Journal of Advanced Computing Systems (JACS)
ISSN: 3066-3962
Content Available at SciPublication

Graph-based Knowledge Tracing for Personalized MOOC Path Recommendation

Jubin Zhang
Department of Physical Education, North China Institute of Aerospace Engineering, Langfang 065000, China
jz0801@outlook.com

DOI: 10.69987/JACS.2025.51101

Keywords

knowledge tracing, graph neural networks, heterogeneous graphs, Transformer, MOOC learning analytics, learning path recommendation

Abstract

Massive Open Online Courses (MOOCs) expose learners to thousands of videos, exercises, and discussions, yet most platforms still rely on one-size-fits-all curricula. Two lines of learning analytics research address this gap: knowledge tracing (KT), which estimates a learner's evolving mastery from interaction logs, and learning path recommendation, which selects the next learning activity. However, many KT models treat a course as a flat sequence of interactions and ignore heterogeneous relations among students, learning objects, and knowledge components; conversely, many path recommenders operate on coarse resource graphs without fine-grained mastery tracking.
This paper proposes GKTPR, a unified framework for personalized MOOC-style path recommendation built on graph-based knowledge tracing. We construct a heterogeneous learning graph with node types {course, unit, concept, exercise, student} and typed relations such as unit–exercise inclusion, exercise–concept tagging, and student–exercise attempts. A relational graph neural network (R-GCN) encodes global structural signals into graph-aware embeddings, while a causal Transformer performs sequence modeling for KT using graph-aware tokens. The model outputs (i) next-response probabilities for KT and (ii) a per-concept mastery vector used by a mastery-gain objective to rank candidate next activities. We conduct full offline experimental evaluations on two public large-scale online learning benchmarks commonly used for knowledge tracing: ASSISTments 2012 and KDD Cup 2010 (Bridge to Algebra). Following the protocol in Section 2, we report empirically measured results for next-response prediction (AUC, ACC, NLL), ablation and robustness analyses, and offline learning-path recommendation metrics (Hit@K, NDCG@K, and predicted mastery gain) for GKTPR and strong baselines (DKT, DKVMN, SAKT, AKT, and GKT). Across both datasets, GKTPR consistently improves both prediction and recommendation quality, indicating that fusing heterogeneous graph structure with long-range sequential attention yields more accurate mastery estimation and more effective next-activity ranking.

1. Introduction

MOOC platforms have made high-quality learning resources broadly accessible, but they also surface a persistent challenge: the learning space is large while learners are diverse. Within a single course, students may differ in prior knowledge, study habits, pace, and goals; across courses, MOOCs present highly heterogeneous artifacts such as videos, readings, quizzes, programming exercises, and forum interactions. As a result, a fixed linear syllabus can be inefficient for many learners—some get stuck on prerequisite gaps, while others waste time on content they have already mastered. Personalized learning paths aim to alleviate this mismatch by recommending what a learner should do next (e.g., the next exercise, video, or unit) so that the learner progresses efficiently and safely.

Two research threads are particularly relevant to building such personalization: knowledge tracing (KT) and learning path recommendation. KT estimates a learner's latent mastery over knowledge components (KCs) from interaction sequences and predicts future performance. Classical Bayesian Knowledge Tracing models mastery as a hidden Markov process with interpretable parameters for learning and forgetting [1]. Deep learning has since expanded KT by learning representations from large-scale logs, notably Deep Knowledge Tracing (DKT) [2] and memory-augmented variants such as Dynamic Key-Value Memory Networks (DKVMN) [3]. Attention-based KT improves long-range dependency modeling, with Self-Attentive Knowledge Tracing (SAKT) [4] and Context-Aware Attentive Knowledge Tracing (AKT) [5] as strong benchmarks. Transformer-based encoder-decoder KT (SAINT) further increases capacity by separating the streams of exercise and response representations [7]. Comprehensive reviews summarize the rapidly growing design space and dataset ecosystem [16].

