Expert Systems With Applications 206 (2022) 117681

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

# SGKT: Session graph-based knowledge tracing for student performance prediction

Zhengyang Wu [a], Li Huang [a], Qionghao Huang [b], Changqin Huang [b,*], Yong Tang [c,*]

[a] School of Computer Science, South China Normal University, Guangzhou, Guangdong, 510631, China
[b] Key Laboratory of Intelligent Education Technology and Application of Zhejiang Province, Zhejiang Normal University, Jinhua, Zhejiang, 321004, China
[c] Guangzhou Key Laboratory of Big Data and Intelligent Education, Guangzhou, Guangdong, 510631, China

ARTICLE INFO

*Keywords:*
Knowledge tracing
Graph Convolutional Network
Gated Graph Neural Networks
Attention Mechanism

ABSTRACT

Knowledge tracing is a modeling method of students' knowledge mastery. The deep knowledge tracing (DKT) model uses long short-term memory (LSTM) to process the sequence data of students exercises. However, the LSTM-based model pays more attention to the short-term response status of students while ignoring the long-term learning process. Moreover, existing graph-based knowledge tracing models focus on the static relationship between exercises and skills, ignoring the dynamic graphs formed by students exercises in a session. In this work, we propose a novel knowledge tracing model which is based on an exercise session graph, named session graph based knowledge tracing (SGKT). The session graph is used to model the students' answering process. In addition, a relationship graph is used to model the relationship between exercises and skills. Then we use gated graph neural networks to obtain the students' knowledge state from the session graph and use graph convolutional networks to obtain the embedding representations of exercises and skills in the relationship graph. Next, through the interaction mechanism, multiple interaction states composed of knowledge states and embedding representations are obtained. The attention mechanism is used to find the focus from these states and make predictions. Experiments are conducted on three publicly available datasets and the results show that our approach has advantages over some existing baseline methods.

## 1. Introduction

Learning management systems (LMS) have been widely used in various educational stages from primary to tertiary education for student administration, documentation, tracking, reporting, and the delivery of educational courses, training programs, or learning and development programs (McGill and Klobas, 2009). The development of educational data mining technology has played an important role in improving LMS (Miah et al., 2020). Monitoring students' knowledge mastery based on their performance in completing exercises is an important task (Tomasevic et al., 2020) and provides a foundation for more effective learning assessment and learning recommendations (Sweeney et al., 2016). This problem, known as *knowledge tracing* (Liu, Shen et al., 2021), aims to analyze the exercises that the students have worked on and infers the students' level of knowledge mastery based on whether their answers to the exercises are correct, and then predicts the probability that students can correctly answer the exercises which will follow.

*Knowledge tracing* was first introduced in 1994. It aims to estimate students' learning performance in a learning guidance system (Corbett and Anderson, 1994). After adopting the Bayesian network, it is now called *Bayesian Knowledge Tracing* (BKT). After that, BKT has been improved in various ways. In 2015, a recurrent neural network (RNN)-based model for student performance prediction was proposed, named *Deep Knowledge Tracing* (DKT) (Piech et al., 2015). DKT adopts LSTM to perform prediction tasks since the historical records of students' answering exercises are in the form of sequences. LSTM is a variant of RNN, which is considered a high-performance neural network for sequence processing tasks (Graves et al., 2013) and has been widely used in various sequence-based models, such as the methods in the field of educational data mining proposed by Xiong et al. (2016), Wang et al. (2017), Mongkhonvanit et al. (2019), and Liu, Huang et al. (2021).

While knowledge tracing has been significantly improved using various RNN-based models, the existing approaches analyze exercises in the sequential order in which the students worked. This implies that these RNN-based models tend to weigh the students' performance on recent exercises and ignore their performance on earlier ones. This can be true in many application scenarios but critical information can be

* Corresponding authors.
*E-mail addresses:* wuzhengyang@m.scnu.edu.cn (Z. Wu), lhuang@m.scnu.edu.cn (L. Huang), qhhuang@zjnu.edu.cn (Q. Huang), cqhuang@zjnu.edu.cn (C. Huang), yongt@m.scnu.edu.cn (Y. Tang).

https://doi.org/10.1016/j.eswa.2022.117681
Received 26 January 2022; Received in revised form 5 April 2022; Accepted 28 May 2022
Available online 15 June 2022

missed in the practice of education. For example, students may give an incorrect answer to an exercise on their first attempt, however, after working on the same exercise repeatedly, they may answer it correctly, which is a sign of better knowledge mastery than if they had answered the exercise correctly the first time. Second, knowledge mastery is relatively stable over a long period, hence it is reasonable to assume that when students work on an exercise for a second time, they may remember their experience of working on it the first time. Therefore, graph-based knowledge tracing methods are proposed to solve these problems. Graph-based knowledge tracing (GKT) proposed in Nakagawa et al. (2019) uses several ways to cast the knowledge structure as a graph, where nodes correspond to skills and edges correspond to their relationships. These graphs are used as input to the model to predict the students' responses. However, it does not analyze which kind of graph is more reasonable. In addition, some existing graph-based knowledge tracing models focus more on the relational information between exercises and skills (Yang et al., 2020). However, the graph structure information constituted by the students' answering behaviors is ignored.

In RNN-based knowledge tracing models, session sequence is the basic data for students' knowledge state modeling. To improve the representation of session sequences, it is reasonable to structure exercise session sequences into a graph structure. This is because compared to sequences, graph structures can capture richer local dependencies through graph neural networks. Furthermore, a graph neural network is capable of providing rich local contextual information by encoding edge or node attribute features (Battaglia et al., 2018). Based on the session graph, the graph-based model can capture transitions of neighbor items and generate the latent vectors for all nodes involved in the graph.

In this work, we propose a novel session graph-based knowledge tracing model named SGKT. SGKT uses gated graph neural networks (GGNN) to model the students' knowledge state and uses graph convolutional network (GCN) and self-attention with a forgetting mechanism to model exercises and skills. Then, an interactive mechanism combines the student's knowledge state with the representation of exercises and skills to obtain the final prediction result. In particular, SGKT employs a heterogeneous relation graph to represent the relationship between exercises and skills and it organizes the exercises on which each student has worked as a directed session graph. SGKT employs gated recurrent units (GRUs) to generate a hidden state of students based on a session graph because a GRU not only captures the features in the current time-step, it also integrates the features from the neighbor time-step. We conducted extensive experiments on three real datasets. The results verify the superiority of SGKT in various aspects. The contributions of this study are threefold:

- This study proposes a novel graph-based knowledge tracing model that uses a session graph to model students' knowledge states.
- To better simulate a scenario in which students answer exercises and for model knowledge tracing, this study applies heterogeneous relation graphs and forgetting mechanisms and develops a more powerful graph-based solution.
- This study verifies the robustness of the knowledge tracing solution based on the session graph structure model compared with the current mainstream solutions through various experiments.

The rest of the paper is organized as follows. We review the related work in Section 2. In Section 3, we introduce some important notations and preliminaries. In Section 4, we describe the framework and implementation of SGKT. Extensive experimental studies are detailed in Section 5. Finally, we present the conclusion in Section 6.

## 2. Related work

In this section, we first review the related work on knowledge tracing. Then, the related research on *graph-based knowledge tracing* is summarized.

### 2.1. Knowledge tracing

The term knowledge tracing was first proposed by Corbett and Anderson in 1995. A knowledge tracing model comprises three parts: (1) student knowledge state modeling; (2) exercises and skills (or knowledge concepts) modeling; (3) prediction module. In the earliest knowledge tracing model, the probability estimation of students' learning performance employs Bayesian networks, called Bayesian knowledge tracing (BKT). BKT adopts a two-state student modeling framework, that is, skills are learned or not learned by students. Baker et al. (2008) contextually estimate whether each student's response is a guess or a slip to improve BKT. Lin and Chi (2016) proposed the intervention-BKT by adding instructional intervention information in student modeling. Zhang and Yao (2018) divided the two states defined by the original BKT into three states based on tripartite decision-making, which enhanced the semantics of student modeling. Deep knowledge tracing (Piech et al., 2015) is the first approach to introduce deep neural networks into knowledge tracing. DKT adopts LSTM to model the students' knowledge states, and since there is no exercise model, it can predict the probability of a student correctly answering each skill in the next time step instead of exercise. To further verify the generalization effect of DKT, Xiong et al. (2016) extended it to other datasets. Wang et al. (2017) applied DKT to the performance tracing of students on programming task and used RNN to process the input sequence of programming interactions over time. Minn et al. (2018) clustered the students based on the sequence of their responses in the exercises to improve the relevance of student modeling in DKT. Mongkhonvanit et al. (2019) used RNN to model students' knowledge states while watching learning videos and predict the students' responses to the next item in the video. Previous DKT-based research focused on modeling students' knowledge states while ignoring the modeling of exercises and skills (or knowledge concepts) using one-hot encoding. In the work on EKT, Liu, Huang et al. (2021) proposed the EERNN model which adopts an exercise embedding module to generate an exercise model, which is used to predict student performance on future exercises. To trace the students' knowledge acquisition on multiple explicit concepts, knowledge embedding is incorporated in the modeling process to extend EERNN. Yang et al. (2020) proposed GIKT which uses a bipartite graph to model the relationships between exercises and skills and employs LSTM to model students' knowledge states. Bi-CLKT (Song et al., 2022) employs contrastive learning and applies graph-level and node-level GCNs to the model relationship of exercise-to-exercise (e2e) and concept-to-concept (c2c), and then uses deep neural networks to build prediction layers. Song et al. (2021) proposed a joint graph convolutional network based deep knowledge tracing (JKT) framework to model exercises and skills by learning the multi-dimensional relationships of e2e and c2c, and represent exercises and skills as dense vectors, where the prediction module of JKT follows the structure of DKT.

