaction record;

1 **for** *each time index* $1 \leq t < \mathcal{T}$ **do**
2 Apply the compatible domain knowledge modeling and student knowledge modeling on observed data Ω_{obs} to obtain $\mathbf{w}_q$ and $\hat{x}_{u,q}^t$ for all $q \in \{1, \cdots, \mathcal{N}\}$ and $\mathbf{k}_u^t$.
3 **for** *each problem q in the problem pool* **do**
4 compute the proximity score:

$$\text{prox}_{u,q}^{t+1} = \begin{cases} (1 - \hat{x}_{u,q}^t) \cdot \text{sim}(\mathbf{k}_u^t, \mathbf{w}_q) & \text{if} \quad x_{u,q}^t \notin \Omega_{obs} \\ (1 - x_{u,q}^t) \cdot \text{sim}(\mathbf{k}_u^t, \mathbf{w}_q) & \text{if} \quad x_{u,q}^t \in \Omega_{obs} \end{cases}$$

5 **end**
6 Sort the problems based on proximity scores from high to low.
7 Recommend top-k problems to the target user u to work on at time $t + 1$.
8 The target user u selects one of k problems to solve, and the score $x_{u,q}^t$ is observed and saved into Ω_{obs}.
9 **end**

3.2.2 Requirements for Domain and Student Knowledge Models . PEAR can work with various student and domain knowledge modeling methods to provide personalized problems to students in different stages of their sequential learning. Particularly, in addition to the recommendation algorithm, PEAR relies on two other components to work with: 1) a domain knowledge model that maps problems or problems to the latent concepts, topics, or knowledge components in problems, that can be represented in a matrix format called a Q-matrix ; and 2) a student knowledge

model (K) that is compatible with that domain knowledge model and can accurately estimate students' knowledge in the same latent subspace as the domain knowledge model and predict their performance in future problems.

These two components must adopt the following specifications to be compatible with PEAR. The aforementioned domain knowledge model Q should represent how much each latent course concept ($c \in \{1, \cdots, C\}$) is presented in each problem ($q \in \{1, \cdots, \mathcal{N}\}$). In other words, in such a domain model, each problem q can be defined using a C-dimensional vector or embedding $\mathbf{w}_q$, where $w_{q,c}$ is the importance of concept c in that problem. Typically, the problem embedding $\mathbf{w}_q$ is sparse, and sums to 1. Additionally, the compatible student knowledge model should be able to quantify and estimate each student's knowledge in the same set of latent concepts $c \in \{1, \cdots, C\}$ at each attempt or time step t. That is, for student u at time step t, student knowledge can be represented as a C-dimensional vector $\mathbf{k}_u^t$. It is important to note that student performance $x_{u,q}^t$ is a function of student knowledge $\mathbf{k}_u^t$ and problem-concept correlation embedding $\mathbf{w}_q$. For example, the dot product of these two embeddings could produce a good estimate of student's performance $x_{u,q}^t$, such as:

$$x_{u,q}^t \approx \hat{x}_{u,q}^t = \mathbf{k}_u^{t^\top} \mathbf{w}_q \tag{2}$$

Accordingly, any domain and student knowledge model that fulfills these requirements can be used in PEAR.

3.3 DQKT: An Example of Domain and Student Knowledge Models

Here, we present Deep Q-matrix based Knowledge Tracing (DQKT) as an example of a compatible student and knowledge model for PEAR. DQKT is a revised version of Dynamic Key-Value Memory Networks for Knowledge Tracing (DKVMN) [35], which is a pioneer deep knowledge tracing model. We revise DKVMN as DQKT to satisfy the requirements of PEAR framework. The network architecture of DQKT is shown in Fig. 1.

Both DQKT and DKVMN are based on a recurrent version of Key-Value memory networks that consist of two main components: the read component to extract domain and student knowledge and predict the student's performance, and the write component to update the student knowledge. DQKT and DKVMN have exactly the same write components, and so we omit the details of this component. The main difference between DQKT and DKVMN is in the read component (as shown on the left part of Fig. 1). As explained in the following, this component is used to extract domain and student knowledge to predict the student's performance.

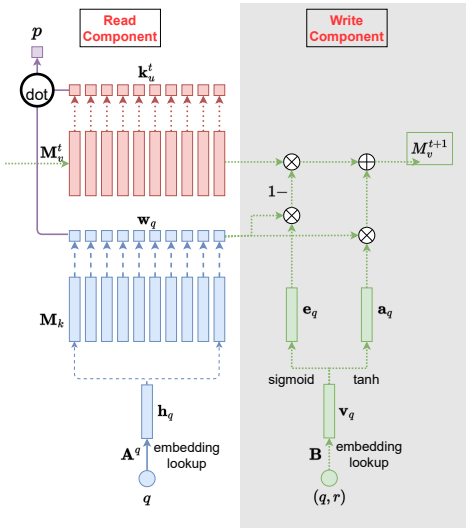

Fig. 1. Deep Q-matrix Based Knowledge Tracing Model. The model could be viewed as two main components: read (left) and write (right). The details are illustrated in Section 3.3

.

Domain Knowledge Modeling. Assuming that there are C latent knowledge concepts $\{\mathbf{c}_1, \cdots, \mathbf{c}_C\}$ or knowledge components for each problem, DKVMN represents each latent concept $\mathbf{c}_i$ by a d_h-dimensional embedding, $\mathbf{c}_i \in \mathbf{R}^{d_h}$ that can be interpreted as latent concept features. Both DQKT and DKVMN have a static key matrix $\mathbf{M}_k$ of size $C \times d_h$ to store the C latent concept embeddings (represented as blue vectors in the read component of Fig. 1). They also

represent each problem q using a d_h-dimensional embedding $\mathbf{h}_q$. However, DKVMN does not have a compatible sparse representation of problems in the concepts space to be used as the C-dimensional $\mathbf{w}_q \in \mathbf{R}^C$ vector in the Q-matrix.

