Leveraging Student Profiles and the Mamba Framework to Enhance Knowledge Tracing

Mingxing Shao[1], Tiancheng Zhang[1] ✉, Minghe Yu[2], Zhenghao Liu[1], Yifang Yin[3], Hengyu Liu[4], and Ge Yu[1]

[1] School of Computer Science and Engineering, Northeastern University, Shenyang, China
{2301935@stu.neu.edu.cn, tczhang@mail.neu.edu.cn, liuzhenghao@mail.neu.edu.cn, yuge@mail.neu.edu.cn}
[2] College of Software, Northeastern University, Shenyang, China
yuminghe@mail.neu.edu.cn
[3] Institute for Infocomm Research, A*STAR, Singapore
yin_yifang@i2r.a-star.edu.sg
[4] Department of Computer Science, Aalborg University, Aalborg, Denmark
heli@cs.aau.dk

Abstract. Knowledge tracing (KT) predicts students' knowledge mastery based on their interaction history to forecast future performance. Although current KT methods have achieved good results, because of the lacking of students' information, these methods can only make sequence-level inference, assuming that students are independent and homogeneous. Additionally, due to the typically long student sequences in KT tasks, mainstream RNN-based student state modeling methods suffer from long-sequence forgetting, while attention-based models require manually set bias functions. To address these issues, this paper proposes a **D**ual-**G**raph **M**amba framework for **K**nowledge **T**racing (DGMKT), which models student profiles based on students' interaction sequence through a **D**ual-**G**raph **S**tudent-**P**rofile Aware **M**odule (DGSPM). Meanwhile, we model student mastery states based on Mamba, avoiding the long-sequence forgetting problem in RNN-based models and the need for bias functions in attention-based models for KT tasks. To the best of our knowledge, this is the first application of the Mamba architecture in KT tasks. We evaluate DGMKT on four datasets and compare it with ten baselines to demonstrate its superiority. Furthermore, we showcase its broad adaptability by integrating DGSPM with various KT models.

Keywords: Knowledge Tracing · Student Profile · Mamba Structure

1 Introduction

With the rapid growth of computer science and cognitive diagnosis, online education platforms like MOOCs, EDX, and Coursera have become increasingly popular, generating vast amounts of student learning data daily. Using the data to evaluate students' knowledge and recommend appropriate exercises is a major challenge for these platforms.

Knowledge tracing (KT) addresses this by tracking students' mastery of knowledge points to predict their future performance. Most knowledge tracing models are primarily based on Recurrent Neural Network and Self-Attention mechanism. Although some researchers have explored other neural networks such as [9,24], the mainstream research still focuses on the exploration of DNN (Deep Neural network) applied to knowledge tracing tasks.

However, most existing DNN-based studies lack a unique identifier for each student. These models only receive interaction sequence information and don't know the student-specific details corresponding to the sequence, so they can only make sequence-level inferences. As shown in Figure 1, although the two students' sequences are the same before time step 6, as distinct individuals, they are likely to provide different answers at step 6. Existing models often assume that the two students will provide the same response at time step 6 in this case.

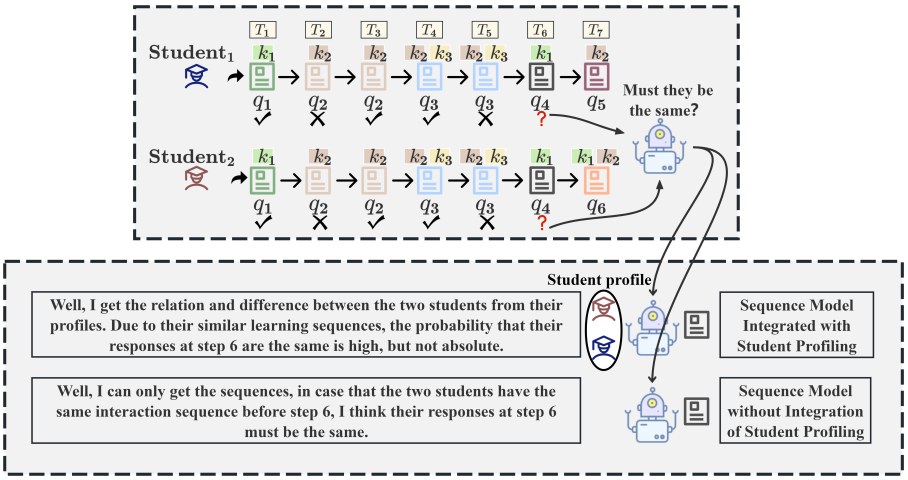


Fig. 1: An illustration comparing sequence modeling methods with and without the integration of student profiling.

Therefore, we propose the definition of the student profile in the context of knowledge tracing tasks: **The student profile serves as a unique identifier for students, allowing for the distinction between different students while also enabling the aggregation of information across similar students**. In contrast to recent studies that focus on generating high-quality embeddings for individual items or concepts, our proposed method can be understood as generating students' identifier based on the sequence and frequency of interactions with items by students. These identifiers are then used to guide the model in tracking the students' knowledge states throughout the entire process.

Another issue is that since the student interaction sequences in knowledge tracing tasks are usually quite long, knowledge tracing models based on RNN

[22,26,15,16] and Self-Attention mechanism [18,2,6,19,13] inevitably suffer from performance issues due to inherent limitations in their architectures, such as the forgetting problem in RNN caused by vanishing gradients and the need for manually defined bias functions in Self-Attention-based architectures. In 2023, Gu et al. [3] proposed the Mamba architecture, which has been demonstrated to perform well in long-sequence tasks [28]. Due to the long-sequence issue in KT tasks, this characteristic makes Mamba particularly suitable for application in knowledge tracing tasks.

To address these issues, we propose the DGMKT framework, which consists of three components: the **D**ual-**G**raph **S**tudent **P**rofile Aware **M**odule (DGSPM) and the **M**amba **S**equence **M**odeling **M**odule (MSMM) and the Integration Module. The DGSPM generates two student profiles for each student through their interaction sequence. Specifically, the dual-graph in DGSPM includes a Student-Exercise Association Hypergraph (SEAHG) and an Exercise-Directed Transition Graph (EDTG). For students' interaction sequences, SEAHG is designed to capture what exercises students interact with, without considering the order of interactions. As a complement, EDTG is specifically designed to capture the sequential order in which students interact with the exercises. In SEAHG, Define a student as a node, an exercise as a hyperedge, since each student can interact with an exercise many times, so we allow a hyperedge to connect the same node many times. as shown in Figure 2, given sequences of two students as in Figure 2(a), Construct the SEAHG as in Figure 2(b). In EDTG, an exercise is defined as a node, and a transition between two exercises is represented as a directed edge. Based on the sequence of $student_2$ in Figure 2(a), the corresponding EDTG is constructed as shown in Figure 2(c).