In practice, students may demonstrate their mastery of specific skills many times, because these skills may be required to complete various exercises (Liu, Zou et al., 2021). At this point, if we take exercises and skills as nodes, the data on how students answer exercises can form a graph structure. However, most existing knowledge tracing models are still based on sequence-structure data.

### 2.2. Graph neural networks

A graph is a non-linear structure which is more complex than a tree structure. Graph structure data can intuitively express the entities and relationships in the real world (Battaglia et al., 2018). GNN is a type of neural network which directly operates on graph structure data. The concept of GNN was first proposed by Scarselli et al. (2009), who extended existing neural networks to process the data represented in graph domains. Each node is naturally defined by its features and the related nodes in a graph. The aim of GNN is to learn a state embedding

$h_v \in R_s$ which contains the information of the neighborhood for each node. The state embedding $h_v$ is a $s$-dimension vector of node $v$ and can produce an output $o_v$ such as the node label. In application, the GNN retains a state that can represent information from its neighborhood with arbitrary depth. A typical application of GNN is node classification. Essentially, every node in the graph is associated with a label, and we want to predict the nodes' labels without ground truths. In the node classification task setup, each node $v$ is characterized by its feature $x_v$ and associated with a ground-truth label $t_v$. Given a partially labeled graph $G$, the goal is to leverage these labeled nodes to predict the labels of the unlabeled. It learns to represent each node with a $d$-dimensional vector (hidden state) $h_v$, containing its neighborhood information.

GCN (Kipf and Welling, 2017) updates node representations based on itself and its neighbors and is often proposed for semi-supervised graph classification. Using multiple graph convolutional layers, GCN can ensure the updated nodes represent the attributes of neighbor nodes and the information of higher-order neighbors.

To reduce the limitations in the GNN model and improve the long-term propagation of information across the graph structure, GGNN (Li et al., 2016) uses a gate mechanism in the propagation step and unrolls the recurrence for a fixed number of steps $T$ and uses back-propagation through time to compute the gradient.

### 2.3. Graph-based knowledge tracing

Recently, research on knowledge tracing has focused on exploring the use of graph structures to capture more information.

The model proposed by Chanaa and Faddouli (2020) first constructs a dynamic graph that changes over time, where each node represents a student, the nodes are fixed, but the edges are dynamic. Both nodes and graph topology are transforming over time, matching the knowledge tracing of students. The model learns feature representation by aggregating each student (i.e., node) and its neighbors, then extracting the network topology information at each different time step. Finally, the representation of each node is sent to the classifier to determine the student's knowledge state. Nakagawa et al. (2019) proposed GKT. This model sets the course exercises as a graph structure $G = (V, E)$. The $n$ skills are the nodes $V = \{v_1, \ldots, v_n\}$ of the graph, and the shared dependency of these skills is the edge $E \subseteq V \times V$ of the graph. Assume that the student independently has the knowledge state $h_t = \{h_{v_i}^t\}$ at that time for each skill at time $t$, and the knowledge state is updated with time as follows: when answering the exercises related to the skill $v_i$, the student's knowledge state is not only the skill of the answer itself, it also contains information on its neighbor skills; then, the updated knowledge state is used to predict the probability of the student correctly answering each skill at $t$. GIKT proposed by Yang et al. (2020) considers the exercise–skill relationship graph a bipartite graph and then uses the GCN to fully integrate the exercise–skill correlation through embedding propagation. PEBG-KT proposed by Liu et al. (2020) employs a pre-training approach to learn a low-dimensional embedding via the exercise–skill Bipartite Graph for each exercise with useful side information. However, the model proposed by Chanaa and Faddouli (2020) and GKT ignore the relationship between exercises and skills. GIKT and PEBG-KT use bipartite graphs which focus on the relationship between exercises and skills but ignore the relationship between exercises. Furthermore, they all ignore the graph structure formed during the students' exercise answering sessions.

## 3. Preliminaries

In this section, we first formally define the task of student performance prediction in an intelligent learning system. Then, we describe how to represent a heterogeneous relation graph about the students, exercises, and skills. Next, we explain the construction of the session graph. Table 1 shows some important notations. The following section gives a more detailed explanation of their roles.

**Table 1**
A summary of notations.

| Notation | Description |
|---|---|
| HRG | The heterogeneous relation graph of students, exercises and skills |
| SG | The exercise session graph |
| $e_i$ | The $i$th exercise |
| $k_j$ | The $j$th skill |
| $a_i$ | The answer of the $i$th exercise |
| $h_i$ | The hidden state of $i$th student |
| $p_t$ | The probability of correctly answering the next candidate exercise at time-step $t$ |

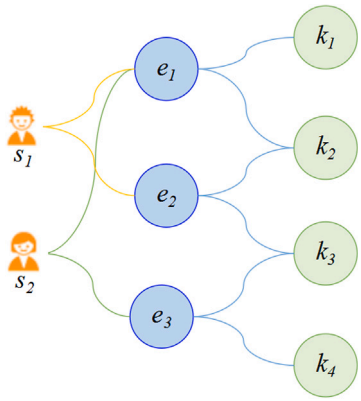



**Fig. 1.** Heterogeneous relation graph of students answering exercises.

**Definition 1** (*Student Performance Prediction Problem*)**.** Given the exercise answering records of each student from time-step 1 to $t-1$, our goal is to predict the probability $p_t$ of correctly answering the next candidate exercise at time-step $t$.

**Definition 2** (*Heterogeneous Relation Graph*)**.** In the process of answering exercises, there are heterogeneous relationships among students, exercises, and skills. A heterogeneous relation graph (HRG) is denoted as $HRG = \{\mathbb{V}; \mathbb{E}\}$ consisting of an object set $\mathbb{V}$ and a link set $\mathbb{E}$. $\mathbb{V}$ is a set of object types (student, exercise or skill). $\mathbb{E}$ is a set of relation types, and $\mathbb{E} = \{r_A, r_C\}$, where $r_A$ represents the relation *answered*, and $r_C$ represents the relation *contains*.

Take Fig. 1 as an example. The student $s_1$ answered the exercises $e_1$ and $e_2$, and the student $s_2$ answered the exercises $e_2$ and $e_3$. $e_1$ requires skills $k_1$, $k_2$. $e_2$ requires skills $k_2$, $k_3$, and $e_3$ requires skill $k_4$. In this HRG, $V = \{s_1, s_2, e_1, e_2, k_1, k_2, k_3, k_4\}$, From the figure, we can observe that there is a relationship between $e_1$ and $e_2$ through knowledge $k_2$, and there is also a relationship between $e_2$ and $e_3$ through student $s_2$. This is because the student answered these two exercises in a session. Therefore, for the exercise and skill, the meta relationship path of this HRG includes exercise–skill–exercise (eKe), exercise–student–exercise (eSe), and skill–exercise–skill (kEk).

**Definition 3** (*Session Graph*)**.** In an exercise answering session, each answer $a_i$ is related to an exercise $e_i$, denoted by the tuple $e_i^a = (e_i, a_i)$. Students answer the exercises individually, so there is a partial ordering relationship among all $e^a$ in the session. Consider a pair of linked exercise and answer as a node, a session graph is denoted as $SG = \{e^A, P\}$, where $e^A = \{e_i^a | 1 \le i \le n\}$, $n$ is the length of the session sequence, and $P$ represents the partial order.

Take Fig. 2 as an example. The upper part shows the sequence in which the exercises were answered $e_1^{\surd} \to e_2^{\times} \to e_3^{\times} \to e_4^{\surd} \to e_2^{\surd}$, where skill $e_2$ appears twice. This sequence can be transformed into a directed graph, shown in the lower part. Nodes $e_1$, $e_3$, and $e_4$ are all neighbor nodes of node $e_2$.