To solve this incompatibility problem and obtain the sparse problem-concept vector $\mathbf{w}_q$ for problem q, in DQKT we use an embedding lookup table to compute the embedding $\mathbf{h}_q$, multiply it with latent concept embeddings $\mathbf{M}_k$, and apply the Sparsemax operation [19] to ensure the sparsity of the resulting correlation vector $\mathbf{w}_q$:

$$\mathbf{w}_q = \text{Sparsemax}(\mathbf{h}_q \mathbf{M}_k^\top) \tag{3}$$

Here, the ith entry of $\mathbf{w}_q$, denoted by $w_q(i)$, shows the representation of concept $\mathbf{c}_i$ in problem q.

Student Knowledge Modeling. Both DQKT and DKVMN update a value matrix, $\mathbf{M}_v^t$ of size $C \times d_h$, as the model observes student attempts on problems. This matrix can be interpreted as the student's mastery levels of each concept feature, at time step t. In DQKT, to represent student knowledge in each concept as a C-dimensional vector $\mathbf{k}_u^t \in \mathbf{R}^C$, we summarize the value matrix $\mathbf{M}_v^t$ by summing over the concept latent dimension. The value matrix $\mathbf{M}_v^t$ could be updated with an *erase-follow-by-add* operation as shown in the write component, which is exactly the same as in the DKVMN.

Student Performance Prediction. In the end, the student's performance at time step t is estimated by Eq.(4), which aims to capture the linear dependence between student's knowledge-concept mastery embedding and the problem-concept correlation embedding. Here, σ is the sigmoid function. The model is trained by minimizing the cross entropy between the true score $x_{u,q}^t$ and the predicted score $\hat{x}_{u,q}^t$, and we could derive the domain and student knowledge representations $\mathbf{w}_{q_t}$ and $\mathbf{k}^t$, respectively.

$$x_{u,q}^t \approx \hat{x}_{u,q}^t = \sigma(\mathbf{k}_u^{t^\top} \mathbf{w}_q) \tag{4}$$

Integration with PEAR. Now, we can calculate PEAR's $\text{prox}_{u,q}^{t+1}$ for DQKT by defining a similarity measure between student knowledge $\mathbf{k}_u^t$ and sparse problem concepts vector $\mathbf{w}_q$. Given that $\mathbf{k}_u^t$ and $\mathbf{w}_q$ are in the same latent subspace, Cosine similarity is a natural choice for calculating $\text{sim}(\mathbf{k}_u^t, \mathbf{w}_q)$. Accordingly, $\text{prox}_{u,q}^{t+1}$ for DQKT will be calculated as:

$$\text{prox}_{u,q}^{t+1} = \begin{cases} (1 - \hat{x}_{u,q}^t) \cdot \text{cosine}(\mathbf{k}_u^t, \mathbf{w}_q) & \text{if} \quad x_{u,q}^t \notin \Omega_{obs} \\ (1 - x_{u,q}^t) \cdot \text{cosine}(\mathbf{k}_u^t, \mathbf{w}_q) & \text{if} \quad x_{u,q}^t \in \Omega_{obs} \end{cases} \tag{5}$$

4 EXPERIMENTS

We conduct our experiments on a real-world dataset collected from an intelligent tutoring system and compare PEAR's performance with several instructional policies to answer the two following research questions:

- **RQ1.** Is PEAR's multi-objective policy better than other baseline policies, including an expert guided policy?
- **RQ2.** When is PEAR's policy most beneficial in terms of learning trajectory length?

We evaluate the recommendation performance with a state-of-the-art model-based offline evaluation method, robust evaluation matrix (REM), that is particularly designed for the educational context. We describe the general REM settings for our experiments and show the promises of our proposed PEAR recommendation policy.

4.1 Dataset

Mastery Grids [17] is a visually-rich, interactive, adaptive social E-learning portal that provides access to multiple kinds of smart learning content such as various kinds of problems and annotated examples for three programming languages (Java, Python, and SQL). We use the students' learning trajectory data collected during Spring 2012, Fall 2012, and Spring 2013 semesters in a Java introductory course with the same curriculum from the Mastery Grids platform.

In this course, the learning materials are ordered by 21 curriculum topics, and each topic includes multiple learning examples and problems. The topics cover a wide range of programming concepts including simple "`Variables`" and more complex "`Wrapper Classes`" topics ordered by domain experts. Although the topics are shown to the students, they can freely work on or attempt the problems or quizzes in any order as many times as they would like. Note that PEAR and DQKT do not use these topics as an input. However, these topics are used to define the required knowledge components for student knowledge models, like BKT, that rely on such information.

For our experiments, we use 86 student trajectories in solving 103 programming problems. These problems ask the students to read a code snippet and answer simple questions, such as the final output or a variable's value. For the purpose of reward estimations in REM evaluation, we use the pre-test and post-test scores of these students. Specifically, the students were asked to complete a pre-test before starting their class and a post-test, with the same problems, at the end of their course. The pre-test and post-test scores are normalized to be between zero and one, and students' knowledge gain is computed by deducing their normalized pre-test score from their normalized post-test score. Descriptive statistics of the dataset are shown in Table 1. Additionally, score and knowledge gain distributions are presented in Figure 2. As we can see, with a median trajectory length of 168, most students get to attempt all topics and problems by the end of the course. Also, the test and knowledge gain distribution shows that most of the students gain a substantial amount of knowledge after practicing with the system.

Table 1. Descriptive Statistics of Mastery Grids Dataset.

Dataset	#Users	#Problems	#Topics	#Records	Trajectory Length		
					Min	Median	Max
Mastery Grids	86	103	21	17741	11	168	988

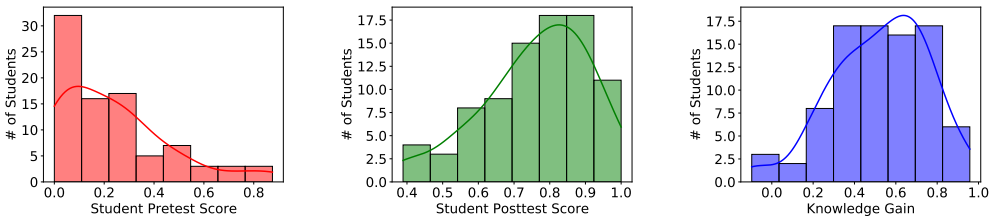

Fig. 2. Distribution of Pre-test Score (Left), Post-test Score (Mid.), and Knowledge Gain (Right).