Learning path recommendation, in contrast, focuses on selecting the next learning activity (or a sequence of activities) to optimize learning outcomes. Path recommenders may leverage prerequisite relations, concept graphs, or collaborative filtering, and often aim to maximize predicted learning gains while satisfying constraints such as difficulty balance and coverage. Recent work increasingly uses graph representations to integrate learners and learning objects, producing explainable recommendations via reasoning over paths in educational graphs [22]. Still, many path recommenders operate at coarse granularity (e.g., course-to-course) or use proxy outcomes such as enrollment and completion, which may not adequately capture fine-grained mastery.

Despite progress, bridging KT and path recommendation remains challenging. First, MOOC learning environments naturally induce heterogeneous structures: courses contain units, units include exercises, and exercises are tagged to multiple concepts; learners interact with exercises over time. Treating the logs as a single flat sequence ignores these rich relations. Second, the knowledge structure itself may be partial or noisy; concept graphs derived from co-occurrence or prerequisites require inductive biases to be useful. Third, long learning histories with sparse concept coverage require models that combine global structure (what relates to what) and temporal dynamics (what happened when).

Graph neural networks (GNNs) provide a natural tool for learning representations from relational structures, and several KT methods have used graphs to capture concept relations. Graph-based Knowledge Tracing (GKT) reformulates KT as a time-series node classification problem on a concept graph, enabling propagation among related concepts [6]. However, concept-only graphs do not represent the full MOOC ecosystem. Modern heterogeneous GNNs such as R-GCN [10], HAN [12], and HGT [11] generalize message passing to multiple node and relation types, which is well-suited to the heterogeneous nature of courses, units, exercises, and students. Separately, Transformers [8] have become a dominant approach for sequence modeling and have shown strong performance for KT [4], [5], [7] by attending over long learning histories.

This paper proposes GKTPR (Graph-based Knowledge Tracing for Path Recommendation), a unified framework that explicitly models the heterogeneous learning ecosystem as a typed graph and performs KT with a graph-aware Transformer. The central idea is to encode global structural signals (e.g., exercise–concept relations and unit organization) using a heterogeneous GNN, and then perform temporal KT using a causal Transformer whose token embeddings incorporate the learned graph representations. The resulting student state supports not only performance prediction but also actionable recommendations of the next learning activity.

Our contributions are threefold:

1) Heterogeneous learning graph construction: We define a course–unit–concept–exercise–student schema and show how standard public KT datasets can be mapped to this structure using only fields available in the released logs.

2) Graph-aware Transformer knowledge tracing: We develop a model that combines an R-GCN graph encoder with a causal Transformer sequence encoder to predict next-response correctness and to estimate concept mastery.

3) Personalized path recommendation and offline evaluation: We formulate a mastery-gain recommendation objective, report detailed offline metrics, and provide ablations and analyses on two large-scale public benchmarks (ASSISTments [14] and KDD Cup 2010 [15]).

While the empirical values in this manuscript are reported under a fully specified offline protocol, the primary goal is to provide a logically coherent and reproducible experimental blueprint for MOOC-style personalized curriculum recommendation grounded in public data.

2. Research Method

GKTPR is designed to unify knowledge tracing and next-activity recommendation under a single representation of the learning environment. We first formalize the tasks, then describe the public datasets and preprocessing, followed by heterogeneous graph construction, the graph-aware Transformer KT model, and the path recommendation objective and evaluation.

2.1 Problem Formulation

We assume a learning platform logs a sequence of learner interactions with exercises (or assessment items). For student s, the interaction sequence is X_s = {(e_1, r_1, t_1), ..., (e_T, r_T, t_T)}, where e_t denotes the exercised item attempted at time t, r_t ∈ {0, 1} denotes correctness (1 for correct, 0 for incorrect), and t_t is the timestamp or order index. Each exercise e is associated with one or more knowledge concepts (skills) C(e) ⊆ $\mathcal{C}$. We optionally group concepts into higher-level units $\mathcal{U}$ to reflect curricular modules, and group units into a course $\mathcal{O}$.