The Mamba Sequence Modeling Module consists of two components: the linear layer projects input features to the question embedding dimension, and the Mamba layer captures sequential dependencies to model students' states.

Finally, after obtaining the predictions of the two sets of linear-Mamba layers, In Integration Module, we integrate the two predictions through a method similar to the online knowledge distillation proposed by DGEKT [1].

In summary, our contributions are as follows:

- We propose the definition of student profiles and introduce a Dual-Graph Student Profile Aware Module (DGSPM) for modeling student profiles in Knowledge Tracing tasks through students' interaction sequences.
- We propose Mamba Sequence Modeling Module (MSMM) to model students' knowledge states, adapting to the characteristics of long sequences in knowledge tracing tasks.
- We demonstrate the effectiveness and adaptability of the proposed DGSPM in modeling student profiles by integrating it with different knowledge tracing methods. Additionally, we showcase the superiority of the proposed MSMM in modeling students' mastery compared to mainstream RNN-based and Self-Attention-based modeling approaches.

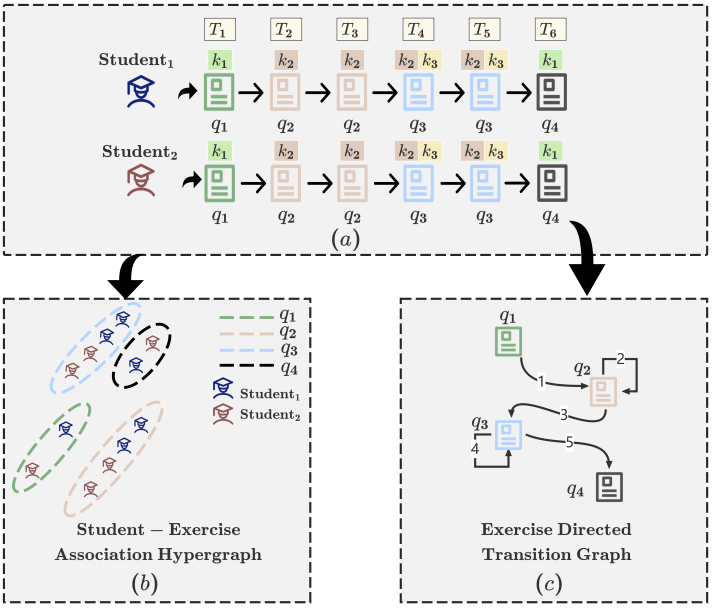


Fig. 2: The process of constructing SEAHG and EDTG through students' interaction sequence.

2 Related Work

2.1 Knowledge Tracing

The development of knowledge tracing methods can be divided into two stages: the probabilistic graphical model-based stage and the Deep Neural Network-based stage. Among probabilistic models, Bayesian Knowledge Tracing (BKT) is one of the most representative models. BKT is a Hidden Markov Model that models a learner's latent knowledge state as a set of binary variables, where each variable represents whether a student has mastered a particular knowledge concept. Since BKT assumes the homogeneity of both students and problems, researchers have gradually incorporated personalized components into BKT to enhance its effectiveness [7,25]. Additionally, some researchers have made personalized improvements to BKT's skill-specific parameters [20,21] and student-specific parameters [29].

With the advent of DNN, the first DNN-based knowledge tracing model [22] made traditional models less competitive. Modern research focuses on RNN-based [22,16] and Self-Attention-based [19,18,2] KT models, as well as hybrid architectures [4,24,13]. However, beacause the length of sequence in KT tasks is long, RNN often suffer from forgetting issues [4], and Self-Attention models require manually designed bias functions, which affect performance [19,2,6].

To address these limitations, we model student mastery states using Mamba, which has been proven effective for long-sequence tasks [28] and does not require the setting of a bias function.

2.2 Student Profile

While student profiling is not commonly discussed in Knowledge Tracing tasks, it is frequently explored in areas like academic failure prediction. For example, Liang et al. [12] propose a student profiling method based on online learning behavior data. This method analyzes learning features, behavioral similarity, and learning attitudes to enhance the quality of student learning and the level of personalized services in E-Learning platforms through intelligent guidance. Khasanah et al. [8] employ a feature selection and classification algorithm approach, using selected features as the basis for student profiling to predict the likelihood of academic failure. Shen et al. [23] construct student profiles by combining labels that describe students' personal information and learning behaviors with the code they submitted, and used these profiles to predict their scores in programming tests.

Although effective, these methods often depend on personal information like age or economic status, which is unavailable in traditional Knowledge Tracing tasks that only provide interaction sequences. To address this issue, we propose a method capable of modeling student profiles based on their response sequences. The inspiration for this approach stems from the observation that students with similar response sequences often share analogous learning factors, such as learning pace and prior knowledge. Moreover, the correctness of their responses tends to be more similar.

2.3 Mamba

As a highly promising neural network architecture, Mamba is similar to RNN in its recursive information propagation. However, it avoids the forgetting issues commonly associated with RNN. Additionally, its parameterized input structure dynamically determines the importance of input components without the need for manually setting bias functions, as required by attention mechanism.

Mamba has demonstrated competitive performance and has even achieved state-of-the-art results in various tasks, such as biomedical image segmentation [14], pan-sharpening [5], computer vision [31], and natural language processing [3]. Notably, Mamba's advantages become particularly evident in tasks involving long sequences. These achievements suggest that modeling the student mastery state in knowledge tracing task as a new application of Mamba could significantly improve the performance of knowledge tracing models.