4.2 REM Model-based Offline Evaluation

To evaluate PEAR with offline data, we leverage the robust evaluation matrix (REM) [9]. REM is the model-based offline evaluation method that uses various simulators, yielding more confidence in the target recommendation policy. It helps to determine if the target policy is promising and worth to be deployed for the costly online experiments. A diagram

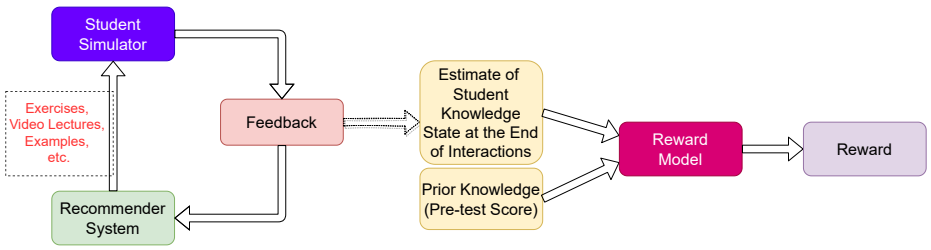

Fig. 3. Model-based Offline Evaluation.

of how REM works is shown in Fig. 3. Model-based evaluations in the education domain, such as REM, use a student knowledge model as a *student simulator* to simulate and estimate the students' performance under the policies that are being evaluated. In detail, the educational recommender system provides a programming problem to the target student, and then a student simulator provides the feedback (e.g., predicted student score) to one of the problems. The target student's feedback could be used to enhance the recommender system. Please note that these simulator models are different from the student knowledge models that are integrated with and are a part of the recommendation policies.

Since the offline evaluations are simulation-based, they have to also estimate the reward or utility for the students after attempting the recommended learning materials. Unlike many domains, the rewards in the education domain are not observed at every attempt and are delayed. For example, it is customary to use the overall course knowledge gain (the difference between post-test and pre-test) as the reward. This reward can only be observed at the end of the course or student sequence. As a result, a *reward model* is needed to estimate such a delayed reward at the end of each simulated student trajectory. This reward model can use various information to estimate the final reward. In REM, students' prior knowledge (as in their pre-test score) and an estimation of their knowledge state according to their simulated feedback is used to estimate the reward.

Due to imperfection of student simulator models, which will never behave exactly like a real student, the estimated performance using them will generally not be accurate. This may yield misleading evaluation conclusions. To resolve this issue, REM proposes to use multiple student simulators to help inform comparisons among different recommendation policies. The idea is to first estimate the performance of different recommendation policies by simulating them using multiple well-trained student models. Then the recommendation policies are evaluated in a conservative way: policy $\mathcal{A}$ is considered to be better than policy $\mathcal{B}$ if and only if policy $\mathcal{A}$ outperforms policy $\mathcal{B}$ under all student simulator models. Accordingly, in our experiment setup, we use three different student knowledge models to serve as the student simulators. We evaluate PEAR in comparison with 11 recommendation policies with baseline knowledge models and three recommendation policies with the DQKT knowledge model. The details of the student simulators, recommendation policies, and reward model are presented below.

Student Simulators. We use three student knowledge models as student simulators:

- Bayesian Knowledge Tracing (BKT) is a pioneer model based on Hidden Markov Models (HMMs) that estimates student probability of success based on the probability that the student has learned a topic (mastery) [8]. BKT needs predefined knowledge components as its input.
- Deep Knowledge Tracing (DKT) is the first deep learning based knowledge tracing model that sequentially predicts student's correctness probability using LSTM [12] according to their knowledge estimate [23].
- Deep Q-Matrix Based Knowledge Tracing (DQKT) is our proposed knowledge modeling method, which serves as an example of domain and student knowledge modeling that is compatible with our proposed PEAR framework.

Baseline Policies. We compare our proposed PEAR integrated with DQKT (or DQKT-PEAR) with the commonly used baseline educational recommendation policies, using REM. Please note that some of these baseline recommendation policies are achieved by combining a general policy (e.g., "Uncertain") with an underlying student knowledge model (e.g., "BKT"). Among BKT, DKT, and DQKT, only DQKT is compatible with PEAR and can be integrated with it for evaluation. The following is the list of baseline recommendation policies:

- Random[10]: randomly selects a problem that has not been previously answered correctly by the student. If all problems have been answered correctly, then it will randomly select a problem from the problem pool.

- InstructSeq[9, 10]: is the expert-guided policy that selects the next problem based on the curriculum. In our dataset, it is based on the ordered topics that are provided in MasteryGrids and moves from the problems from the basic course topics to the more advanced ones.
- InvInstructSeq[10]: is the inverse of InstructSeq policy that select the next problem based on the inverse of the curriculum. This policy is expected to perform poorly.
- BKT-MP[8, 9]: is the state-of-the-art Mastery Policy that is based on Bayesian Knowledge Tracing (BKT). It has been previously shown to be effective to improve student's knowledge level[8]. In detail, it chooses the problem that the student knowledge in it is the farthest away from a predefined mastery threshold, such as 0.95. The student knowledge is estimated using BKT. Consequently, it focuses on introducing the student to newer and unknown knowledge components.
- BKT-InvMP[10]: is the inverse policy of BKT-MP, and chooses the problem that the student knowledge in it is the closest to the predefined mastery threshold.
- BKT-HighestProbCorr[9]: is the BKT-based policy that chooses the problem that has the highest probability for the student to solve correctly and has not been answered correctly before. It focuses on recommending problems that are easy for the target student to solve.
- BKT-HighestProbIncorr[10]: is the BKT based policy that chooses the problem that has the lowest probability for the target student to solve correctly and has not been answered correctly before.
- BKT-Uncertain[10]: is the BKT based policy that recommend the problem with the highest uncertainty of if the student would be able to solve it correctly. The uncertainty is defined as the product of probability of correctness and probability of incorrectness.
- DKT-HighestProbCorr[9]: is similar to the BKT-HighestProbIncorr policy. However, DKT is used as the student knowledge model to estimate the probability of the student answering the problems correctly.
- DKT-HighestProbIncorr[10]: replaces the BKT in BKT-HighestProbIncorr with DKT student knowledge model.
- DKT-Uncertain[10]: is similar to the BKT-Uncertain policy. But, DKT is used as the student knowledge model.
- DQKT-HighestProbCorr: is similar to the BKT-HighestProbIncorr policy. However, DQKT is used as the student knowledge model to estimate the probability of the student answering the problems correctly.
- DQKT-HighestProbIncorr: replaces the BKT in BKT-HighestProbIncorr with DQKT student knowledge model.
- DQKT-Uncertain: is similar to the BKT-Uncertain policy. But, DQKT is used as the student knowledge model.