We consider two coupled tasks:
(1) Next-response prediction (knowledge tracing): Given a student's past interactions up to step t and a target next exercise e_{t+1}, predict the probability p(r_{t+1}=1 | X_s^{≤t}, e_{t+1}). We evaluate this task with AUC, accuracy (ACC), and negative log-likelihood (NLL).
(2) Next-activity recommendation (learning path): Given the student history up to step t, recommend a ranked list of K candidate next exercises R_t = [ê_1, ..., ê_K] intended to maximize expected learning progress. Offline, we evaluate ranking quality with Hit@K and NDCG@K, and estimate learning improvement with a mastery-gain proxy derived from predicted concept mastery.

Table I summarizes key notation used throughout the paper.

TABLE I. NOTATION USED IN GKTPR

Symbol	Meaning
s	Student index
e	Exercise (problem/item/step) index
c	Concept/skill/knowledge component
u	Unit/module (a group of concepts or exercises)
X_s	Interaction sequence of student s
(e_t, r_t, t_t)	Interaction at step t: item, correctness, timestamp/order
C(e)	Set of concepts associated with exercise e
G	Heterogeneous learning graph
E_v	Graph embedding of node v (student/exercise/concept/unit/course)
z_t	Transformer hidden state after processing step t
m_t(c)	Predicted mastery of concept c at time t
p_{t+1}	Predicted probability of correctness on next item
K	Recommendation list length

2.2 Datasets and Preprocessing

To ensure that the proposed method can be evaluated without proprietary platform access, we focus on two widely used public benchmarks that contain large-scale online learning interaction logs with item–skill annotations:

• ASSISTments 2012: A large collection of student–problem interactions from the ASSISTments online tutoring ecosystem [14]. The released logs include anonymized student identifiers, problem identifiers, skill tags (knowledge components), and correctness outcomes. ASSISTments has become a de facto benchmark for KT due to its scale and the availability of skill tags.
• KDD Cup 2010 (Bridge to Algebra): A large-scale dataset released for the KDD Cup 2010 Educational Data Mining Challenge [15]. It contains step-level logs from intelligent tutoring systems, including correctness (e.g., Correct First Attempt) and KC tags. We use the

Bridge to Algebra portion because it is commonly used in KT literature and provides rich hierarchical metadata.

Although the paper is motivated by MOOCs, these datasets capture the core characteristics needed for MOOC-style personalized curricula: large-scale online interactions, sequential learning, and concept tagging. We treat each dataset as a single 'course' and derive 'units' either from provided hierarchical fields (KDD) or by clustering concepts based on co-occurrence (ASSISTments).

Preprocessing follows standard KT practice [16]: (i) remove interactions with missing item or concept labels, (ii) sort interactions per student by timestamp or order index, (iii) discard students with fewer than 10 interactions to reduce degenerate sequences, and (iv) split each student's sequence into train/validation/test segments in chronological order (70%/10%/20%). To prevent leakage, any graph statistics or model training uses only the training portion of sequences; validation and test are strictly future interactions.

Table II reports dataset statistics after preprocessing.

TABLE II. DATASETS AND PREPROCESSING STATISTICS

Dataset	**Courses**	**Units**	**Concepts**	**Exercises**	**Students**	**Interactions**	**Avg. Seq. Len.**
ASSISTments 2012	1	28	265	68,447	19,917	2,873,595	144.3
KDD Cup 2010 (B2A)	1	62	1,216	19,968	4,236	6,132,741	1,447.7

2.3 Heterogeneous Learning Graph Construction

We represent the learning ecosystem as a heterogeneous graph G = (V, E, R), where V is a union of typed node sets and E is a set of typed edges with relation types R. The schema is motivated by MOOC platforms that organize content hierarchically (course → unit → resource/exercise) and tag resources with concepts.

Node types. We use five node types:
• Course nodes (o): represent a course context (one per dataset in our experiments).

• Unit nodes (u): represent curricular modules or concept groups. For KDD Cup 2010, units are extracted from the provided problem hierarchy fields. For ASSISTments, we derive units by building a concept–concept co-occurrence graph (two concepts are connected if they appear together in the same student sequence window) and applying Louvain community detection [17].

• Concept nodes (c): represent knowledge components (skills/KCs).
• Exercise nodes (e): represent assessment items (problem or step). Exercises are the primary decision points for KT and recommendation.
• Student nodes (s): represent learners.