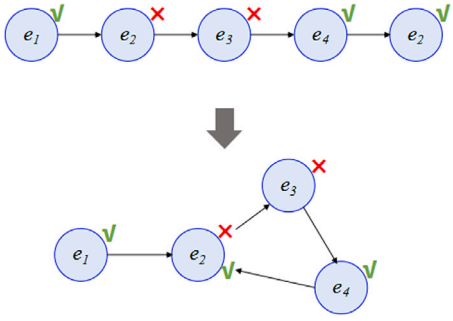



**Fig. 2.** A session-graph example.

**Definition 4** (*Forgetting Mechanism*)**.** In the process of learning, forgetting is inevitable. Due to forgetting, the exercises that students have answered correctly may be answered incorrectly after a period. Furthermore, the forgetting rate is positively correlated with the length of time. We formalize the forgetting mechanism (Chen et al., 2021) as:

$$m(\Delta\tau) = \theta_1 \times exp^{-\theta_2 \Delta\tau} + \theta_3, \tag{1}$$

where $\theta_1$, $\theta_2$, and $\theta_3$ are the fitting parameters and $m(\Delta\tau)$ represents the residual memory ratio. A higher $m(\Delta\tau)$ means that the student has more memory about the corresponding content. $\Delta\tau$ is the time step between the current moment and the initial moment.

## 4. Session graph-based knowledge tracing model

To address the problem defined in Section 3, we propose SGKT, a session graph-based knowledge tracing framework. As shown in Fig. 3, SGKT comprises three parts: the HRG embedding module (for exercises and skills modeling), the SG embedding module (for student knowledge state modeling), and the prediction module. The HRG embedding module is used to learn the embedding of exercise-to-exercise and skill-to-skill relational information from HRG. It generates the representation vector, and then employs self-attention mechanisms with FM to add dynamic semantics to them. The SG embedding module is used to generate the students' knowledge state. The prediction module predicts the probability of students answering the next exercise correctly. The SG embedding and prediction modules are jointly optimized during training, where the exercises and skills embeddings are dynamically obtained from the HRG embedding module.

### 4.1. HRG embedding module

This module uses the GCN model to generate the embedding representation of the nodes in the HRG composed of students, exercises, and skills. Before embedding HRG, we first define neighbors according to the meta relationship path in Definition 2, where three meta-relational paths are defined, namely eSe, eKe, and kEk. Specifically, we process the dataset according to the following rules: the neighbors of an exercise are the exercises answered by the same student, i.e., eSe, or the exercises that require the same skill, i.e., eKe. A skill's neighbor is that skill covered by the same exercise, i.e., kEk. Therefore, during the propagation of the GCN, we use two matrices, one is the exercise–exercise relation matrix, and the other is the exercise–skill relation matrix. As shown in the upper part of Fig. 3, although HRG is a tripartite graph, the embedding layer of this module only focuses on the state of exercises and skills, and students are regarded as a connection between exercises. In the matrices of exercises and skills propagated in GCN, the neighbors of an exercise are the other exercises answered by the same student and related skills. The neighbors of skills are all the exercises related to it. To extract the high-order information, we use GCN to generate the embedding of exercises and skills i.e., $\tilde{e}$ and $\tilde{k}$.

GCN has a lot of graph convolution layers to encode high-order neighbor information, and each layers' node can be updated by states of itself and neighbor nodes. We set the $i$th node in the graph as $\beta_i$, indicating skill state $k_i$ or exercise state $e_i$. We also set $\beta_i$'s neighbor nodes as $\mathbf{N}_i$. So the exercise of an $l$th layer of GCN can be expressed as:

$$\beta_i^l = ReLU\left(\frac{1}{\mathbf{N}_i}\sum_{j\in\{i\}\cup\mathbf{N}_i}\mathbf{y}^l\beta_j^{l-1} + \mathbf{z}^l\right), \tag{2}$$

where $\mathbf{y}^l$ and $\mathbf{z}^l$ are the aggregate weight and bias, which can be learned in the $l$th GCN layer, and the activation function *ReLU* is used in it. After embedding the propagation by GCN, we can obtain the $\tilde{e}$ and $\tilde{k}$.

### 4.2. SG embedding module

This embedding module uses the GGNN model to generate students' knowledge hidden states based on the session graph.

#### 4.2.1. Knowledge hidden state

During the exercises and corresponding answers propagated in GGNN, they are combined in the following way:

$$c_i = e_i \oplus a_i, \tag{3}$$

where $e_i$ denotes the $i$th exercise in the session and $\oplus$ indicates an element-wise addition. The process of information propagation in GGNN is shown in Fig. 4. $\alpha_i$ is a new state, being the input of the GRU which contains information about the neighbor nodes of the current node $c_i$, which indicates the effect of the previous exercise on the next exercise in the exercise session. $\alpha_i$ can be formalized as:

$$\begin{aligned}\alpha_i = \mathbf{Concat}\,&\left(\mathbf{M_i^I}\left(\left[c_1, c_2, \ldots, c_n\right]\mathbf{W^I} + \mathbf{b^I}\right),\right.\\ &\left.\mathbf{M_i^o}\left(\left[c_1, c_2, \ldots, c_n\right]\mathbf{W^o} + \mathbf{b^o}\right)\right),\end{aligned} \tag{4}$$

where $\mathbf{W}^I, \mathbf{W}^O \in R^{d\times d}$ are the parameter matrices, and $b^I, b^O \in R^d$ are the bias vectors. $\left[c_1, c_2, \ldots, c_n\right]$ is the list of exercises and answers that have been processed, and $\mathbf{M_i^I}, \mathbf{M_i^O} \in \mathbb{R}^{1\times\mathbf{n}}$ are the $i$th row of each matrix corresponding to the $c_i$. An example of information propagation between layer $t-1$ and layer $t$ of GGNN is shown in Fig. 4, where the solid line represents the post-order relationship, and the dashed line represents the pre-order relationship. $[\alpha_1, \alpha_2, \ldots, \alpha_n]$ is the input of GRU, and $[h_1, h_2, \ldots, h_n]$ is the output which is the knowledge hidden state of the student. This hidden state can be updated not only by the states of the current exercise and answer but also by its neighbor nodes' states. The update process of $h_i$ can be formalized as follows:

$$c_i^t = \alpha_i^{t-1} \oplus c_i^{t-1}, \tag{5}$$

$$\omega_i^t = \sigma\left(U_\omega c_i^t + V_\omega h_i^{t-1}\right), \tag{6}$$

$$r_i^t = \sigma\left(U_r c_i^t + V_r h_i^{t-1}\right), \tag{7}$$

$$\tilde{h}_i^t = tanh\left(U_h c_i^t + V_h\left(r_i^t \otimes h_i^{t-1}\right)\right), \tag{8}$$

$$h_i^t = \left(1-\omega_i^t\right)\otimes h_i^{t-1} + \omega_i^t \otimes \tilde{h}_i^t, \tag{9}$$

where $U_\omega$, $U_r$, $U_h \in R^{2d\times d}$, $V_\omega, V_r$, $V_h \in R^{d\times d}$, are the parameters that can be learned, $\sigma$ is the activation function *Sigmoid*, $\oplus$ indicates element-wise addition, and $\otimes$ indicates dot product. $\omega_i^t$ is the update gate that can update the forgetting information. $r_i^t$ is the *reset gate* that can filter information. In Eq. (8), $r_i^t \otimes h_i^{t-1}$ indicates that new information is generated from some past information. In Eq. (9), $\left(1-\omega_i^t\right)\otimes h_i^{t-1}$ indicates some past information has been forgotten and $\omega_i^t \otimes \tilde{h}_i^t$ indicates some new information has been remembered.