Reward Model. The reward model aims to estimate the final delayed reward for simulated students. Estimated knowledge gain, or the difference between the estimated post-test and observed pre-test, is used as the delayed reward in our evaluations. The reward model uses student pretest scores and knowledge state at the end of student trajectory, that is estimated by a student knowledge model, to estimate the post-test and knowledge gain. Following [9], we use BKT for the student knowledge model that is needed for reward estimation. The reward model is trained on the simulator parameters that are learned according to training students' trajectories as their independent variables and the training students' final utilities (e.g., post-test scores) as the dependent variables. We use the trained parameters to infer the mastery probabilities of the 21 topics in Mastery Grids. So, BKT model's knowledge representation is a 21-dimensional vector. To train the reward model for BKT simulator, we fit a regression model with the concatenation of pre-test score and knowledge vector as independent variables to predict the post-test score. Estimated knowledge gain can be calculated based on the difference between the estimated post-test and pre-test scores.

4.3 Experimental Results

Knowledge Modeling. Since the recommendation policies rely on the correctness of their underlying student knowledge models and REM relies on the accuracy of its underlying student simulator models, we first compare the prediction performance of the student knowledge models. We compare DQKT with the other two student simulators: BKT and DKT. Since DQKT is a model revised based on DKVMN, we also compare the prediction performance of DKVMN. For this, we use 5-fold cross-validation on the offline MasteryGrids dataset. The ROC-AUC (area under the receiver operating characteristic curve) results are shown in Table 2. Firstly, as we can see, by revising the DKVMN to suit our proposed PEAR framework, DQKT still has a similar prediction performance as the DKVMN. Secondly, DKT, DQKT, and BKT perform significantly different from each other (DKT > DQKT > BKT). As a result, their performance as student knowledge models may affect their associated recommendation policy performances. Additionally, as significantly different student simulators, they increase the robustness of REM offline evaluations.

Table 2. Predictive accuracy and 95% confidence intervals of comparing student models. Higher ROC-AUC is better.

Dataset	BKT	DKT	DKVMN	DQKT
	ROC-AUC	ROC-AUC	ROC-AUC	ROC-AUC
Mastery Grids	0.7009 ± 0.0288	0.7824 ± 0.0184	0.7585 ± 0.0220	0.7521 ± 0.0166

RQ1: PEAR vs. Baselines. To use REM for comparing PEAR with the baselines, we simulate 10 learning trajectories of length 20 for each of the 86 students with each of the student simulators. Then, we use DQKT-PEAR and the baseline policies to generate the recommendations and calculate the estimated reward using the reward model. Since we only simulate student's performance on a specific problem, we set the recommendation size $k = 1$ for all policies.

The experimental results (including average rewards and ±95% confidence intervals) are shown in Table 3. First, comparing DQKT-PEAR with the baseline policies, we can see that it achieves higher rewards in all three simulator models. It works well in both DQKT student simulator (aligned with the DQKT student knowledge model used in PEARL), and the different BKT and DKT simulators. This shows that DQKT-PEAR has a high potential to provide better recommendations to students in online real-world education systems. It is also the only policy that consistently outperforms the InstructSeq policy. InstructSeq is the topic sequence created by the experts that moves from basic to advanced problems. For example, if the students, whose knowledge are simulated by BKT simulator, follow PEAR recommendations, their predicted estimated post-test score according to the reward model will be on average 0.7398. So, the students who follow PEAR will have on average 0.0264 points more increase in their knowledge gain, compared to if they follow InstructSeq recommendations. This difference is statistically significant. This shows that the personalized DQKT-PEAR policy can do better than the one-size-fits-all solutions, even if they are expert-defined.

Comparing the policies that do not rely on a knowledge model with each other, we have InstructSeq > Random > InvInstructSeq. This means that the students who follow the topic sequence created by the experts have a higher estimated reward compared to the ones who follow a random problem sequence, which is expected. Also, it shows that following a random problem sequence is better than following the reverse of an expert-defined problem sequence that goes from advanced to basic topics. Compared to the knowledge model-based policies, the expert guided InstructSeq policy has a relatively good performance under different simulators.

In terms of the well-known BKT-MP method, we find that BKT-InvMP could perform better than BKT-MP. This result makes sense because BKT-InvMP tends to recommend the unanswered problems that are closest to the student mastery threshold, whereas BKT-MP tends to recommend the unanswered problems that are farthest to the mastery threshold. As a result, BKT-MP tends to recommend advanced problems that could be difficult for the student to learn from. On

Table 3. REM evaluation of DQKT-PEAR compared to baseline policies under three student simulators with trajectory length = 20.