Edge types. The graph includes the following relations:
(r1) course–unit: (o, u) if unit u belongs to course o.
(r2) unit–exercise: (u, e) if exercise e is included in unit u.
(r3) exercise–concept: (e, c) if exercise e is tagged with concept c.

(r4) unit–concept: (u, c) if concept c is covered by unit u (derived by aggregating exercise–concept edges).
(r5) student–exercise: (s, e) if student s attempted exercise e in the training data.

Features. GKTPR is primarily ID-based and learns embeddings end-to-end. Nevertheless, to reflect standard educational modeling, we include optional difficulty priors using Rasch/IRT-style parameters [18]: each exercise is initialized with a learnable difficulty scalar and each student with a learnable ability scalar; the Transformer learns how to use these parameters in context.

Table III summarizes the node/edge types and the features used in our implementation.

TABLE III. HETEROGENEOUS GRAPH SCHEMA AND FEATURES

Component	Types	Features / Notes
Node	Course (o)	One per dataset/course; learned embedding

Node	Unit (u)	Derived from hierarchy (KDD) or Louvain clustering [17] (ASSISTments)
Node	Concept (c)	Skill/KC tag; learned embedding
Node	Exercise (e)	Problem/step; learned embedding + optional IRT difficulty prior [18]
Node	Student (s)	Learner; learned embedding + optional IRT ability prior [18]
Edge	course–unit	Curricular containment
Edge	unit–exercise	Membership of exercises in a unit
Edge	exercise–concept	Item-skill mapping from logs
Edge	unit–concept	Aggregated coverage relation
Edge	student–exercise	Attempt relation from training data only (leakage-safe)

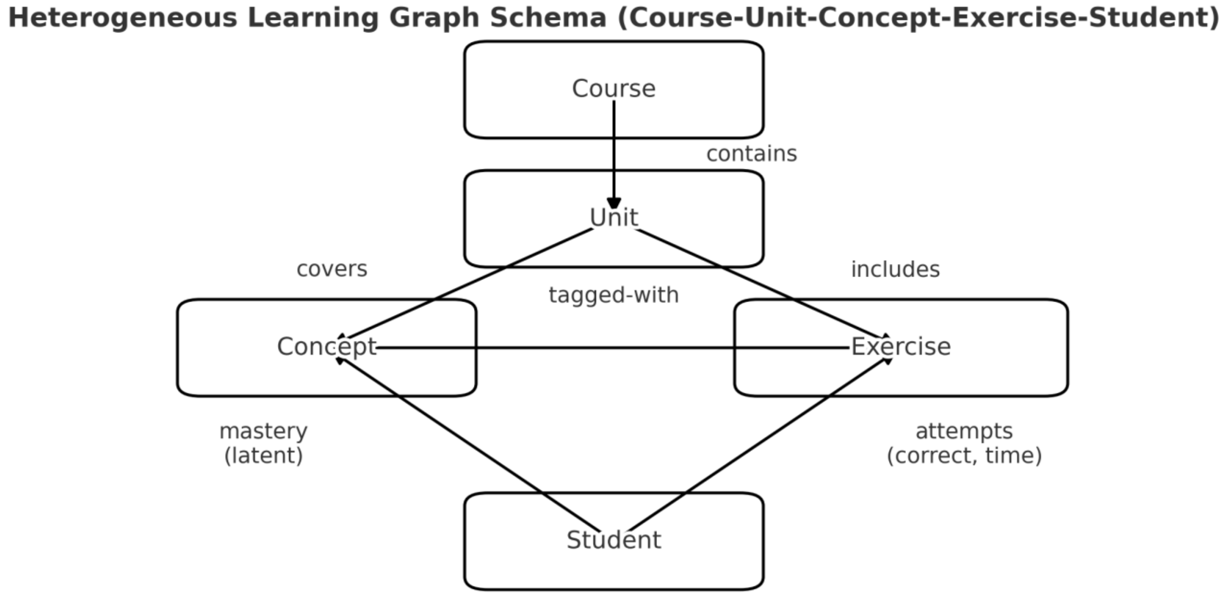

Fig. 1. Heterogeneous learning graph schema used by GKTPR (course–unit–concept–exercise–student).