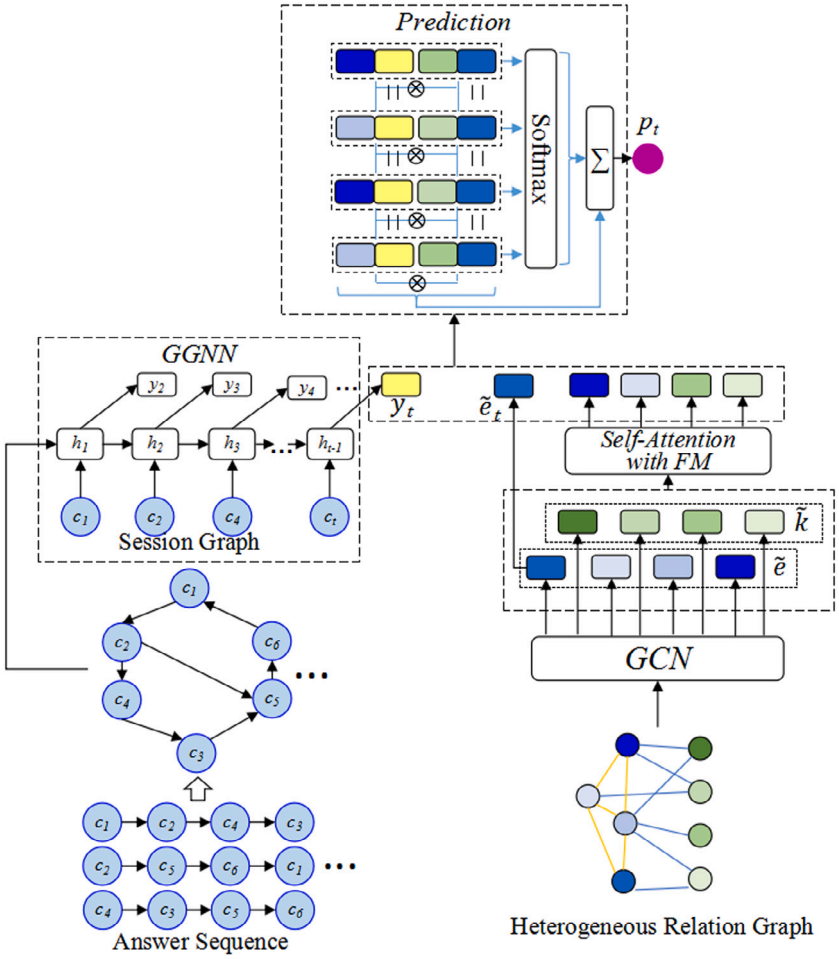



**Fig. 3.** Framework of the proposed approach SGKT.

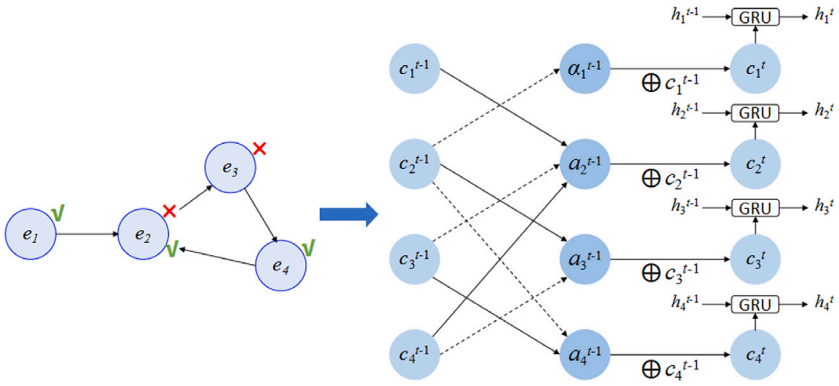



**Fig. 4.** An example of information propagation between layer $t-1$ and layer $t$ of GGNN where the solid line represents the post-order relationship and the dashed line represents the pre-order relationship.

### 4.3. Self-attention with FM module

Self-attention with a forgetting mechanism is used to model the exercises and skills. Based on the HRG, we believe that GCN can generate the embedding representation of the static relationship between exercises and skills. However, this embedding representation does not contain the semantics information of students' exercise behavior. Therefore, in this module, a self-attention mechanism is used to learn the semantics of students' exercises and answering behaviors to enhance the modeling of exercises and skills.

We denote the globe exercises answering states by $E = \left\{ e_i \middle| i \in [1, 2, \ldots, t-1] \right\}$. Then, the self-attention mechanism is used to obtain the student's exercise answering state $F$. This process is formalized as:

$$F = Softmax(\frac{(EW^Q)(EW^K)^T}{\sqrt{d^k}})(EW^V), \tag{10}$$

where $W^Q$, $W^K$, $W^V \in R^{d \times d}$ are the projection matrices, $E \in R^{2d \times d}$, and $\sqrt{d^k}$ is the scaling factor, which reduces the influence of a significant increase in the magnitude of the dot product.

To enhance the personalization of answering behavior information to students' answering state, we add a layer of the forgetting mechanism. This process is formalized as:

$$\hat{E} = \{f_i(\theta_1 \times exp^{-\theta_2 \Delta\tau_i} + \theta_3) | i \in \{1, 2, \ldots, t-1\}\}, \tag{11}$$

where $f_i \in F$, $\Delta\tau_i$ indicates the time-step interval between the $i$th moment and the initial moment. $f_i$ is the vector generated by Eq. (10) based on the elements in $E$ at each time step. At the $i$th time step, after adjusting the forgetting weight of $f_i$ through Eq. (11), input Eq. (10) again to get $f_{i+1}$. Similarly, after adjusting the forgetting weight using Eq. (11), $f_{i+1}$ is also input again into Eq. (10) to get $f_{i+2}$. Iterate $t-1$ times according to this, until all $f_i$ has completed the adjustment of the forgetting weight and they constitute $\hat{E}$.

### 4.4. Prediction module

This module predicts the probability of answering the next exercise correctly. In this module, the skills embedding $\tilde{k}_j \in K_t$, the answered exercises embedding $\tilde{e}_i \in \hat{E}_t$, and the embedding of target exercise $\tilde{e}_t$ are combined as the *exercise knowledge state*. The $h_t$ of the GGNN generated is the *student knowledge state*. These states are interacted by element-wise addition. The details are as follows:

Tuple $\langle \tilde{e}_t, h_t \rangle$ indicates the mastery degree of the target exercise. $\langle \tilde{k}_j, h_t \rangle$ indicates the mastery degree of the student on the $j$th skill, which is related to the target exercise. $\langle \tilde{e}_t, \hat{E}_t \rangle$ and $\langle \tilde{k}_j, \hat{E}_t \rangle$ represent the influence of the student's exercise answering experience on the target exercise and the associated skills in the target exercise.

Then, we use an attention mechanism to obtain the weights of all interaction terms and compute the weighted sum to predict:

$$\gamma_{i,j} = Softmax_{i,j}\left(\mathbf{W}^T \left[\mathbf{f}_i, \mathbf{f}_j\right] + b\right), \tag{12}$$

$$p_t = \sum_{\mathbf{f}_i \in \hat{E} \cup \{h_t\}} \sum_{\mathbf{f}_j \in \tilde{K}_{e_t} \cup \{\tilde{e}_t\}} \gamma_{i,j} \mathbf{g}\left(\mathbf{f}_i, \mathbf{f}_j\right), \tag{13}$$

where $\mathbf{W} \in R^{d \times d}$ is the parameter matrix, and $b \in R^d$ is the bias vector. $p_t$ represents the predicted probability of students answering the exercise $e_t$ correctly. In the Eq. (13), $\tilde{K}_{e_t}$ is the aggregated neighbor skill embedding of the exercise $e_t$, and function $\mathbf{g}$ represents the dot product.

Finally, *Gradient Descent* is used to minimize the cross-entropy loss between the predicted probability of answering correctly, which can be formalized as:

$$L = -\sum_t \left(\gamma_t log p_t + \left(1 - \gamma_t\right) log\left(1 - p_t\right)\right), \tag{14}$$

where $\gamma_t$ represents the truth label.

## 5. Experiments

In this section, we describe the extensive experiments that were conducted to verify the effectiveness and advantage of our approach. We compare the performance of SGKT [1] with several baselines on three real datasets. To compare the performance between the sequence-based and graph-based KT model, we conduct the knowledge state evolution experiment and the cold start experiment using our approaches, DKT and DKT+. Finally, we conducted two ablation studies to observe the core module and the key super-parameter of our approach.

[1] https://github.com/lhuang65/SGKT

**Table 2**
Datasets statistics.

| Items | ASSIST09 | ASSIST12 | EdNet |
|---|---|---|---|
| # students | 3852 | 27,485 | 5000 |
| # exercises | 17,737 | 53,065 | 12,161 |
| # skills | 123 | 265 | 189 |
| # exercises per skill | 173 | 200 | 147 |
| # skills per exercise | 1.197 | 1.000 | 2280 |
| # attempts per exercise | 16 | 51 | 56 |
| # attempts per skill | 2743 | 10,224 | 8420 |

### 5.1. Overview of datasets

Our experiments are performed on three classic knowledge tracing application datasets. The detailed statistics of these datasets are shown in Table 2.

**ASSIST09** was collected during the school year 2009–2010 from ASSISTments online education platform. We conducted our experiments on the *skill builder* dataset. Similar to Xiong et al. (2016), we removed the redundant data from the original dataset. As a result, the dataset has 3852 students, 123 skills, and 17,737 exercises.

**ASSIST12** was collected from the same platform as ASSIST09 during the school year 2012–2013. Each exercise is related to one skill in this dataset, but one skill is still related to more than one exercise. We undertake the same data processing as for ATTIST09. The dataset has 27,485 students, 2,709,436 exercises, and 265 skills.

**EdNet** was collected by Choi et al. (2020) from which we randomly chose 5000 students, 189 skills, and 12,161 exercises.

We used 80 percent of the data as training sets and 20 percent as test sets for each dataset. We used AUC as the evaluation metric to evaluate the performance of these datasets.

### 5.2. Baselines

To evaluate the performance of SGKT, we need other baselines for comparison purposes. We introduce six baselines that are used in our comparison experiments. The details of the baselines are as follows:

BKT and DKT are two classic models of knowledge tracing. Therefore, we use them as baselines. DKT+ is a modified model to address the shortcomings of DKT, and we also use it as one of our baselines. KTM is a special knowledge tracing model. The other two mainstream models, DKVMN and GIKT, have good performance and adopt novel methods, so we also use them as the baselines.