Methods	BKT Simulator Reward	DKT Simulator Reward	DQKT Simulator Reward
Random	0.6944 ± 0.0038	0.6956 ± 0.0051	0.6935 ± 0.0035
InstructSeq	0.7134 ± 0.0012	0.7169 ± 0.0013	0.7141 ± 0.0012
InvInstructSeq	0.6671 ± 0.0012	0.6675 ± 0.0012	0.6659 ± 0.0012
BKT-MP	0.6838 ± 0.0013	0.6941 ± 0.0015	0.6926 ± 0.0014
BKT-InvMP	0.7042 ± 0.0063	0.7084 ± 0.0059	0.7127 ± 0.0043
BKT-HighestProbCorr	0.7048 ± 0.0045	0.7098 ± 0.0027	0.7112 ± 0.0022
BKT-HighestProbIncorr	0.6606 ± 0.0015	0.6689 ± 0.0016	0.6687 ± 0.0014
BKT-Uncertain	0.6593 ± 0.0023	0.6549 ± 0.0018	0.6542 ± 0.0018
DKT-HighestProbCorr	0.7253 ± 0.0018	0.7312 ± 0.0019	0.7299 ± 0.0019
DKT-HighestProbIncorr	0.6888 ± 0.0020	0.6677 ± 0.0037	0.6607 ± 0.0022
DKT-Uncertain	0.6814 ± 0.0050	0.6375 ± 0.0080	0.6561 ± 0.0069
DQKT-HighestProbCorr	0.6832 ± 0.0037	0.6598 ± 0.0070	0.7077 ± 0.0057
DQKT-HighestProbIncorr	0.7196 ± 0.0059	0.7002 ± 0.0089	0.6690 ± 0.0106
DQKT-Uncertain	0.7240 ± 0.0059	0.6925 ± 0.0103	0.6491 ± 0.0130
DQKT-PEAR	**0.7398 ± 0.0030**	**0.7459 ± 0.0021**	**0.7361 ± 0.0056**

the other hand, BKT-InvMP may be able to help students reinforce the concepts that they are partially knowledgeable about. Yet, BKT-InvMP does not perform as well as the expert guided InstructSeq policy.

The HighestProbCorr policy based on either BKT or DKT yield higher reward than HighestProbIncorr and Uncertain under different simulators. The HighestProbIncorr policy tends to recommend problems that the student is not likely to solve. These problems could be too advanced or too difficult for the student to learn from. The Uncertain policy recommends problems as a way for it to gain more information about student's knowledge, but not necessarily improve student's knowledge. But, HighestProbCorr recommends problems that have not been seen or answered correctly before, but are highly likely for the student to answer correctly. Similar to BKT-InvMP, HighestProbCorr may help students reinforce the concepts that need a deeper understanding. However, we cannot draw any conclusion by comparing HighestProbCorr to the InstructSeq policy, since HighestProbCorr is better than InstructSeq with DKT and worse than it with BKT. This performance difference could be because of the better performance of the DKT student model, compared to BKT (in Table 2). Also, we cannot draw any conclusion on the three baseline policies with DQKT, since they do not have consistent performances in all three simulators. For example, HighestProbIncorr is better than Uncertain using the DKT simulator, but worse than it under the BKT simulator.

RQ2: Learning Trajectory Length. In RQ1, we have shown that PEAR policy could achieve the highest reward under robust model-based offline evaluation, which has shown the potential effectiveness of PEAR under certain learning trajectory length. Furthermore, we would like to investigate our second research question to have a better understanding of efficacy of PEAR policy with different lengths of simulated learning trajectories. In addition to the trajectory length 20, we also simulate 10 learning trajectories of lengths $\{10, 30, 50, 100\}$ for each user, which ends up with a total of $4,300$ simulations. We report the average simulated rewards and 95% confidence interval for all competing policies under different student simulators in Tables 4, 5, 6, and 7.

As shown in the Tables 4 and 5, we observe similar results when trajectory length equals to 10, 20, and 30. The results show that PEAR policy based on the DQKT method could achieve the highest reward among all competing policies. This shows that DQKT-PEAR can be very useful for recommending problems to students at the beginning of their learning sequence, when they have not attempted most of the provided topics or problems.

Table 4. REM evaluation of DQKT-PEAR compared to baseline policies under three student simulators with trajectory length = 10

Recommendation Policy	BKT Simulator Reward	DKT Simulator Reward	DQKT Simulator Reward
Random	0.7206 ± 0.0039	0.6786 ± 0.0038	0.6727 ± 0.0040
InstructSeq	0.7128 ± 0.0012	0.7233 ± 0.0012	0.7233 ± 0.0011
InvInstructSeq	0.6717 ± 0.0013	0.6637 ± 0.0012	0.6644 ± 0.0012
BKT-MP	0.6981 ± 0.0016	0.6873 ± 0.0013	0.6830 ± 0.0013
BKT-InvMP	0.7170 ± 0.0038	0.7092 ± 0.0044	0.7046 ± 0.0052
BKT-HighestProbCorr	0.7334 ± 0.0023	0.7155 ± 0.0042	0.7160 ± 0.0042
BKT-HighestProbIncorr	0.6762 ± 0.0017	0.6607 ± 0.0013	0.6590 ± 0.0013
BKT-Uncertain	0.6563 ± 0.0020	0.6553 ± 0.0016	0.6541 ± 0.0015
DKT-HighestProbCorr	0.7491 ± 0.0017	0.7245 ± 0.0018	0.7235 ± 0.0018
DKT-HighestProbIncorr	0.6660 ± 0.0043	0.6713 ± 0.0022	0.6611 ± 0.0018
DKT-Uncertain	0.6207 ± 0.0119	0.6465 ± 0.0058	0.6434 ± 0.0053
DQKT-HighestProbCorr	0.6633 ± 0.0103	0.6694 ± 0.0028	0.6771 ± 0.0035
DQKT-HighestProbIncorr	0.7354 ± 0.0092	0.6888 ± 0.0066	0.6579 ± 0.0088
DQKT-Uncertain	0.7185 ± 0.0104	0.6784 ± 0.0083	0.6428 ± 0.0105
DQKT-PEAR	**0.7572 ± 0.0026**	**0.7313 ± 0.0029**	**0.7246 ± 0.0037**

Table 5. REM evaluation of DQKT-PEAR compared to baseline policies under three student simulators with trajectory length = 30