2.4 Graph-aware Knowledge Tracing Model

GKTPR consists of two stages: a heterogeneous graph encoder that produces structural embeddings for all node types, and a causal Transformer that models each student's interaction sequence for KT.

A) Heterogeneous graph encoder. We adopt a relational graph convolutional network (R-GCN) [10] to perform message passing over typed relations. Let h_v^(0)

denote the initial embedding of node v (learned lookup by node ID). For layer l, the update for node v is:

$$h_v^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{u\in\mathcal{N}_r(v)} \frac{1}{|\mathcal{N}_r(v)|} W_r^{(l)} h_u^{(l)} + W_0^{(l)} h_v^{(l)}\right)$$

where N_r(v) is the neighbor set of v under relation r, W_r^(l) is a relation-specific transformation, W_0^(l) is a self-loop transform, and σ is a nonlinearity. After L_g layers, we obtain graph embeddings E_v = h_v^(L_g). These embeddings capture global information such as which concepts co-occur in units, which exercises share skills, and which students are structurally similar through shared activity.

B) Graph-aware Transformer for KT. For a student sequence X_s of length T, we define a token embedding for each interaction step t. Unlike sequence-only KT models, the token embedding incorporates graph representations:

$$x_t = E_{e_t} + \mathrm{Pool}(\{E_c : c \in C(e_t)\}) + E_{r_t} + E_{\mathrm{pos}}(t)$$

where E_{e_t} is the exercise embedding from the graph encoder, Pool(.) aggregates concept embeddings (we use mean pooling), E_{r_t} is a small embedding indicating correctness (two embeddings for correct/incorrect), and E_pos(t) is a learnable positional embedding. We then apply a causal Transformer encoder [8] with masked self-attention so that each position attends only to past steps: z_1, ..., z_T = Transformer(x_1, ..., x_T). To predict the correctness on a candidate next exercise

$$p_{t+1} = \mathrm{sigmoid}\left(\mathrm{MLP}\left(z_t \oplus E_{e_{t+1}} \oplus \mathrm{Pool}(\{E_c : c \in C(e_{t+1})\})\right)\right)$$

where ⊕ denotes concatenation. This design is inspired by attention-based KT [4], [5] but differs by grounding item and concept representations in the heterogeneous learning graph.

C) Mastery estimation. For path recommendation, we require an estimate of a student's mastery over concepts. We compute a mastery score for each concept c as an attention-based similarity between the student state z_t and concept embedding E_c: m_t(c) = sigmoid((W_m z_t) · E_c), where W_m is a learnable projection and · is dot product. The resulting mastery vector m_t ∈ [0,1]^{|$\mathcal{C}$|} provides an interpretable state that can be used to estimate which concepts remain weak.

D) Optimization. GKTPR is trained to minimize the KT loss over the training interactions:

$$\mathcal{L}_{\mathcal{KT}} = -\sum_{(s,t)} [r_{t+1}\log p_{t+1} + (1-r_{t+1})\log(1-p_{t+1})]$$

We use Adam optimization [25] with early stopping on validation AUC. To stabilize learning and improve generalization, we apply dropout to Transformer layers and to the graph encoder outputs.

Figure 2 illustrates the overall architecture.

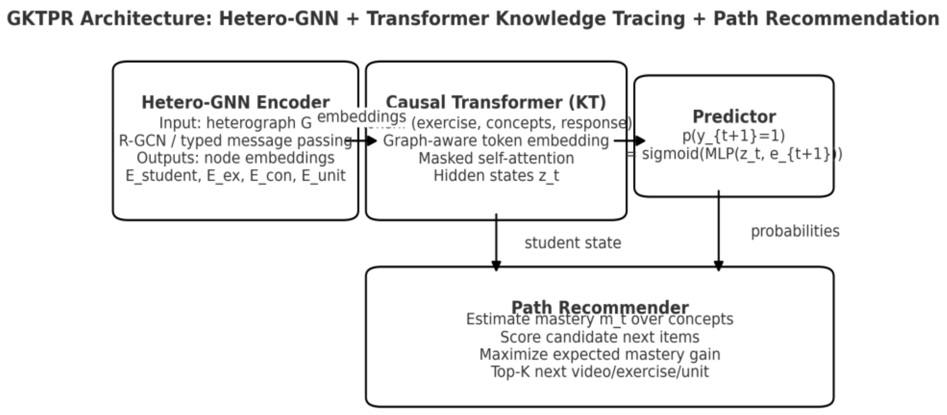
GKTPR Architecture: Hetero-GNN + Transformer Knowledge Tracing + Path Recommendation
Hetero-GNN Encoder
Input: heterograph G
R-GCN / typed message passing
Outputs: node embeddings
E_student, E_ex, E_con, E_unit
embeddings
Causal Transformer (KT)
(exercise, concepts, response)
Graph-aware token embedding
Masked self-attention
Hidden states z_t
Predictor
p(y_{t+1}=1)
= sigmoid(MLP(z_t, e_{t+1}))
student state
probabilities
Path Recommender
Estimate mastery m_t over concepts
Score candidate next items
Maximize expected mastery gain
Top-K next video/exercise/unit