3 Method

Given a student s_k's exercise sequence up to time $t-1$, denoted as $E_k = \{e_1, e_2, ..., e_{t-1} \mid i = 1, \ldots, t-1\}$, where $e_i \in \mathcal{E}$ represents the exercise attempted

by the student at time i, and $\mathcal{E}$ denotes the set of all exercises with cardinality $|\mathcal{E}|$. The SEAHG and EDTG are constructed as shown in Figure 2. With the two graphs, we will get two student profiles through the Hypergraph Convolution and the Directed Graph Convolution illustrated in Section 3.1. Then, given E_k and the corresponding response sequence $A_k = \{a_1, a_2, ..., a_{t-1} \mid i = 1, \ldots, t-1\}$ where a_i indicates the correctness of the response at time i: $a_i = 1$ denotes a correct response and $a_i = 0$ denotes an incorrect response, and the two student s_k's profiles, we will get s_k's two sets of knowledge mastery state at each time step through the Mamba Sequence Modeling Module illustrated in Section 3.2. Finally, we integrate the two sets of knowledge mastery state through the Integration Module illustrated in Section 3.3 and get the final predictions of time t: $P(a_t = 1 \mid I_k, e_t, s_k)$. The above process can be illustrated in Figure 3.

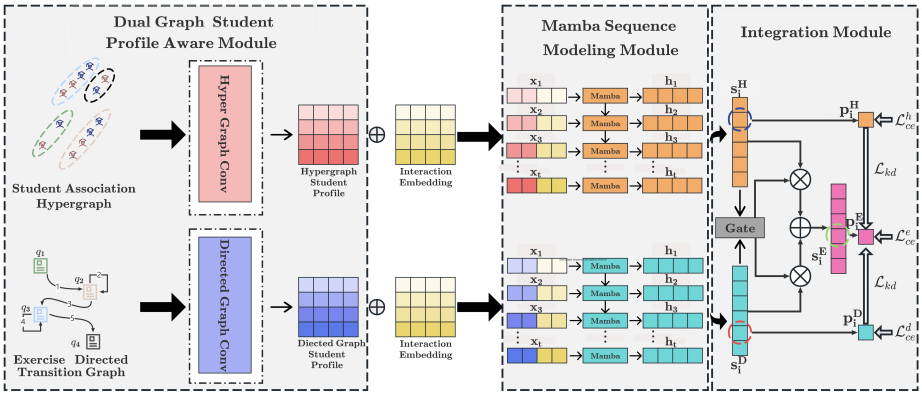


Fig. 3: The overall framework of proposed DGMKT. Student profiles (sp) are generated from DGSPM. After concatenating with interaction embeddings (ie), sp $\oplus$ ie is passed through the Mamba Sequence Model to generate predictions, which will be sent to the Integration Module.

3.1 Dual-Graph Student-Profile Aware Module

Student-Exercise Association Hypergraph In student association hypergraph, we consider the student set $\mathcal{S}$, with a sample student represented by s_i and $|S| = n$. Similarly, we define an exercise set $\mathcal{E}$ with a sample exercise e_i and $|\mathcal{E}| = m$. To avoid confusion, each student in the hypergraph is represented by a node v_i, and each exercise by a hyperedge h_j. Since a student may complete the same exercise multiple times, we allow hyperedges to connect to the same nodes multiple times.

Based on this, let H_i denote the set of hyperedges connected to a node v_i, with $|H_i| \leq m$, and let h_i^o represent the number of times this node connects to

each hyperedge h_o. The degree of a node can thus be defined as $d_v^i = \sum_{o=1} h_i^o$, representing the total number of exercises completed by student s_i.

Next, we define V_j as the set of nodes connected by hyperedge h_j, with $|V_j| \leq n$, and let v_j^k represent the number of times a particular node v_k is connected by the hyperedge. The degree of a hyperedge h_j is then given by $d_h^j = \sum_{k=1} v_j^k$, reflecting the total number of interactions with the exercise.

With these definitions, we employ an information propagation rule defined by hypergraph convolutional networks. The convolution operator in each layer aggregates information from v_i itself and from its local neighbors within each hyperedge to which v_i is connected, thereby updating x_i:

$$x_{i,H}^{(l)} = \sigma\Big(\sum_{h_j \in H_i} \frac{1}{d_h^j} \sum_{v_k \in V_j} \frac{1}{\sqrt{d_v^k d_v^i}} \Theta^{l-1} x_{i,H}^{(l-1)}\Big) \tag{1}$$

Where $\Theta \in \mathbb{R}^{d_{model} \times d_{model}}$ is a learnable weight parameter, and σ is the ReLU activation function. $x_{i,H}^{(l-1)} \in \mathbb{R}^{1 \times d_{model}}$ and $x_{i,H}^{(l)} \in \mathbb{R}^{1 \times d_{model}}$ represent the input and output embeddings of node v_i in the l-th layer, respectively.

Thus, we obtain the profile $x_{i,H}$ for student s_i generated by the hypergraph, which primarily captures information on which exercises the student interacted with.

Exercise Directed Transition Graph In Directed Graph, let the set of exercises be denoted as $\mathcal{E}$, where $|\mathcal{E}| = m$. For each student s_k, the interaction sequence $\mathcal{I}_k$ includes several tuples $(e_x, a_x) \in \mathcal{I}_k$, representing exercises and responses. From this, we obtain the student's exercise sequence $E_k = \{e_1, e_2, e_3, \ldots, e_L\}$. This sequence can be broken down into a series of exercise pairs, such as $\bar{E}_k = \{(e_1, e_2), (e_2, e_3), \ldots, (e_{L-1}, e_L)\}$. In student s_k directed graph, we represent all exercises $\mathcal{E}$ as nodes $\mathcal{V}$ and each individual exercise pair in a student's sequence $\bar{E}_k$ as directed edges $h_{i,j}$. A separate directed graph is constructed for each student, where $h_{i,j}$ represents that exercise e_i is immediately followed by e_j. We use $g_{i,j}$ to denote the number of times the pair (e_i, e_j) appears. The adjacency matrix A of this directed graph is defined as follows:

$$A_{i,j} = \begin{cases} 1 \times g_{i,j} & \text{if } h_{i,j} \text{ exists,} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Next, we add self-loops to the adjacency matrix by setting $\hat{A}_{i,i} = A_{i,i} + 1$, thereby constructing the matrix $\hat{A}$. In $\hat{A} \in \mathbb{R}^{m \times m}$, we treat $\hat{A}$ as an adjacency graph where the degree of node $v_i \in \mathcal{V}$ is defined as $d_v^i = \sum_{j=1}^{m} A_{i,j}$. From matrix $\hat{A}$, we obtain the degree matrix $\hat{D} \in \mathbb{R}^{m \times m}$ and apply the following formula for graph convolution:

$$\mathbf{x}_{k,D}^{(l)} = \sigma\left(\hat{\mathbf{D}}^{-\frac{1}{2}} \hat{\mathbf{A}} \hat{\mathbf{D}}^{-\frac{1}{2}} \mathbf{x}_{k,D}^{(l-1)} \mathbf{W}^{(l-1)}\right), \tag{3}$$

where $W^{(l)} \in \mathbb{R}^{d_{model} \times d_{model}}$ denote a learnable weight matrix, and let σ represent the ReLU activation function. Specifically, $x_{k,D}^{l-1}$ and $x_{k,D}^{l} \in \mathbb{R}^{m \times d_{model}}$ are the input and output embeddings of node set $\mathcal{V}$, respectively.