**BKT** is proposed in Corbett and Anderson (1994). BKT uses a Bayesian network for prediction to model the knowledge state of the skill as a binary variable.

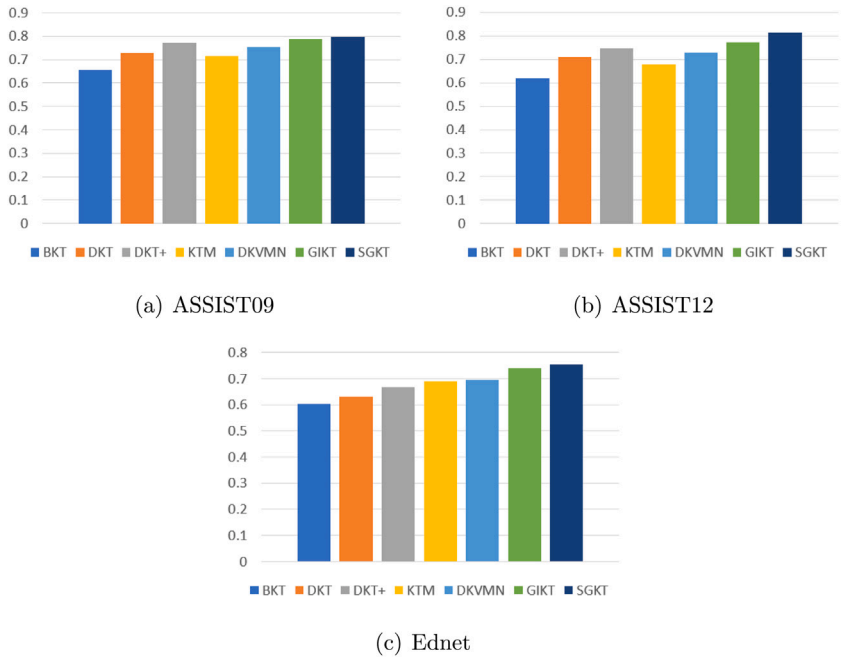
**DKT** is proposed in Piech et al. (2015). DKT uses LSTM to model students' knowledge states. DKT is the first model use to deep learning to achieve the task of knowledge tracing.

**DKT+** is proposed in Yeung and Yeung (2018), which is a modified model based on DKT. DKT+ reconstructs the regularization of the DKT loss function to address two problems of DKT, namely the inability of DKT to reconstruct the observed input, and its inconsistent prediction performance on skills across time steps.

**KTM** is proposed in Vie and Kashima (2019). KTM uses a factorization machine to interact with each feature for prediction and it can handle many types of features. This experiment considers exercises, skills, and answers as its inputs.

**DKVMN** is proposed in Zhang et al. (2017). DKVMN uses a static matrix to store all skills and a dynamic matrix to store and update students' knowledge states on skills.

**GIKT** is proposed in Yang et al. (2020). GIKT takes exercises, skills, and answers as the model's input. It uses GCN to process the exercises and skills and then loads the processing results and answers into LSTM. Finally, it uses the recap module and interaction mechanism for the prediction task.




(a) ASSIST09 (b) ASSIST12




(c) Ednet

**Fig. 5.** Comparison of AUC between SGKT and baselines.

**Table 3**
Comparison of AUC on different datasets.

| Model | ASSIST09 | ASSIST12 | EdNet |
|---|---|---|---|
| BKT | 0.6571 | 0.6204 | 0.6027 |
| DKT | 0.7300 | 0.7092 | 0.6320 |
| DKT+ | 0.7740 | 0.7459 | 0.6670 |
| KTM | 0.7169 | 0.6788 | 0.6888 |
| DKVMN | 0.7550 | 0.7283 | 0.6967 |
| GIKT | 0.7881 | 0.7719 | 0.7404 |
| **SGKT** | **0.7975** | **0.8135** | **0.7550** |

### 5.3. Performance

In SGKT, we set the embedding size of exercise and skill to 100. All parameter matrices are randomly initialized and they can be updated during the training process. In the whole process, we set the hidden size of the student's knowledge state to 100. In the embedding propagation module, we set the aggregated layers to 3. The learning and dropout rates are set to 0.00025 and 0.8, respectively, to prevent overfitting. The values of three parameters in the forgetting mechanism (i.e., $\theta_1$, $\theta_2$, and $\theta_3$) are randomly selected within the interval $(-0.1, 0.1)$. After extensive experiments, we found it most suitable to set the batch size to 6. In the process of the attention network, it is experimentally shown that considering exercise similarity is more effective than considering skill similarity. We also set the attribute bound to 0.7 to filter irrelevant embeddings in the attention network. Table 3 and Fig. 5 show that SGKT offers better AUC performance than the baseline methods.

### 5.4. Knowledge state evolution description

Knowledge state evolution description, that is, tracing the level of students' skills, is an important application of knowledge tracing models (Piech et al., 2015). The description of knowledge states can visually show the strength and weakness of each student's mastery level of each skill. Students can quickly find the shortcomings in their knowledge and be more motivated to actively address this using the knowledge state evolution depiction.

Figs. 6–8 show the heat-map of a student in the dataset ASSIST09 whose knowledge state is constantly changing on certain skills. Figs. 6–8 illustrate the difference in the knowledge state evolution using SGKT, DKT, and DKT+ in predicting a student's mastery level on four skills.

In these three figures, the label in the vertical dimension corresponds to the skills. To show the results clearly, we randomly chose four skills (ID: 14, 64, 80, and 114) in those answered by the student. The horizontal dimension shows a sequence of knowledge concepts and answers (0 indicates incorrectly answered and 1 indicates correctly answered) from the test set, where the labels refer to the knowledge concepts input into the model at each time step. The heat-map color indicates the predicted probability that the student will demonstrate the corresponding skill correctly in the next step. The darker the color, the higher the probability. As Figs. 6 and 8 show, after answering the 26 exercises, the student almost masters these four skills, but the student's mastery level fluctuates during the learning process shown in Fig. 8. However, as Fig. 7 shows, the student masters skills 64 and 114 but fails to master skills 14 and 80.

We can also observe from Figs. 6–8 that SGKT has a better knowledge state performance than DKT and DKT+. The knowledge state prediction using SGKT is more stable than DKT and DKT+. Second, since the *HRG Embedding Module* enhances the association between exercises, between skills, and between exercises and skills, an exercise which requires only one skill can affect the student's mastery level of other skills. For example, when the student repeatedly fails to demonstrate skill 80, the prediction accuracy for all skills decreases. But after the student correctly demonstrates skill 80 four consecutive times, the prediction accuracy for all skills gradually increases. On the contrary, DKT and DKT+ do not exhibit this phenomenon. In real life, such as unit exams, although different exercises may require different skills, there are potential connections among skills. Therefore, SGKT is better able to simulate reality.

Figs. 6–8 only describe the difference in knowledge state prediction stability on SGKT, DKT, and DKT+. To give a better illustration, we further compare the overall waviness of the knowledge state prediction on SGKT, DKT, and DKT+ at each time step in an exercise session. We plot the change in the knowledge state prediction of these four skills (ID: 14, 64, 80, and 114) using SGKT, DKT, and DKT+ in Fig. 9 to display the prediction waviness distribution at each time step. Fig. 9 also shows