Recommendation Policy	BKT Simulator Reward	DKT Simulator Reward	DQKT Simulator Reward
Random	0.7140 ± 0.0038	0.7206 ± 0.0039	0.7168 ± 0.0034
InstructSeq	0.7139 ± 0.0012	0.7128 ± 0.0012	0.7130 ± 0.0012
InvInstructSeq	0.6700 ± 0.0013	0.6717 ± 0.0013	0.6694 ± 0.0012
BKT-MP	0.6876 ± 0.0016	0.6981 ± 0.0016	0.6970 ± 0.0015
BKT-InvMP	0.7087 ± 0.0070	0.7170 ± 0.0038	0.7194 ± 0.0041
BKT-HighestProbCorr	0.7359 ± 0.0033	0.7334 ± 0.0023	0.7360 ± 0.0023
BKT-HighestProbIncorr	0.6674 ± 0.0016	0.6762 ± 0.0017	0.6751 ± 0.0015
BKT-Uncertain	0.6615 ± 0.0026	0.6563 ± 0.0020	0.6598 ± 0.0021
DKT-HighestProbCorr	0.7417 ± 0.0018	0.7491 ± 0.0017	0.7482 ± 0.0016
DKT-HighestProbIncorr	0.6946 ± 0.0024	0.6660 ± 0.0043	0.6620 ± 0.0029
DKT-Uncertain	0.6967 ± 0.0063	0.6207 ± 0.0119	0.6814 ± 0.0072
DQKT-HighestProbCorr	0.6982 ± 0.0054	0.6633 ± 0.0103	0.7579 ± 0.0060
DQKT-HighestProbIncorr	0.7477 ± 0.0057	0.7354 ± 0.0092	0.6873 ± 0.0123
DQKT-Uncertain	0.7392 ± 0.0067	0.7185 ± 0.0104	0.6795 ± 0.0128
DQKT-PEAR	**0.7516 ± 0.0023**	**0.7572 ± 0.0026**	**0.7607 ± 0.0030**

However, when the length of learning trajectory increases to 50 or 100, some baseline policies catch up with DQKT-PEAR. When the learning trajectory length is 50, DQKT-based PEAR policy could still outperform other baselines, except for two other DQKT-based policies: DQKT-HighestProbIncorr and DQKT-Uncertain. Since in some simulators DQKT-PEAR outperforms DQKT-HighestProbIncorr and DQKT-Uncertain (e.g., under the DQKT simulator) and in others these baselines outperform DQKT-PEAR (e.g., under the BKT simulator), no conclusion can be drawn between these three policies integrated with DQKT. One potential reason could be that at trajectory length 50, almost half of the problems and course topics are covered by the target students. These students are likely to have an advanced understanding of the basic course concepts, a broad but average knowledge of the medium concepts, and a weak knowledge of the advanced ones. As a result, they are ready to move forward in the course concepts. In this

case, practicing the problems that are equally likely to be solved correctly or incorrectly (Uncertain), or are more difficult for the student to solve (HighestProbIncorr) could help students: Uncertain policy would help deepen the average-knowledge concepts and HighestProbIncorr would help in learning the new advanced ones. As a result, these single-objective policies, when paired with DQKT, can perform similar to the multi-objective DQKT-PEAR policy.

Table 6. REM evaluation of DQKT-PEAR compared to baseline policies under three student simulators with trajectory length = 50

Recommendation Policy	BKT Simulator Reward	DKT Simulator Reward	DQKT Simulator Reward
Random	0.7399 ± 0.0032	0.7467 ± 0.0033	0.7489 ± 0.0032
InstructSeq	0.7218 ± 0.0014	0.7169 ± 0.0013	0.7219 ± 0.0014
InvInstructSeq	0.6781 ± 0.0016	0.6834 ± 0.0017	0.6794 ± 0.0015
BKT-MP	0.6957 ± 0.0018	0.7046 ± 0.0019	0.7014 ± 0.0018
BKT-InvMP	0.7312 ± 0.0050	0.7295 ± 0.0033	0.7356 ± 0.0017
BKT-HighestProbCorr	0.7607 ± 0.0031	0.7596 ± 0.0024	0.7659 ± 0.0034
BKT-HighestProbIncorr	0.6793 ± 0.0020	0.6839 ± 0.0019	0.6809 ± 0.0018
BKT-Uncertain	0.6610 ± 0.0027	0.6557 ± 0.0023	0.6640 ± 0.0020
DKT-HighestProbCorr	0.7520 ± 0.0019	0.7694 ± 0.0016	0.7713 ± 0.0015
DKT-HighestProbIncorr	0.7063 ± 0.0044	0.6519 ± 0.0077	0.6804 ± 0.0043
DKT-Uncertain	0.7268 ± 0.0065	0.6213 ± 0.0148	0.7074 ± 0.0085
DQKT-HighestProbCorr	0.7452 ± 0.0047	0.7255 ± 0.0096	0.8149 ± 0.0014
DQKT-HighestProbIncorr	**0.7727 ± 0.0025**	**0.7798 ± 0.0039**	0.7433 ± 0.0087
DQKT-Uncertain	**0.7728 ± 0.0032**	0.7653 ± 0.0075	0.7399 ± 0.0107
DQKT-PEAR	**0.7698 ± 0.0023**	**0.7787 ± 0.0024**	**0.7844 ± 0.0040**

In addition, when the learning trajectory length is 100, as shown in Table 7, our results yield no conclusion between DQKT-PEAR with DKT-HighestProbCorr and DQKT-HighestProbCorr. Although DQKT-PEAR policy performs better than both DKT-HighestProbCorr and DQKT-HighestProbCorr under the BKT simulator, it cannot perform better than them under the DQKT simulator. At the same time, no conclusions can be made to suggest DQKT-HighestProbCorr is better or worse than DQKT-HighestProbInCorr or DQKT-Uncertain. One potential reason could be that at trajectory length 100, almost all the problems and course topics are covered by the target student. At this point, the students could have a deep knowledge of most of the course concepts. The recommendation policy can suggest from the very few problems that the student has not seen yet, but is already knowledgeable about, or the few problems that the student has answered incorrectly before. As a result, any of these policies can result in reasonable suggestions for the student. However, at 100, DQKT-PEAR still significantly outperforms the rest of baselines on all three simulators.