Using the student's exercise sequence E_k, we index these embeddings to retrieve the convolutional representations of the exercises $\hat{x}_{k,D} \in \mathbb{R}^{L \times d_{model}}$ in the sequence:

$$\hat{x}_{k,D} = E_k \rightarrow x_{k,D}. \tag{4}$$

For a student s_k with an interaction sequence length of L, we introduce a set of learnable weights $A = \{\alpha_1, \alpha_2, \ldots, \alpha_L\}$, $A \in \mathbb{R}^L$ as the weighting parameters. After applying the softmax function, these weights are multiplied by the obtained sequence embeddings $\hat{x}_{k,D}$ to get the student profile of directed graph:

$$\hat{A} = \text{softmax}(\mathbf{A}) = \left[\hat{\alpha}_1 = \frac{\exp(\alpha_1)}{\sum_{i=1}^{L} \exp(\alpha_i)}, \ldots, \hat{\alpha}_L = \frac{\exp(\alpha_L)}{\sum_{i=1}^{L} \exp(\alpha_i)} \right], \tag{5}$$

$$\tilde{x}_{k,D} = \sum_{o=1}^{L} \hat{\alpha}_o \cdot \hat{x}_{k,D}^{o}. \tag{6}$$

Thus, we derive the representation $\tilde{x}_{k,D} \in \mathbb{R}^{d_{model}}$ for student s_k, generated from the directed graph, which captures the ordering of exercises completed by the student.

3.2 Mamba Sequence Modeling Module

At this stage, we have obtained the hyper graph student profile $x_{i,H}$ and directed graph student profile $\tilde{x}_{i,D}$ for student s_i. Next, we concatenate $x_{i,H}$ and $\tilde{x}_{i,D}$ respectively with the interactive embedding x_i for further analysis:

$$\begin{aligned} x_{input,i}^{H} &= x_{i,H} \oplus x_i, \\ x_{input,i}^{D} &= \hat{x}_{i,D} \oplus x_i. \end{aligned} \tag{7}$$

The interactive embedding x_i is derived from the exercise sequence $E_i = \{e_1, e_2, \ldots, e_L\}$ completed by student s_i and the corresponding response sequence $A_i = \{a_1, a_2, \ldots, a_L\}$, which are processed through an embedding layer to obtain x_i^e and x_i^a. These are then concatenated as described below:

$$x_i = \begin{cases} x_i^e \oplus x_i^a, & \text{if } a_i = 1, \\ x_i^a \oplus x_i^e, & \text{if } a_i = 0, \end{cases} \tag{8}$$

where the embeddings $x_{input,i}^{H} \in \mathbb{R}^{3d_{model}}$ and $x_{input,i}^{D} \in \mathbb{R}^{3d_{model}}$ are each passed through a linear layer and a Mamba layer to obtain the knowledge mastery state of the student at each time step, denoted as $h_i^H \in \mathbb{R}^{L \times d_{model}}$ and $h_i^D \in \mathbb{R}^{L \times d_{model}}$:

$$\begin{aligned} h_i^H &= Mamba_1(Linear_1(x_{input,i}^{H})), \\ h_i^D &= Mamba_2(Linear_2(x_{input,i}^{D})), \end{aligned} \tag{9}$$

where Mamba1 and Mamba2 represent two distinct Mamba layers, while Linear1 and Linear2 are two different linear layers with weight matrices shaped as $\mathbb{R}^{3d_{model} \times d_{model}}$. The execution process of the Mamba block is illustrated in Figure 4.

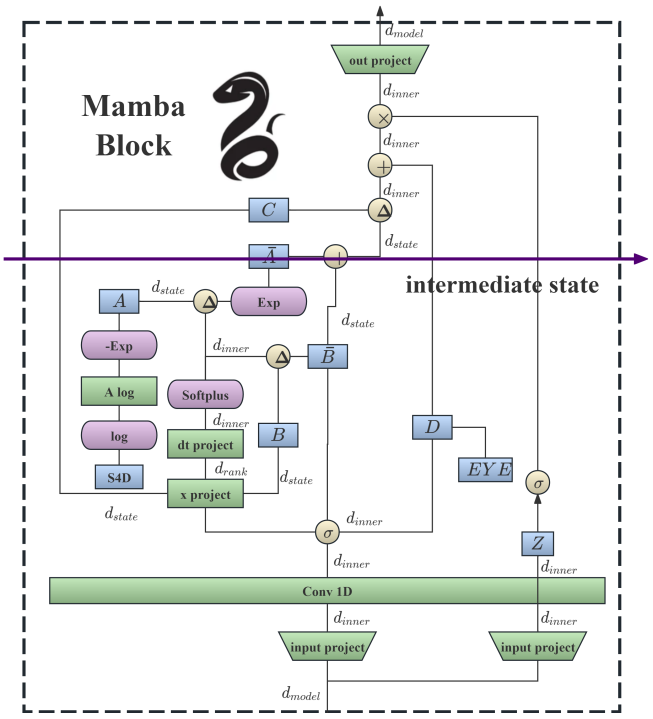


Fig. 4: The detail of Mamba Layer, Where **xx project** represents a simple linear layer, **Conv 1D** refers to a one-dimensional convolution. **S4D** stands for "shape for dimension," indicating a one-dimensional tensor to be expanded. The σ function corresponds to the SiLU activation function, while Δ and $\times$ denote matrix multiplication, and $+$ represents matrix addition. Finally, **EYE** is used to initialize a tensor with ones.