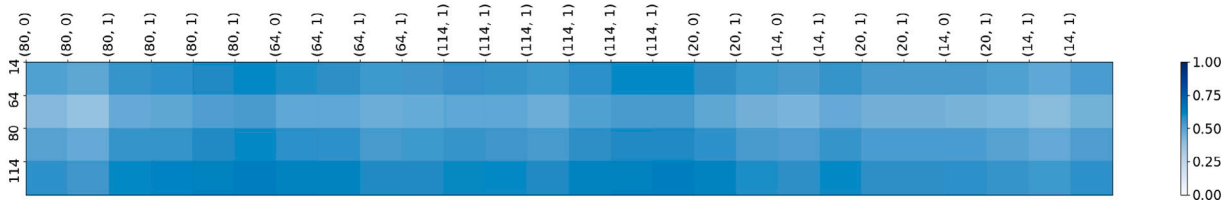



**Fig. 6.** The knowledge state prediction of one student using SGKT.

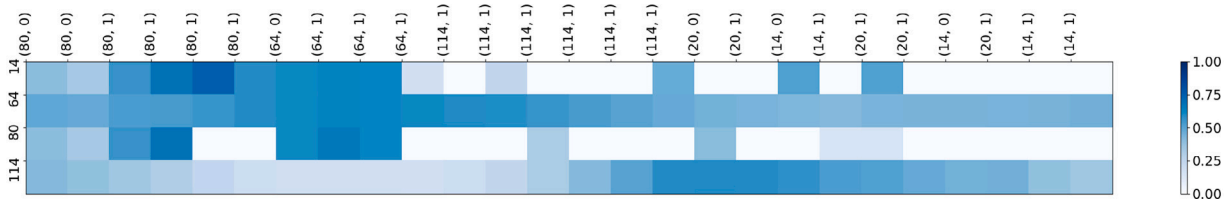



**Fig. 7.** The knowledge state prediction of one student using DKT.

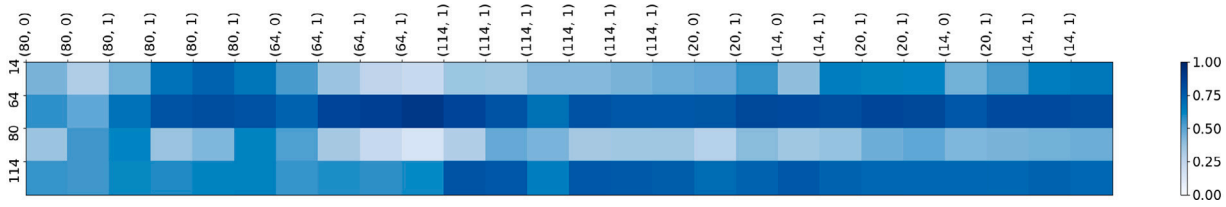



**Fig. 8.** The knowledge state prediction of one student using DKT+.

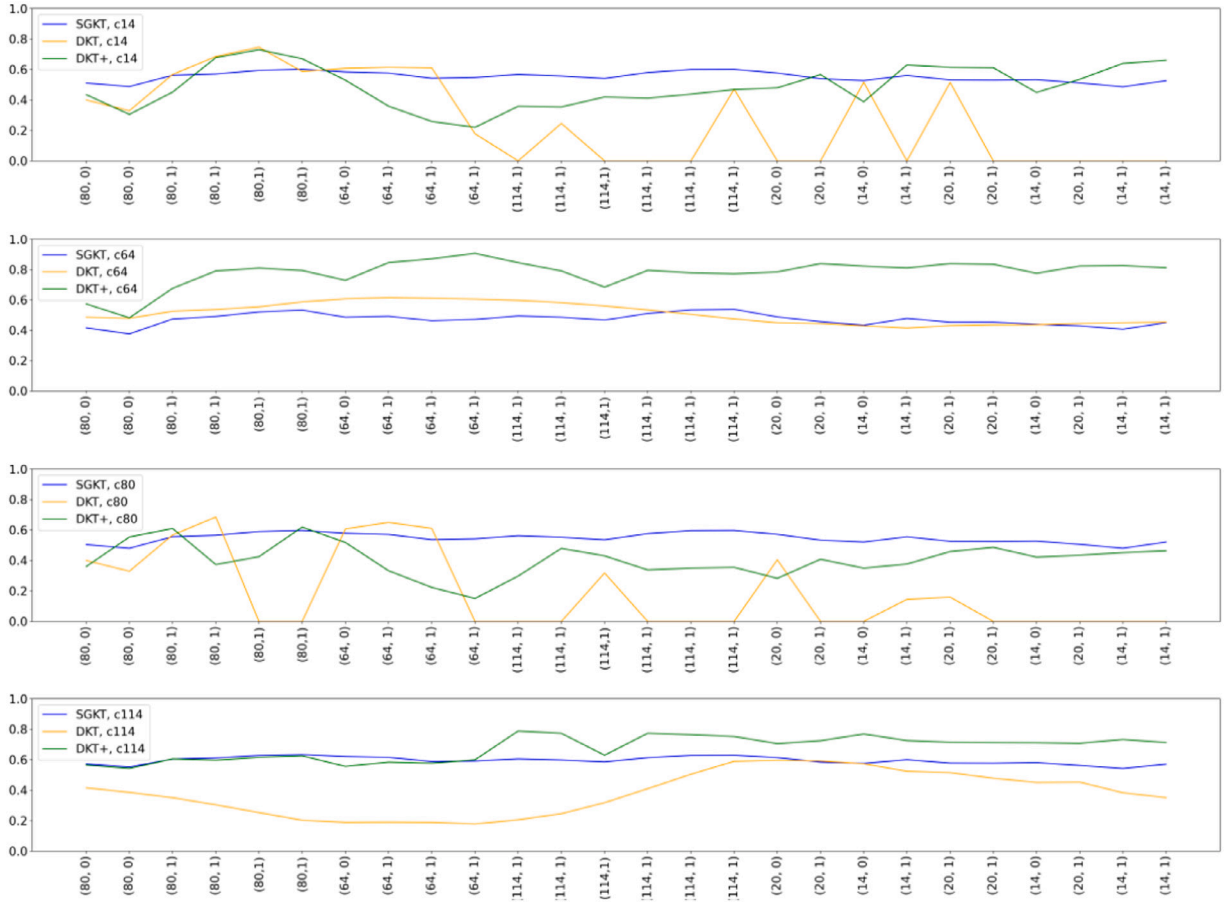



**Fig. 9.** Comparison of the waviness of an individual student's knowledge state prediction using SGKT, DKT, and DKT+.

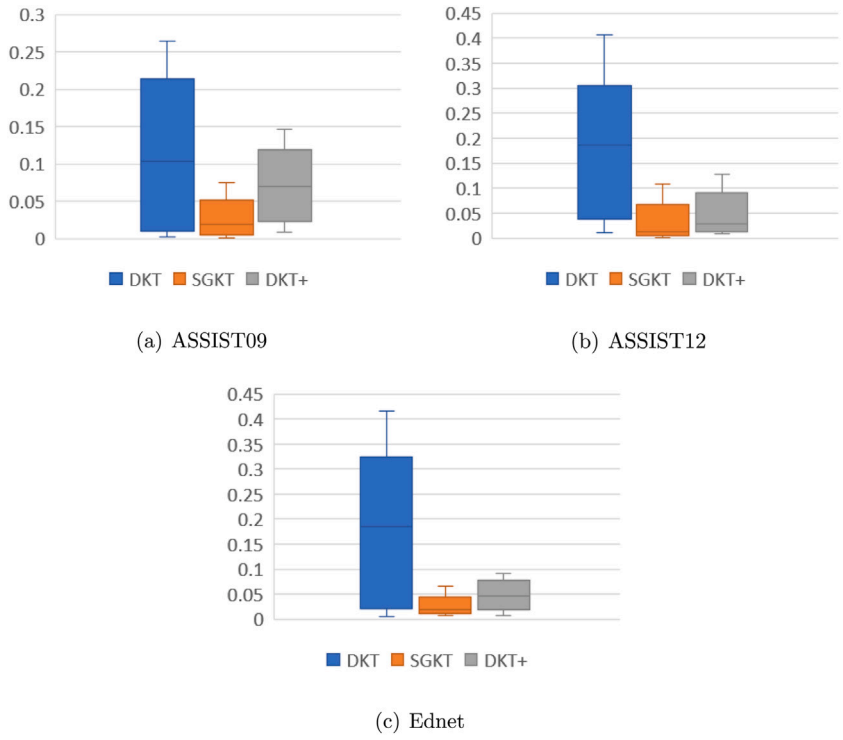



**Fig. 10.** Comparison of the waviness of the knowledge state prediction using DKT, SGKT, and DKT+ in the test set.

that the knowledge state prediction curve of SGKT is smoother than that of DKT and DKT+.

Prediction waviness is defined as follows: for an individual student, prediction waviness can be obtained according to the observed input and the change in the corresponding prediction at two different time steps. We denote prediction waviness by $\Delta_t^i$, which is obtained from Eq. (15) :

$$\Delta_t^i = |y_t^i - y_{t-1}^i|, \tag{15}$$

where $y_{t-1}^i$ and $y_t^i$ represent the knowledge state prediction of $i$th skill at time-step $t$ and $t-1$.

Fig. 10 describes the knowledge state prediction waviness distribution of all students in the test set on DKT, SGKT, and DKT+ using boxplots. A boxplot is a graph with a good indication of how the data are spread. It is a standardized way of displaying a dataset based on a five-number summary: the minimum, the maximum, median, and the first and third quartiles. A boxplot can visually show the distribution of numerical data by displaying the data quartiles and averages. From Fig. 10, we can observe that the prediction waviness distributions of DKT and DKT+ are more comprehensive than that of SGKT, and their medians are also higher than our proposed approach, indicating that the prediction ability of our approach is more stable than DKT and DKT+.

### 5.5. Cold start

Knowledge tracing also faces the *cold start* issue. Referring to the literature (Zhao et al., 2020), we study the performance in two cold start scenarios, i.e., *training with data of a few students* and *new students with short exercise sequences*.

- scenario 1: *Training with data of a few students* is to train the knowledge tracing model with as few students' interactions as possible and applies it to new unseen samples.
- scenario 2: *New students with short exercise sequences* occurs when new students enter the online learning system and the lack of exercise records result in insufficient characteristics to input into the knowledge tracing model.