Another interesting observation is that the expert-designed InstructSeq policy cannot compete with DQKT-PEAR and several baselines such as BKT-HighestProbCorr and DKT-HighestProbCorr policies in all trajectory lengths. This shows the potential of personalized education policies compared to the one-size-fits-all ones. Also, as the trajectory lengths increases, the estimated reward for all policies has an upward trend. This shows that with more practice, the students can achieve higher knowledge levels. For example, even the random policy performs acceptably well at trajectory length 100, since it eventually covers most of the problems at this trajectory length. We also observe better performance on Random policy than on InstructSeq policy. It could be that in InstructSeq because the students will not move forward until they answer the current questions correctly, they can get stuck and not improve even with longer sequences. So, they won't have enough knowledge coverage. Also, it could be that when a student answers all questions correctly, when going back to review, random review can get a better recalling achievement. But our student simulators are not aware of this. However, we can see that under a constant student simulator, some policies, especially

Table 7. REM evaluation of DQKT-PEAR compared to baseline policies under three student simulators with trajectory length = 100

Recommendation Policy	BKT Simulator Reward	DKT Simulator Reward	DQKT Simulator Reward
Random	0.7778 ± 0.0028	0.7836 ± 0.0031	0.7934 ± 0.0024
InstructSeq	0.7630 ± 0.0027	0.7430 ± 0.0023	0.7644 ± 0.0024
InvInstructSeq	0.7067 ± 0.0023	0.7090 ± 0.0021	0.7313 ± 0.0026
BKT-MP	0.7276 ± 0.0029	0.7046 ± 0.0019	0.7511 ± 0.0033
BKT-InvMP	0.7480 ± 0.0041	0.7295 ± 0.0033	0.7459 ± 0.0026
BKT-HighestProbCorr	0.7750 ± 0.0022	0.7605 ± 0.0020	0.7787 ± 0.0019
BKT-HighestProbIncorr	0.6885 ± 0.0026	0.6823 ± 0.0027	0.7072 ± 0.0027
BKT-Uncertain	0.6999 ± 0.0034	0.6767 ± 0.0031	0.7237 ± 0.0029
DKT-HighestProbCorr	0.7738 ± 0.0021	**0.8015 ± 0.0017**	**0.8273 ± 0.0021**
DKT-HighestProbIncorr	0.7619 ± 0.0061	0.5992 ± 0.0180	0.8075 ± 0.0026
DKT-Uncertain	0.7723 ± 0.0044	0.6720 ± 0.0166	0.7870 ± 0.0049
DQKT-HighestProbCorr	0.7771 ± 0.0023	0.7848 ± 0.0022	**0.8523 ± 0.0014**
DQKT-HighestProbIncorr	0.7879 ± 0.0033	0.7843 ± 0.0056	0.7455 ± 0.0078
DQKT-Uncertain	0.7887 ± 0.0035	0.7695 ± 0.0072	0.7484 ± 0.0105
DQKT-PEAR	**0.7956 ± 0.0028**	**0.7981 ± 0.0030**	0.7986 ± 0.0033

DQKT-PEAR, can achieve a given reward level faster than others. For example, under the DKT simulator, DQKT-PEAR achieves the reward level 0.73 at trajectory length = 10. While for the DKT-HighestProbCorr policy a trajectory length of 20 and for the InstructSeq a trajectory length of 100 is needed to achieve a similar reward level. This means that, under the DKT simulator, DQKT-PEAR can lead to higher knowledge gain in shorter trajectories: e.g., the students following DKT-HighestProbCorr should practice twice as much as the students who follow DQKT-PEAR to achieve the same 0.73 knowledge gain.

Overall, our experiments show that PEAR can help students significantly better than the baseline policies in shorter trajectory lengths, and can help students achieve a higher knowledge gain with shorter attempt sequences. Also, as student trajectory length increases, the distinction between recommendation policies diminishes. But, the proposed PEAR policy is still robust in longer sequence lengths to provide high rewards, compared to many baseline policies.

5 CONCLUSION AND FUTURE WORK

In this paper, we proposed PEAR, Proximity-based Educational Recommendation framework, which is a multi-objective recommendation framework, inspired by the Zone of Proximal Development. PEAR provides personalized problem recommendations by balancing between estimated student knowledge improvement and problem difficulty to efficiently improve students' knowledge gain over time. It can integrate with various student and domain knowledge models. We introduced DQKT as one of these student and domain knowledge models and showed how it can be integrated with PEAR. We conducted our experiments with the REM offline evaluation method to compare our proposed recommendation policy with 11 baseline policies and showed that DQKT-PEAR is promising to improve students' knowledge at the early stages of learning, helps the students achieve higher knowledge gains in shorter trajectory lengths, and is robust in the long run as well. In the future work, we would like to deploy our proposed framework into an online education system to validate its efficiency with online A/B testings.

ACKNOWLEDGMENTS

This paper is based upon work supported by the National Science Foundation under Grant No. 2047500.

REFERENCES

[1] Tiffany Barnes, John Stamper, and Tara Madhyastha. 2006. Comparative Analysis of Concept Derivation Using the Q-Matrix Method and Facets. In *Workshop at Educational Data Mining at AAAI*. 21–30.

[2] Ahmad Baylari and Gh A Montazer. 2009. Design a Personalized E-Learning System Based on Item Response Theory and Artificial Neural Network Approach. *Expert Systems with Applications* 36, 4 (2009), 8013–8021.

[3] Yoav Bergner, Stefan Droschler, Gerd Kortemeyer, Saif Rayyan, Daniel Seaton, and David E Pritchard. 2012. Model-Based Collaborative Filtering Analysis of Student Response Data: Machine-Learning Item Response Theory. *International Educational Data Mining Society* (2012).

[4] Chih-Ming Chen, Hahn-Ming Lee, and Ya-Hui Chen. 2005. Personalized E-Learning System Using Item Response Theory. *Computers & Education* 44, 3 (2005), 237–255.

[5] Min Chi, Kurt VanLehn, Diane Litman, and Pamela Jordan. 2011. Empirically Evaluating the Application of Reinforcement Learning to the Induction of Effective and Adaptive Pedagogical Strategies. *User Modeling and User-Adapted Interaction* 21, 1-2 (2011), 137–180.

[6] Youngduck Choi, Youngnam Lee, Junghyun Cho, Jineon Baek, Byungsoo Kim, Yeongmin Cha, Dongmin Shin, Chan Bae, and Jaewe Heo. 2020. Towards an appropriate query, key, and value computation for knowledge tracing. In *Proceedings of the Seventh ACM Conference on Learning@ Scale*. 341–344.