3.3 Integration Module

To further integrate the results produced by the two graphs, we employ the online knowledge distillation integration method proposed in DGEKT. Through a gating mechanism, h_i^H and h_i^D are combined into h_i^E:

$$h_i^E = g \odot h_i^H + (1 - g) \odot h_i^D, \tag{10}$$

$$g = \sigma\left((W_H h_i^H + b_H) + (W_D h_i^D + b_D)\right), \tag{11}$$

where $W_H, W_D \in \mathbb{R}^{d_{model} \times d_{model}}$, $b_H, b_D \in \mathbb{R}^{d_{model}}$ are learnable parameters, σ is a *sigmoid* function, $\odot$ is a simple multiplication.

These combined representations are then mapped to the problem dimension through three linear layers with weight matrices shaped as $\mathbb{R}^{d_{model} \times d_c}$, yielding $logitD$, $logitH$, and $logitE \in \mathbb{R}^{L \times d_c}$:

$$\begin{aligned} logit_i^H &= Linear_3(h_i^H), \\ logit_i^D &= Linear_4(h_i^D), \\ logit_i^E &= Linear_5(h_i^E). \end{aligned} \tag{12}$$

Given $logit_i^D$, $logit_i^H$, and $logit_i^E$, we apply the sigmoid activation function to each, resulting in the final predictions of the three models: p_i^d, p_i^h, and p_i^e. The equations are as follows:

$$\begin{aligned} p_i^D &= \frac{1}{1 + e^{-logit_i^D}}, \\ p_i^H &= \frac{1}{1 + e^{-logit_i^H}}, \\ p_i^E &= \frac{1}{1 + e^{-logit_i^E}}. \end{aligned} \tag{13}$$

Subsequently, to encourage the predictions of the two single-graph models to align more closely with those of the dual-graph teacher model, we follow the approach in DGEKT by using the L_1-norm to measure the discrepancy between the teacher and student models:

$$\mathcal{L}_{kd} = \frac{1}{n} \sum_{i=1}^{n} \left(\|p_i^E - p_i^H\|_1 + \|p_i^E - p_i^D\|_1 \right). \tag{14}$$

During training, we optimize the losses of the two single-graph models, denoted as $\mathcal{L}_{ce}^h$ and $\mathcal{L}_{ce}^d$, along with the loss of the dual-graph integrated model $\mathcal{L}_{ce}^e$, as well as a distillation loss controlled by a constant λ. Unlike in DGEKT, here λ is fixed as $1/(\text{batchsize} \times \text{seqlen})$, where seqlen is the length of the student's response sequence. During testing, we use the average of the predictions from the three models as the final result, instead of using only the teacher model as the final result in DGEKT [1]:

$$\mathcal{L} = \mathcal{L}_{ce}^h + \mathcal{L}_{ce}^d + \mathcal{L}_{ce}^e + \lambda \mathcal{L}_{kd}, \tag{15}$$

$$p_i = \frac{(p_i^E + p_i^D + p_i^H)}{3}. \tag{16}$$

Notably, in this paper, $\mathcal{L}_{ce}$ represents the cross-entropy loss function, which can be calculated as:

$$\mathcal{L}_{ce} = -\frac{1}{L} \sum_{l=1}^{L} (y_i^l \log(p_i^l) + (1 - y_i^l) \log(1 - p_i^l)), \tag{17}$$

where y_i^l represents the correctness of student s_i's actual response at time step l, and p_i^l denotes the model's prediction of the correctness of student s_i's response at time step l. L denotes the length of student s_i's interaction sequence.

4 Experiments

4.1 Experimental Setting

Benchmark Datasets In the experiment, we used four datasets to evaluate our model and the proposed methods. The descriptions of the four datasets are as follows:

Statics2011[1]: The Statics2011 dataset was collected from a collegelevel engineering course on statics containing 189,927 responses with 1223 KSs from 333 students

Kddcup2010[2]: This dataset was originally used for the 2010 kdd cup competition. It has 868,410 responses with 660KSs from 690 students.

Assist2017 [3]: This dataset is gathered from the ASSISTments online tutoring platform, which contains 525,637 responses with 110 KSs from 4151 students.

Assist2009 [4]: This dataset is also from the same platform as Assist2009. It has 942,816 responses with 102 KSs from 1709 students.

The detailed information for each dataset is provided in Table 1. We use the preprocessed versions of these four datasets provided by the literature on AKT [2] and DKT [22] for fair comparison.

Table 1: Statistics of the four benchmark datasets.

Dataset Name	Students	KSs	Responses	Res.per.stu
Kddcup2010	690	660	868,410	1,258.56
Statics2011	333	1,223	189,297	568.45
Assist2009	5,151	110	325,637	63.21
Assist2017	1,709	102	942,816	551.67

Baseline Methods and Evaluation Metric We compared the proposed model with the following seven state-of-the-art models:

DKT [22]: The first work to apply deep neural networks (DNN) for knowledge tracing, using RNN to model students' knowledge states.

DKVMN [30]: Utilizing a key-value memory network, where the key matrix stores static knowledge concepts and the value matrix tracks the student's mastery level, providing a degree of interpretability.

SAKT [18]: The first model to use an attention network for knowledge tracing, predicting mastery by identifying relevant parts of the student's past interactions associated with the current knowledge concept.

[1] https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=507
[2] https://pslcdatashop.web.cmu.edu/KDDCup
[3] https://sites.google.com/view/assistmentsdatamining
[4] https://sites.google.com/site/assistmentsdata/home/2009-2010-assistment-data

AKT [2]: An attention-based method proposes a monotonic attention mechanism to capture the connections between a student's current and previous interactions.

KQN [9]: Modeling knowledge interaction as the dot product of the knowledge state and skill vectors, with neural networks encoding student responses and skills into vectors of equal dimensions.

GKT [17]: Using graph convolutional networks on a graph of knowledge concepts, constructed through statistical or learning-based methods, to model students' proficiency in each concept.

FoLiBiKT [6]: Building on attention networks with a linear bias to model student forgetting behaviors.

DTransformer [27]: Proposing a stable and truly effective framework for tracking students' knowledge status.

StableKT [10]: Proposing StableKT to enhance length generalization in knowledge tracing tasks. It captures the relationships between questions and knowledge components through a multi-head aggregation module.