In scenario 1 *Training with data of few students*, we conducted experiments with different student numbers ranging from 0.1 to 15 percent of the training dataset to compare the prediction performance of SGKT, DKT, and DKT+. As shown in Fig. 11, SGKT achieves better performance than DKT on the ASSIST12 and Ednet datasets and performs better than DKT+ on the Ednet dataset. On ASSIST09, the performance of SGKT is lower than DKT before the students' data reaches 7% and is lower than DKT+ before the students' data reaches 10%. On ASSIST12, the performance of SGKT is lower than DKT+ before the students' data reaches 2%. The reason for this is that the number of repeated exercises in the exercise session is small in this dataset.

In scenario 2 *New students with short exercise sequences*, we first categorize the training data into eight groups, each with a different range of exercise sequence lengths, i.e., $(0,5]$, $(0,10]$, $(0,15]$, $(0,20]$, $(0,25]$, $(0,50]$, $(0,75]$, and $(0,100]$. Each exercise sequence in the training data is constructed by intercepting the corresponding lengths of exercises from the original exercise sequence. It is easy to see that the case for the first group is the most difficult, since students from this group have the fewest exercise answering records. We compare the performance of the different methods in Fig. 12. In this scenario, SGKT achieves better performance than DKT and DKT+ on the ASSIST09 and Ednet datasets. When the length of the sequence of the exercises is less than $(0,50]$, the performance of SGKT on the ASSIST12 dataset is lower than that of DKT and DKT+. The reason for this is that the continuous exercise sequence of students in the testing data is usually very long (the average length is 99.1), which leads to insufficient construction of the session graph due to the short exercise sequence data and during the training of SGKT.

### 5.6. Ablation study

In this section, we conduct an ablation study of the proposed approach through two experiments.

#### 5.6.1. Component adjustment

The first experiment observes the changes in the performance of the various approaches by adjusting the model's components. The results

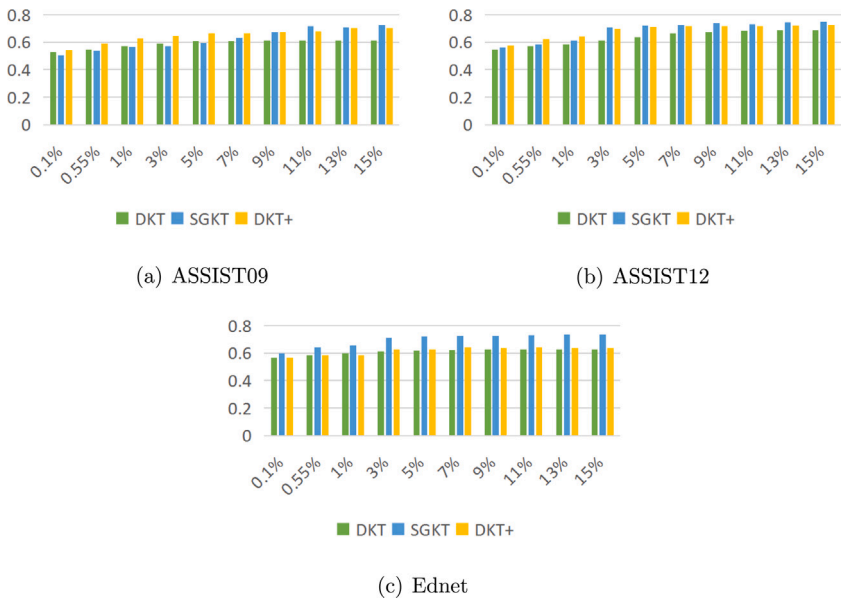



**Fig. 11.** Comparison of the performance of DKT, SGKT and DKT+ with the cold start problem of training data on a few students (scenario 1).

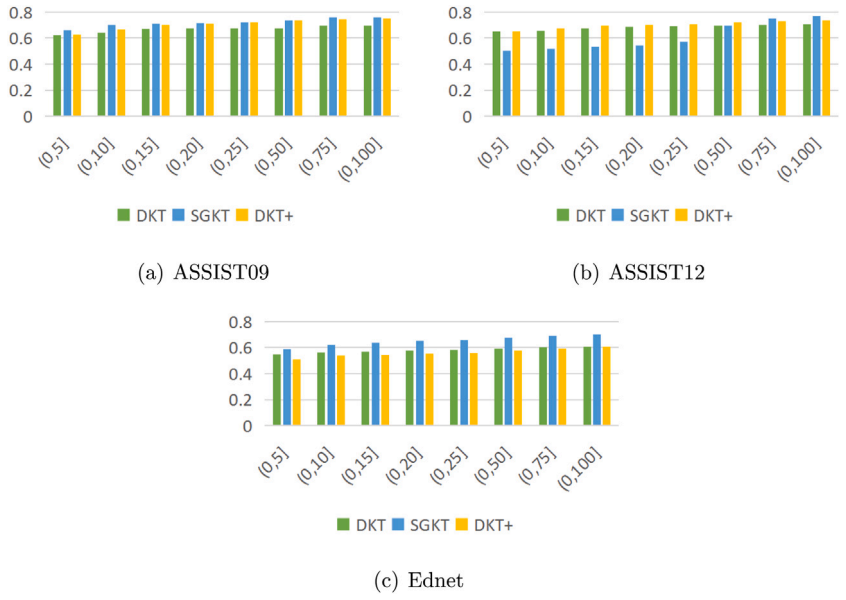



**Fig. 12.** Comparison of the performance of DKT, SGKT, and DKT+ with the cold start problem of new students (scenario 2).

**Table 4**
Comparison of the performance of the SGKT based on the different submodules.

| Model | ASSIST09 | ASSIST12 | EdNet |
|---|---|---|---|
| SGKT-BG&NF | 0.7903 | 0.7744 | 0.7439 |
| SGKT-RG&NF | 0.7859 | 0.7793 | 0.7475 |
| SGKT-BG&FM | 0.7910 | 0.8129 | **0.7572** |
| **SGKT** | **0.7975** | **0.8135** | 0.7545 |

are shown in Table 4. **SGKT-BG&NF** is SGKT with a bi-graph and without the forgetting mechanism. **SGKT-RG&NF** is SGKT with a HRG and without the forgetting mechanism. **SGKT-BG&FM** is SGKT with a bi-graph and with the forgetting mechanism. This table shows that on datasets ASSIST09 and ASSIST12, SGKT performs the best of all the methods. However, SGKT-BG&FM has a better performance on the EdNet dataset.

### 5.6.2. *Hidden size of knowledge state*

The second experiment is to observe the impact of the different hidden sizes of the knowledge state. The results are shown in Fig. 13. The hidden size ranges from 5 to 100. Fig. 13 shows that the hidden size has a different effect on the performance of SGKT compared to the baseline methods. SGKT achieves better performance than the baseline methods with varying hidden sizes of the knowledge state. As shown in Fig. 13(a), on ASSIST09, DKT+ outperforms SGKT when the hidden size is lower than 10.

## 6. Conclusion and future work

In this work, we propose a session graph-based knowledge tracing model. Our approach builds a directed graph using students' exercise-answer records and builds a heterogeneous relation graph of exercises and skills. Then, we apply GGNN and GCN to generate the student's knowledge state, which contains the local context information of the

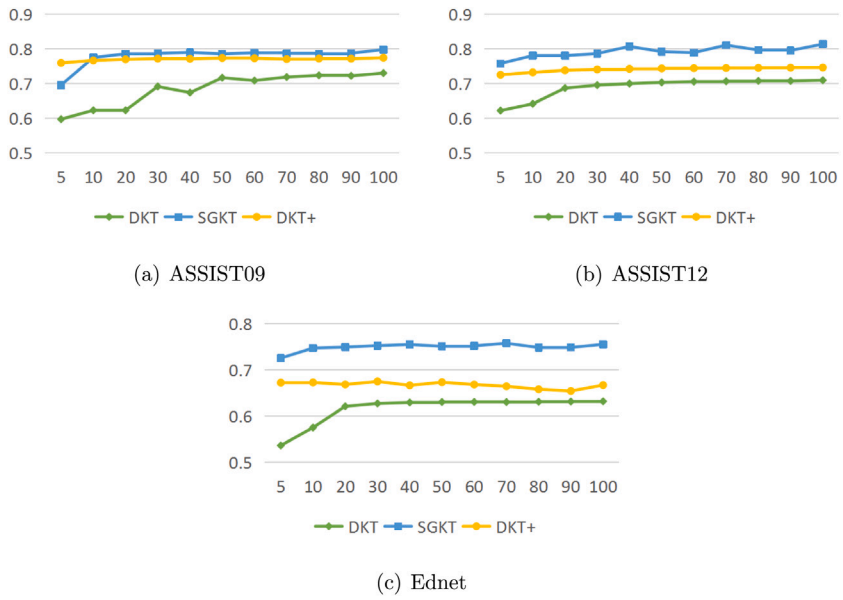



**Fig. 13.** Comparison of the impact of knowledge states of differing hidden sizes.

exercise-answer record and the embedding representation of all exercises and skills. Next, we use the self-attention network with a forgetting mechanism to capture the global dependencies between distant locations. Finally, we combine short-term dynamics (the most recently answered exercise) and the dependence of global self-attention to express the exercise answer sequence in a linear mechanism. We validate the advantages of SGKT on several real-world datasets used in knowledge tracing, compared to some existing methods in terms of AUC, knowledge state evolution description and cold start. Furthermore, we conduct an ablation study.