[7] Benjamin Clement, Didier Roy, Pierre-Yves Oudeyer, and Manuel Lopes. 2013. Multi-Armed Bandits for Intelligent Tutoring Systems. *arXiv preprint arXiv:1310.3174* (2013).

[8] Albert T Corbett and John R Anderson. 1994. Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. *User Modeling and User-Adapted Interaction* 4, 4 (1994), 253–278.

[9] Shayan Doroudi, Vincent Aleven, and Emma Brunskill. 2017. Robust Evaluation Matrix: Towards a More Principled Offline Exploration of Instructional Policies. In *Proceedings of the fourth (2017) ACM conference on learning@ scale*. 3–12.

[10] Shayan Doroudi, Vincent Aleven, and Emma Brunskill. 2019. Where's the Reward? *International Journal of Artificial Intelligence in Education* 29, 4 (2019), 568–620.

[11] Aritra Ghosh, Neil Heffernan, and Andrew S Lan. 2020. Context-Aware Attentive Knowledge Tracing. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 2330–2339.

[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation* 9, 8 (1997), 1735–1780.

[13] Kenneth R Koedinger, Emma Brunskill, Ryan SJd Baker, Elizabeth A McLaughlin, and John Stamper. 2013. New Potentials for Data-Driven Intelligent Tutoring System Development and Optimization. *AI Magazine* 34, 3 (2013), 27–41.

[14] Andrew S Lan and Richard G Baraniuk. 2016. A Contextual Bandits Framework for Personalized Learning Action Selection. In *EDM*. 424–429.

[15] Andrew S Lan, Andrew E Waters, Christoph Studer, and Richard G Baraniuk. 2014. Sparse Factor Analysis for Learning and Content Analytics. *The Journal of Machine Learning Research* 15, 1 (2014), 1959–2008.

[16] Chen Liang, Jianbo Ye, Shuting Wang, Bart Pursel, and C Lee Giles. 2018. Investigating Active Learning for Concept Prerequisite Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[17] Tomasz D Loboda, Julio Guerra, Roya Hosseini, and Peter Brusilovsky. 2014. Mastery grids: An open source social educational progress visualization. In *European conference on technology enhanced learning*. Springer, 235–248.

[18] J Derek Lomas, Jodi Forlizzi, Nikhil Poonwala, Nirmal Patel, Sharan Shodhan, Kishan Patel, Ken Koedinger, and Emma Brunskill. 2016. Interface Design Optimization as a Multi-Armed Bandit Problem. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 4142–4153.

[19] Andre Martins and Ramon Astudillo. 2016. From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. In *International Conference on Machine Learning*. PMLR, 1614–1623.

[20] Shalini Pandey and George Karypis. 2019. A Self-Attentive Model for Knowledge Tracing. *arXiv preprint arXiv:1907.06837* (2019).

[21] Zachary A Pardos and Neil T Heffernan. 2009. Determining the Significance of Item Order in Randomized Problem Sets. *International Working Group on Educational Data Mining* (2009).

[22] Radek Pelánek. 2017. Bayesian Knowledge Tracing, Logistic Models, and Beyond: An Overview of Learner Modeling Techniques. *User Modeling and User-Adapted Interaction* 27, 3-5 (2017), 313–350.

[23] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. 2015. Deep Knowledge Tracing. In *Advances in Neural Information Processing Systems*. 505–513.

[24] Richard Scheines, Elizabeth Silver, and Ilya M Goldin. 2014. Discovering Prerequisite Relationships Among Knowledge Components.. In *EDM*. 355–356.

[25] Avi Segal, Yossi Ben David, Joseph Jay Williams, Kobi Gal, and Yaar Shalom. 2018. Combining Difficulty Ranking with Multi-Armed Bandits to Sequence Educational Content. In *International Conference on Artificial Intelligence in Education*. Springer, 317–321.

[26] Burr Settles and Brendan Meeder. 2016. A Trainable Spaced Repetition Model for Language Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. 1848–1858.

[27] Kikumi K Tatsuoka. 1983. Rule space: An Approach for Dealing with Misconceptions Based on Item Response Theory. *Journal of Educational Measurement* (1983), 345–354.

[28] Cem Tekin, Jonas Braun, and Mihaela van der Schaar. 2015. etutor: Online Learning for Personalized Education. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 5545–5549.

[29] Lev Semenovich Vygotsky. 1980. *Mind in Society: The Development of Higher Psychological Processes.* Harvard University Press.

[30] Chunpai Wang, Shaghayegh Sahebi, and Peter Brusilovsky. 2021. MOCHI: an Offline Evaluation Framework for Educational Recommendations.. In *Perspectives@ RecSys*.

[31] Chunpai Wang, Shaghayegh Sahebi, Siqian Zhao, Peter Brusilovsky, and Laura O Moraes. 2021. Knowledge Tracing for Complex Problem Solving: Granular Rank-Based Tensor Factorization. In *Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization*. 179–188.

[32] Chunpai Wang, Siqian Zhao, and Shaghayegh Sahebi. 2021. Learning from Non-Assessed Resources: Deep Multi-Type Knowledge Tracing. In *Proceedings of the 14th International Conference on Educational Data Mining*.

[33] Joseph Jay Williams, Juho Kim, Anna Rafferty, Samuel Maldonado, Krzysztof Z Gajos, Walter S Lasecki, and Neil Heffernan. 2016. Axis: Generating Explanations at Scale with Learner Sourcing and Machine Learning. In *Proceedings of the 3rd ACM Conference on Learning@ Scale*. ACM, 379–388.

[34] Jie Xu, Tianwei Xing, and Mihaela Van Der Schaar. 2016. Personalized Course Sequence Recommendations. *IEEE Transactions on Signal Processing* 64, 20 (2016), 5340–5352.

[35] Jiani Zhang, Xingjian Shi, Irwin King, and Dit-Yan Yeung. 2017. Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th international conference on World Wide Web*. 765–774.

[36] Siqian Zhao, Chunpai Wang, and Shaghayegh Sahebi. 2020. Modeling Knowledge Acquisition from Multiple Learning Resource Types. In *Proceedings of The 13th International Conference on Educational Data Mining*. 313–324.