ExtraKT [11]: Proposing a framework, which improves length extrapolation capability by negatively biasing attention scores.

These methods provide a strong baseline for evaluating the DGMKT model. Following standard metrics in knowledge tracing, we use AUC and ACC to measure prediction performance.

Implementation Details We conducted 5-fold cross-validation on all datasets with a 3:1:1 split for training, validation, and test sets. For fairness, we compared the baserline variant that uses knowledge skill (KS) information, as our model and all baselines rely exclusively on KS information.

In the experiments, we set exercise and response embedding dimensions to 512 and used four Mamba layers. Sequences longer than 500 were truncated. Training used a batch size of 24 across all datasets, the Adam optimizer with an initial learning rate of 0.001 and a decay factor of 0.5, for up to 500 epochs. Early stopping was applied if validation loss did not improve for five epochs. All experiments were implemented in PyTorch on two NVIDIA GeForce RTX 3080 GPUs.

4.2 Comparison with Baselines

In Table 2, we present the prediction performance of various methods, showing the averages and standard deviations across five test folds. It is evident that the proposed DGMKT consistently outperforms all other methods across all datasets.

4.3 Adaptability Study of DGSPM in Different Architecture

To validate the Adaptability of our proposed student profiling method, we integrated the DGSPM into Self-Attention-based knowledge tracing models as

Table 2: Performance comparison across different models and datasets. The best AUC value in each dataset is highlighted in **bold**, and the second-best in *italic*.

Dataset	**Statics2011**		**ASSIST2009**		**ASSIST2017**		**kddcup2010**	
Model	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC
DKT	0.8106	0.7965	0.8023	0.7609	0.7052	0.6807	0.7874	0.8341
DKVMN	0.7966	0.7951	0.7314	0.7185	0.6704	0.6691	0.7823	0.8355
SAKT	0.8022	0.7975	0.7361	0.7205	0.6492	0.6607	0.7736	0.8272
AKT	*0.8251*	0.8045	0.8053	0.7675	0.6918	0.6821	0.7898	0.8318
FoLiBiKT	0.8232	0.8044	0.8004	0.7643	0.6882	0.6783	0.7917	0.8283
GKT	0.7997	0.7982	0.7708	0.7472	0.6773	0.6720	0.7737	0.8316
KQN	0.8245	0.8041	*0.8107*	*0.7688*	*0.7200*	*0.6889*	*0.7956*	*0.8390*
DTrans	0.8202	0.8044	0.7865	0.7538	0.6859	0.6774	0.7867	0.8323
StableKT	0.8250	*0.8052*	0.8059	0.7658	0.6963	0.6805	0.7943	0.8320
ExtraKT	0.8215	0.7977	0.8090	0.7670	0.7006	0.6814	0.7947	0.8316
DGMKT	**0.8261**	**0.8058**	**0.8180**	**0.7722**	**0.7339**	**0.6966**	**0.7986**	**0.8391**

well as the RNN-based model. We then compared the AUC performance with dual-graph (model with DGSPM) and no-graph (model) configurations under identical parameter settings. The results are shown in Table 3. It can be observed that SAKT and DKT with DGSPM acheive better performance in all of four datasets than their original versions.

Table 3: The performance of proposed DGSPM in RNN-Based method (DKT) and Attention-Based method (SAKT).

Dataset	**statics2011**		**Assist2009**		**Assist2017**		**kddcup2010**	
Model	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC
DKT	0.8106	0.7965	0.8023	0.7609	0.7052	0.6807	0.7874	0.8341
DKT^{DGSPM}	**0.8209**	**0.8028**	**0.8102**	**0.7661**	**0.7183**	**0.6878**	**0.7960**	**0.8378**
SAKT	0.8022	0.7975	0.7361	0.7205	0.6492	0.6607	0.7736	0.8272
$SAKT^{DGSPM}$	**0.8108**	**0.8024**	**0.7453**	**0.7246**	**0.6561**	**0.6646**	**0.7852**	**0.8307**

4.4 Ablation Study of DGSPM and MSMM

Dual-Graph Student Profile Aware Module We validate the effectiveness of the proposed component Dual-Graph Student Profile Aware Module (DGSPM) by comparing its different variants.

The results presented in Figure 5 demonstrate that the model with the complete proposed method outperforms all ablation models, thereby validating the effectiveness of the proposed DGSPM.

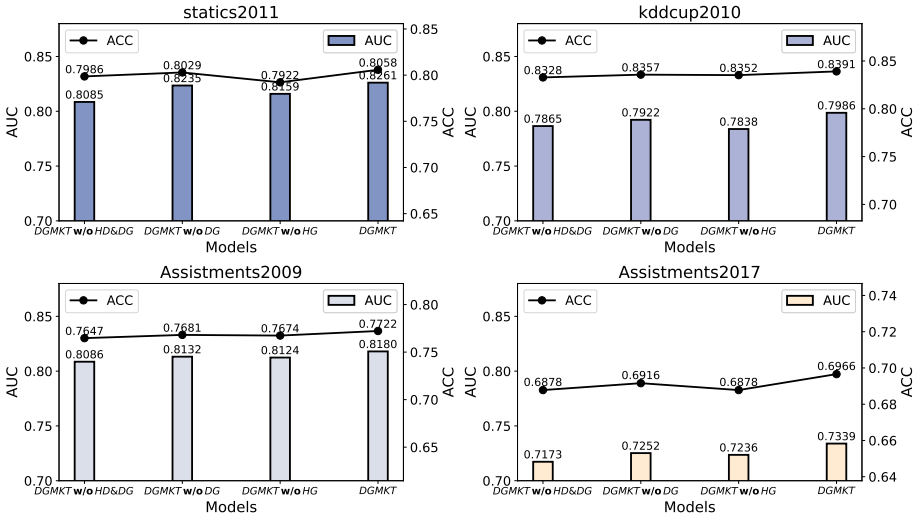


Fig. 5: Different variants of DGSPM are proposed to examine the effectness of DGSPM. Retaining only the SEAHG in DGSPM (DGMKT w/o DG), only the EDTG in DGSPM (DGMKT w/o HG), or removing both of the two graphs in DGSPM (DGMKT w/o HG&DG), as well as the complete proposed approach (DGSPM), while keeping other parameters constant.