Knowledge tracing is a research hotspot in the field of educational data mining. Traditional knowledge tracing methods take exercise answer history as a sequence, ignoring the relationships between the exercises formed during the answering process. In this work, we improved the knowledge tracing model to understand the knowledge state from the exercise session graph node information. Cold start is a new problem in current knowledge tracing research. Although this work considers the cold start issue in the experiments, our model did not perform any optimization and more analysis is needed. We will continue to work on this problem in the future.

## CRediT authorship contribution statement

**Zhengyang Wu:** Conceptualization, Methodology, Writing. **Li Huang:** Software, Writing – original draft. **Qionghao Huang:** Reviewing and editing. **Changqin Huang:** Data planning, Visualization. **Yong Tang:** Investigation.

## Acknowledgments

This research was supported by the National Natural Science Foundation of China (NSFC) under the Grant No. U1811263, 62037001, and 61877020. We would like to thank the anonymous reviewers for their constructive advice.

## References

Baker, R. S. J. D., Corbett, A. T., & Aleven, V. (2008). More accurate student modeling through contextual estimation of slip and guess probabilities in Bayesian knowledge tracing. In *International conference on intelligent tutoring systems* (pp. 406–415). Springer.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V. F., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gülçehre, Ç., Song, H. F., Ballard, A. J., Gilmer, J., Dahl, G. E., Vaswani, A., Allen, K. R., Nash, C., Langston, V., .... Pascanu, R. (2018). Relational inductive biases, deep learning, and graph networks. CoRR, abs/1806.01261.

Chanaa, A., & Faddouli, N. E. (2020). Predicting learners need for recommendation using dynamic graph-based knowledge tracing. In I. I. Bittencourt, M. Cukurova, K. Muldner, R. Luckin, & E. Millán (Eds.), *Lecture notes in computer science: vol. 12164, Artificial intelligence in education - 21st international conference, AIED 2020, Ifrane, Morocco, July 6-10, 2020, proceedings, part II* (pp. 49–53). Springer.

Chen, J., Lu, Y., Shang, F., & Wang, Y. (2021). A fuzzy matrix factor recommendation method with forgetting function and user features. *Applied Soft Computing, 100*, Article 106910.

Choi, Y., Lee, Y., Shin, D., Cho, J., Park, S., Lee, S., Baek, J., Bae, C., Kim, B., & Heo, J. (2020). Ednet: A large-scale hierarchical dataset in education. In *International conference on artificial intelligence in education* (pp. 69–73). Springer.

Corbett, A. T., & Anderson, J. R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Model. User-Adapt. Interact., 4*(4), 253–278.

Graves, A., Mohamed, A.-r., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing* (pp. 6645–6649). IEEE.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *5th international conference on learning representations, ICLR 2017, Toulon, France, April 24-26, 2017, conference track proceedings*. OpenReview.net.

Li, Y., Tarlow, D., Brockschmidt, M., & Zemel, R. S. (2016). Gated graph sequence neural networks. In Y. Bengio, & Y. LeCun (Eds.), *4th international conference on learning representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, conference track proceedings*.

Lin, C., & Chi, M. (2016). Intervention-bkt: incorporating instructional interventions into Bayesian knowledge tracing. In *International conference on intelligent tutoring systems* (pp. 208–218). Springer.

Liu, Q., Huang, Z., Yin, Y., Chen, E., Xiong, H., Su, Y., & Hu, G. (2021). EKT: exercise-aware knowledge tracing for student performance prediction. *IEEE Transactions on Knowledge and Data Engineering, 33*(1), 100–115.

Liu, Q., Shen, S., Huang, Z., Chen, E., & Zheng, Y. (2021). A survey of knowledge tracing. CoRR, abs/2105.15106.

Liu, Y., Yang, Y., Chen, X., Shen, J., Zhang, H., & Yu, Y. (2020). Improving knowledge tracing via pre-training question embeddings. In C. Bessiere (Ed.), *Proceedings of the twenty-ninth international joint conference on artificial intelligence, IJCAI 2020* (pp. 1577–1583). ijcai.org.

Liu, S., Zou, R., Sun, J., Zhang, K., Jiang, L., Zhou, D., & Yang, J. (2021). A hierarchical memory network for knowledge tracing. *Expert Systems with Applications, 177*, Article 114935.

McGill, T. J., & Klobas, J. E. (2009). A task-technology fit view of learning management system impact. *Computer and Education, 52*(2), 496–508.

Miah, S. J., Miah, M., & Shen, J. (2020). Editorial note: Learning management systems and big data technologies for higher education. *Education and Information Technologies, 25*(2), 725–730.

Minn, S., Yu, Y., Desmarais, M. C., Zhu, F., & Vie, J.-J. (2018). Deep knowledge tracing and dynamic student classification for knowledge tracing. In *2018 IEEE international conference on data mining (ICDM)* (pp. 1182–1187). IEEE.

Mongkhonvanit, K., Kanopka, K., & Lang, D. (2019). Deep knowledge tracing and engagement with MOOCs. In *Proceedings of the 9th international conference on learning analytics & knowledge* (pp. 340–342).

Nakagawa, H., Iwasawa, Y., & Matsuo, Y. (2019). Graph-based knowledge tracing: modeling student proficiency using graph neural net. In *Proceedings of the international conference on learning representations (ICLR)* (pp. 1–8).

Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L. J., & Sohl-Dickstein, J. (2015). Deep knowledge tracing. In *Advances in neural information processing systems 28: Annual conference on neural information processing systems 2015, December 7-12, 2015, Montreal, Quebec, Canada* (pp. 505–513).

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks and Learning Systems*, *20*(1), 61–80.

Song, X., Li, J., Lei, Q., Zhao, W., Chen, Y., & Mian, A. (2022). Bi-CLKT: Bi-graph contrastive learning based knowledge tracing. *Knowledge Based Systems*, *241*, Article 108274.

Song, X., Li, J., Tang, Y., Zhao, T., Chen, Y., & Guan, Z. (2021). JKT: a joint graph convolutional network based deep knowledge tracing. *Information Sciences*, *580*, 510–523.

Sweeney, M., Lester, J., Rangwala, H., & Johri, A. (2016). Next-term student performance prediction: A recommender systems approach. In T. Barnes, M. Chi, & M. Feng (Eds.), *Proceedings of the 9th international conference on educational data mining, EDM 2016, Raleigh, North Carolina, USA, June 29 - July 2, 2016* (p. 7). International Educational Data Mining Society (IEDMS).

Tomasevic, N., Gvozdenovic, N., & Vranes, S. (2020). An overview and comparison of supervised data mining techniques for student exam performance prediction. *Computer and Education*, *143*.

Vie, J.-J., & Kashima, H. (2019). Knowledge tracing machines: Factorization machines for knowledge tracing. In *Proceedings of the AAAI conference on artificial intelligence, Vol. 33* (pp. 750–757).

Wang, L., Sy, A., Liu, L., & Piech, C. (2017). Deep knowledge tracing on programming exercises. In *Proceedings of the fourth (2017) ACM conference on learning@ scale* (pp. 201–204).

Xiong, X., Zhao, S., Inwegen, E. V., & Beck, J. (2016). Going deeper with deep knowledge tracing. In *Proceedings of the 9th international conference on educational data mining* (pp. 545–550). IEDMS.

Yang, Y., Shen, J., Qu, Y., Liu, Y., Wang, K., Zhu, Y., Zhang, W., & Yu, Y. (2020). GIKT: a graph-based interaction model for knowledge tracing. In *Lecture notes in computer science*: *vol. 12457*, *Machine learning and knowledge discovery in databases - European conference* (pp. 299–315). Springer.

Yeung, C.-K., & Yeung, D.-Y. (2018). 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). ACM.

Zhang, J., Shi, X., King, I., & Yeung, D.-Y. (2017). Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th international conference on world wide web* (pp. 765–774).

Zhang, K., & Yao, Y. (2018). A three learning states Bayesian knowledge tracing model. *Knowledge-Based Systems*, *148*, 189–201.

Zhao, J., Bhatt, S. P., Thille, C., Gattani, N., & Zimmaro, D. (2020). Cold start knowledge tracing with attentive neural turing machine. In *Proceedings of the 7th ACM conference on learning @ scale, virtual event* (pp. 333–336). ACM.