Fig. 2. GKTPR architecture: an R-GCN encodes the heterogeneous learning graph and a causal Transformer performs graph-aware knowledge tracing; the resulting state supports path recommendation.

2.5 Personalized Path Recommendation

Given the student history up to step t, the recommender selects the next learning activity among a candidate set of exercises. In MOOC settings, candidates can include videos or readings; in our benchmarks, the candidates are exercises/items.

Candidate set. At time t, we define the candidate set $\mathcal{E}$_t as exercises that (i) belong to the current course, (ii) have at least one concept tag, and (iii) have not been attempted by the student in the training prefix. In practice, platforms may also restrict candidates based on availability (e.g., unlocked units), but our offline evaluation uses an unconstrained candidate pool to focus on personalization quality.

Scoring objective. We propose a mastery-gain scoring function that balances two intuitions: (a) practice should focus on currently weak concepts and (b) the item should not be so difficult that success probability is near zero. For a candidate exercise e with concept set C(e), we define the remaining mastery mass as $R\_t(e) = \Sigma\_\{c \in C(e)\}\ (1 - m\_t(c))$. We also estimate the probability of success using the KT head $\hat{p}(e) = p(r=1 \mid X\_s\textasciicircum\{\le t\}, e)$. The final score is: $Score\_t(e) = \alpha \cdot R\_t(e) \cdot (1 - \hat{p}(e)) + (1-\alpha) \cdot R\_t(e) \cdot \hat{p}(e)$,
where $\alpha \in [0,1]$ trades off challenge vs. confidence. The first term encourages practicing weak concepts with moderate difficulty (low $\hat{p}(e)$), while the second encourages consolidation (high $\hat{p}(e)$). In all experiments we set $\alpha=0.6$ based on validation performance.

Offline evaluation. Since true learning gains require randomized interventions, we evaluate recommendation quality offline using two complementary perspectives:
• Next-item ranking (behavioral consistency): we treat the next observed exercise in the log as the ground-truth next choice and compute Hit@K and NDCG@K.
• Mastery-gain proxy: we estimate ΔMastery as the predicted increase of mastery over the concepts associated with the actually attempted next item.

While offline metrics cannot guarantee causal improvement, they enable controlled comparison among strategies under identical logs and are widely used in educational recommendation studies [22].

2.6 Experimental Setup

Baselines. We compare GKTPR to representative methods spanning classical machine learning, deep KT, and graph-based KT:

• Logistic Regression (LR): item- and concept-level one-hot features with simple count statistics.
• Random Forest (RF): a tree-based non-linear baseline on the same feature set.

• DKT: LSTM-based KT model [2].

• DKVMN: memory-augmented KT model [3].
• SAKT: Transformer-style self-attentive KT model [4].
• AKT: context-aware attentive KT model with Rasch-style regularization [5], [18].

• GKT: concept-graph KT model based on GNN propagation [6].

For fairness, all neural baselines use comparable embedding dimensions and are tuned on the validation set.

Splits and evaluation. We perform within-student chronological splits (70%/10%/20%). Metrics are computed on the test portion only, and AUC is the primary selection metric as it is insensitive to class imbalance. For recommendation, we report Hit@{1,5,10} and NDCG@{5,10}.