Mamba Sequence Modeling Module To validate the effectiveness of the proposed component Mamba Sequence Modeling Module (MSMM), we compare Mamba sequence modeling method with RNN-based and Self-Attention-based sequence modeling method. Similarly, we keep other parameters constant, and compare three kinds of methods (Mamba-based, RNN-based, Self-Attention-based) with and without DGSPM. The results in Figure 6 demonstrates that Mamba as an approach for modeling student learning states, demonstrates superior performance compared to the mainstream RNN and Self-Attention methods.

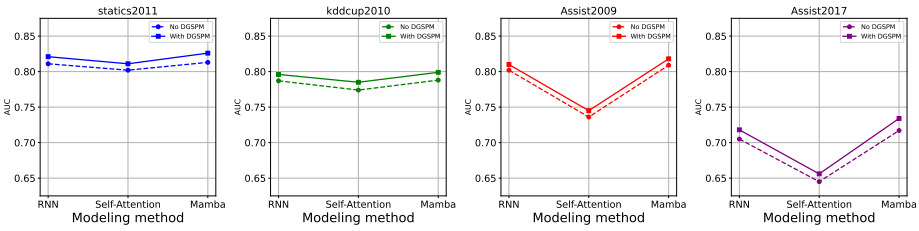


Fig. 6: The performance of the proposed Mamba Sequence Modeling Module compared with RNN-based and Self-Attention based sequence modeling method.

4.5 Computational Efficiency Analysis of DGSPM

To rigorously evaluate the computational resource demands of the proposed DGSPM, we report the resource consumption profiles of various DGMKT variants on the Assist2017 dataset, as detailed in Table 4. To facilitate computational efficiency and ensure comparability, we configure the number of Mamba layers to a single layer (n_layer = 1), while maintaining the embedding dimension at 512, consistent with prior experimental settings.

It can be observed that, in terms of training time, the variant incorporating only directed graphs exhibits training durations comparable to the comprehensive method, both significantly exceeding that of the variant employing hypergraphs. This disparity arises because the Graph Convolutional Network (GCN) necessitates the generation of a distinct directed graph for each student. The sequential processing of individual students decomposes batch operations into serial computations, thereby constraining the parallel computing capabilities of the GPU.

Regarding memory consumption, the variant utilizing solely hypergraphs incurs substantially higher GPU memory usage compared to the variant with directed graphs. This is attributed to the representation of students as nodes and exercises as hyperedges, which requires the construction of a hypergraph structure. Such a structure may encompass millions of connections, forming a complex hypergraph. The hypergraph convolution process involves aggregating information across all connected nodes and hyperedges, resulting in significant GPU memory demands.

Although the dual-graph structure entails additional computational resources, it is noteworthy that the computational overhead of the dual-graph approach remains within a reasonable range.

Table 4: Computational resource requirements of the four DGMKT variants

Model	Parameter	GPU Usage	Training Time
DGMKT	26.75 MB	3919 MB	2458.2 s
DGMKT w/o DG	12.14 MB	2515 MB	271.4 s
DGMKT w/o HG	9.17 MB	1571 MB	1803.4 s
DGMKT w/o HG&DG	9.13 MB	1473 MB	107.7 s

4.6 Model Visualization

To evaluate the performance of the proposed DGMKT model in knowledge tracing, we visualize a student's mastery of various knowledge skills (KS) during 36 exercises, as shown in Figure 7. The top section records the student's performance, with different colors representing the KSs involved in each exercise. The middle heatmap shows the mastery levels of each knowledge point (KS) over time, where darker colors indicate higher mastery levels.

Initially, all KS mastery levels are updated after each exercise. Correct answers increase the mastery level of relevant KSs, while incorrect answers decrease it. For example, after correctly answering exercises related to "finding-percents" at step 11, the mastery level for this KS increases in the heatmap. Changes in other rows highlight potential connections between KSs, where mastering one skill can influence others. More model visualization experiments can be found in Appendix A.

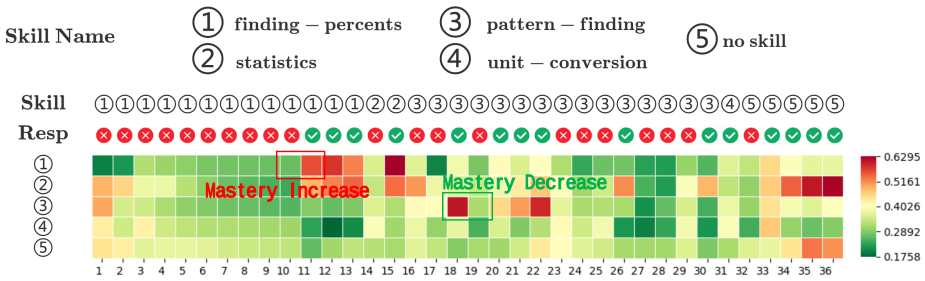


Fig. 7: A student's knowledge state evolution over 36 time steps.

5 Conclusion

We propose a novel method for modeling student profiles and long-term knowledge states, called DGMKT which contains two main Modules (DGSPM, MSMM) and an Integration Module. The DGSPM models student profiles from two perspectives: the exercises the student has interacted with and the sequence in which these exercises were interacted with, through the student association hypergraph and the exercise transition directed graph. Experimental results show that this student profiling method not only enhances the performance of knowledge tracing models but also demonstrates strong adaptability, making it suitable for various knowledge tracing model architectures. More importantly, we introduce MSMM, the first application of the Mamba structure to knowledge tracing tasks, which avoids the forgetting issue in RNN-based models and the need for manually setting bias functions in Self-Attention-based models, enabling long-term modeling of student knowledge states. The source code and datasets are publicly available at https://github.com/collegestu1231/DGMKT/tree/master.

Acknowledgments. This research was supported by the National Natural Science Foundation of China under Grant (Nos. 62137001, 62272093, 62372097).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Cui, C., Yao, Y., Zhang, C., Ma, H., Ma, Y., Ren, Z., Zhang, C., Ko, J.: Dgekt: a dual graph ensemble learning method for knowledge tracing. ACM Transactions on Information Systems **42**(3), 1–24 (2024)
2. Ghosh, A., Heffernan, N., Lan, A.S.: Context-aware attentive knowledge tracing. In: SIGKDD. pp. 2330–2339 (2020)
3. Gu, A., Dao, T.: Mamba: Linear-time sequence modeling with selective state spaces (2023)
4. Guo, X., Huang, Z., Gao, J., Shang, M., Shu, M., Sun, J.: Enhancing knowledge tracing via adversarial training. In: Proceedings of the 29th ACM International Conference on Multimedia. pp. 367–375 (2021)
5. He, X., Cao, K., Zhang, J., Yan, K., Wang, Y., Li, R., Xie, C., Hong, D., Zhou, M.: Pan-mamba: Effective pan-sharpening with state space model. Information Fusion p. 102779 (2024)
6. Im, Y., Choi, E., Kook, H., Lee, J.: Forgetting-aware linear bias for attentive knowledge tracing. In: CIKM. pp. 3958–3962 (2023)
7. Khajah, M.M., Huang, Y., González-Brenes, J.P., Mozer, M.C., Brusilovsky, P.: Integrating knowledge tracing and item response theory: A tale of two frameworks. In: CEUR Workshop proceedings. vol. 1181, pp. 7–15. University of Pittsburgh (2014)
8. Khasanah, A.U., et al.: A comparative study to predict student's performance using educational data mining techniques. In: IOP Conference Series: Materials Science and Engineering. vol. 215, p. 012036. IOP Publishing (2017)
9. Lee, J., Yeung, D.Y.: Knowledge query network for knowledge tracing: How knowledge interacts with skills. In: Proceedings of the 9th international conference on learning analytics & knowledge. pp. 491–500 (2019)
10. Li, X., Bai, Y., Guo, T., Liu, Z., Huang, Y., Zhao, X., Xia, F., Luo, W., Weng, J.: Enhancing length generalization for attention based knowledge tracing models with linear biases. In: IJCAI. pp. 5918–5926 (2024)
11. Li, X., Bai, Y., Guo, T., Zheng, Y., Hou, M., Zhan, B., Huang, Y., Liu, Z., Gao, B., Luo, W.: Extending context window of attention based knowledge tracing models via length extrapolation. In: ECAI, pp. 1479–1486. IOS Press (2024)
12. Liang, K., Zhang, Y., He, Y., Zhou, Y., Tan, W., Li, X.: Online behavior analysis-based student profile for intelligent e-learning. Journal of Electrical and Computer Engineering **2017**(1), 9720396 (2017)
13. Long, T., Qin, J., Shen, J., Zhang, W., Xia, W., Tang, R., He, X., Yu, Y.: Improving knowledge tracing with collaborative information. In: WSDM. pp. 599–607 (2022)
14. Ma, J., Li, F., Wang, B.: U-mamba: Enhancing long-range dependency for biomedical image segmentation (2024)
15. Minn, S., Yu, Y., Desmarais, M.C., Zhu, F., Vie, J.J.: Deep knowledge tracing and dynamic student classification for knowledge tracing. In: 2018 IEEE International conference on data mining (ICDM). pp. 1182–1187. IEEE (2018)
16. Nagatani, K., Zhang, Q., Sato, M., Chen, Y.Y., Chen, F., Ohkuma, T.: Augmenting knowledge tracing by considering forgetting behavior. In: WWW. pp. 3101–3107 (2019)
17. Nakagawa, H., Iwasawa, Y., Matsuo, Y.: Graph-based knowledge tracing: modeling student proficiency using graph neural network. In: IEEE/WIC/ACM International Conference on Web Intelligence. pp. 156–163 (2019)
18. Pandey, S., Karypis, G.: A self-attentive model for knowledge tracing (2019)

19. Pandey, S., Srivastava, J.: Rkt: relation-aware self-attention for knowledge tracing. In: CIKM. pp. 1205–1214 (2020)
20. Pardos, Z.A., Heffernan, N.T.: Modeling individualization in a bayesian networks implementation of knowledge tracing. In: User Modeling, Adaptation, and Personalization: 18th International Conference, UMAP 2010, Big Island, HI, USA, June 20-24, 2010. Proceedings 18. pp. 255–266. Springer (2010)
21. Pardos, Z.A., Heffernan, N.T.: Kt-idem: Introducing item difficulty to the knowledge tracing model. In: User Modeling, Adaption and Personalization: 19th International Conference, UMAP 2011, Girona, Spain, July 11-15, 2011. Proceedings 19. pp. 243–254. Springer (2011)
22. Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L.J., Sohl-Dickstein, J.: Deep knowledge tracing. Advances in neural information processing systems **28** (2015)
23. Shen, G., Yang, S., Huang, Z., Yu, Y., Li, X.: The prediction of programming performance using student profiles. Education and Information Technologies **28**(1), 725–740 (2023)
24. Wang, F., Liu, Q., Chen, E., Huang, Z., Chen, Y., Yin, Y., Huang, Z., Wang, S.: Neural cognitive diagnosis for intelligent education systems. In: AAAI. vol. 34, pp. 6153–6161 (2020)
25. Wilson, K.H., Karklin, Y., Han, B., Ekanadham, C.: Back to the basics: Bayesian extensions of irt outperform neural networks for proficiency estimation (2016)
26. Yeung, C.K., Yeung, D.Y.: Addressing two problems in deep knowledge tracing via prediction-consistent regularization. In: Proceedings of the fifth annual ACM conference on learning at scale. pp. 1–10 (2018)
27. Yin, Y., Dai, L., Huang, Z., Shen, S., Wang, F., Liu, Q., Chen, E., Li, X.: Tracing knowledge instead of patterns: Stable knowledge tracing with diagnostic transformer. In: WWW. pp. 855–864 (2023)
28. Yu, W., Wang, X.: Mambaout: Do we really need mamba for vision? (2024)
29. Yudelson, M.V., Koedinger, K.R., Gordon, G.J.: Individualized bayesian knowledge tracing models. In: Artificial Intelligence in Education: 16th International Conference, AIED 2013, Memphis, TN, USA, July 9-13, 2013. Proceedings 16. pp. 171–180. Springer (2013)
30. Zhang, J., Shi, X., King, I., Yeung, D.Y.: Dynamic key-value memory networks for knowledge tracing. In: WWW. pp. 765–774 (2017)
31. Zhu, L., Liao, B., Zhang, Q., Wang, X., Liu, W., Wang, X.: Vision mamba: Efficient visual representation learning with bidirectional state space model (2024)