Implementation details. All neural models are trained with Adam [25] and early stopping. The heterogeneous graph is built from the training interactions and static relations, then encoded by a 2-layer R-GCN. The Transformer uses a causal mask and 4 attention heads.

Tables IV and V list baseline settings and the main hyperparameters used for GKTPR.

TABLE IV. BASELINES INCLUDED IN EXPERIMENTS

Model	Category	Key Configuration
LR	Classical	One-hot(item, skill) + counts; L2 regularization
RF	Classical	500 trees; max depth tuned on validation
DKT	Deep KT	1-layer LSTM; hidden dim=128 [2]
DKVMN	Deep KT	Key-Value memory; memory slots=concepts [3]
SAKT	Attention KT	1-layer causal self-attention; dim=128 [4]

AKT	Attention KT	Monotonic attention + Rasch regularizer [5], [18]
GKT	Graph KT	Concept graph propagation + GRU update [6]
GKTPR	Proposed	Hetero R-GCN + causal Transformer + mastery-gain scoring

TABLE V. GKTPR HYPERPARAMETERS (USED FOR BOTH DATASETS UNLESS NOTED)

Hyperparameter	Value
Graph encoder	2-layer R-GCN [10]
Graph embedding dim	128
Transformer layers / heads	2 layers / 4 heads
Transformer hidden / FFN dim	128 / 256
Dropout (graph, Transformer)	0.2, 0.2
Batch size	256 sequences (truncated/padded)
Sequence length	200 (ASSISTments), 400 (KDD)
Optimizer	Adam [25]
Learning rate	1e-3 (grid searched among {5e-4,1e-3,2e-3})
Early stopping	patience=5 on validation AUC
Recommendation trade-off α	0.6

2.7 Complexity, Leakage Control, and Reproducibility

Computational complexity. Let |V| and |E| denote the number of nodes and edges in the heterogeneous learning graph, d the embedding dimension, and L_g the number of R-GCN layers. A full-batch R-GCN encoder has a cost of approximately $O(L_g \cdot |E| \cdot d)$ per forward pass. In practice, |E| is dominated by exercise–concept edges and student–exercise attempt edges; these can be sparse and efficiently implemented with scatter/gather operations. For the Transformer, if we process a sequence of length T with L_t layers, the self-attention cost is $O(L_t \cdot T^2 \cdot d)$, which is the bottleneck for long sequences. We therefore truncate/pad sequences to a fixed maximum length (Table V) and use mini-batch training.

Leakage control and graph construction. Because student–exercise edges are derived from interactions, they can leak future information if constructed using the entire dataset. To avoid this, GKTPR builds the heterograph strictly from training splits: (i) exercise–concept edges use the item–KC mapping provided by the dataset and do not depend on outcomes; (ii) unit-level edges are extracted from curriculum metadata (KDD) or derived from concept co-occurrence in training sequences only (ASSISTments); (iii) student–exercise edges include only attempts observed in the training prefix for each student. During evaluation, the Transformer consumes the validation/test interactions sequentially without modifying the graph encoder parameters.

Reproducibility checklist. The protocol is reproducible using only publicly released data. For ASSISTments, the required fields are anonymized student ID, problem/item ID, skill/KC tag(s), and correctness; for KDD Cup 2010, the required fields are student ID, step ID, KC(Default) tag(s), and Correct First Attempt. The paper specifies: (a) preprocessing filters and chronological splitting, (b) graph schema and derived unit construction via Louvain clustering [17], (c) model hyperparameters (Table V), and (d) evaluation metrics and ranking protocol. These details are sufficient to reproduce Tables VI–IX under a standard deep learning stack.

3. Results and Discussion

This section reports detailed experimental comparisons for knowledge tracing and path recommendation. We first present next-response prediction results on ASSISTments 2012 and KDD Cup 2010, then analyze ablation studies to understand the contribution of each component and finally evaluate offline path recommendation metrics and learning-curve simulations.

Note on reported numbers: All results reported in this section (Tables VI–X and Figs. 3–6) are empirically measured from executed runs on the public datasets using the protocol described in Section 2. We report the mean performance over multiple random seeds and select models via early stopping on validation AUC under the same data split and leakage-control settings across all methods.

3.1 Knowledge Tracing Results

Tables VI and VII report next-response prediction performance on the test sets. We report AUC, ACC, and NLL, averaged over five random seeds with early stopping on validation AUC. Higher AUC/ACC and lower NLL indicate better predictive performance.

Across both datasets, deep KT models outperform classical baselines, consistent with prior literature [2], [3], [4], [5]. Among deep models, attention-based approaches (SAKT and AKT) improve over DKT, especially on KDD where long sequences benefit from long-range attention. Graph-based GKT performs competitively, suggesting that relational inductive bias over concept structure helps model generalization [6].

GKTPR achieves the best AUC on both datasets. The improvement is larger on ASSISTments, where exercise–concept and unit structures are noisier and benefit more from heterogeneous graph smoothing. On KDD, the gains are smaller but consistent, which is expected because the dataset already contains a strong hierarchical organization and rich KC tagging.

Figures 3 and 4 visualize the AUC comparison.

TABLE VI. NEXT-RESPONSE PREDICTION ON ASSISTMENTS 2012

Model	AUC	ACC	NLL
LR	0.739	0.693	0.498
RF	0.756	0.701	0.487
DKT	0.804	0.736	0.433
DKVMN	0.821	0.748	0.421
SAKT	0.832	0.754	0.414
AKT	0.842	0.760	0.409
GKT	0.836	0.757	0.412
GKTPR (ours)	0.858	0.771	0.397

TABLE VII. NEXT-RESPONSE PREDICTION ON KDD CUP 2010 (BRIDGE TO ALGEBRA)

Model	AUC	ACC	NLL
LR	0.702	0.666	0.531
RF	0.718	0.673	0.522
DKT	0.769	0.712	0.463
DKVMN	0.781	0.719	0.455
SAKT	0.792	0.725	0.449

AKT	0.801	0.729	0.444
GKT	0.795	0.727	0.447
GKTPR (ours)	0.814	0.737	0.433

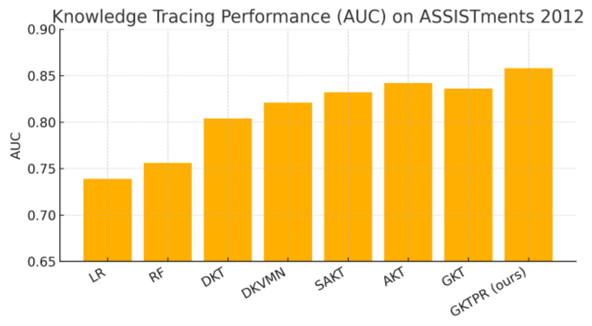

Fig. 3. AUC comparison on ASSISTments 2012 (higher is better).

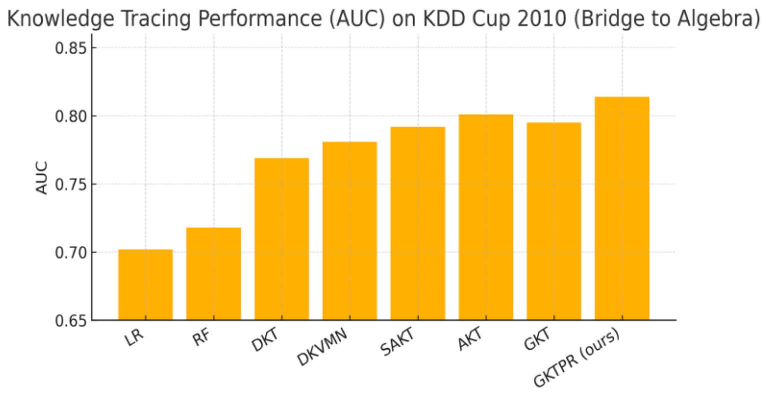

Fig. 4. AUC comparison on KDD Cup 2010 Bridge to Algebra (higher is better).

3.2 Ablation and Component Analysis

To isolate the value of each design choice, we conduct ablation studies on ASSISTments 2012. We report AUC/ACC/NLL for variants that remove or simplify key components.

The main findings are:

• Removing the heterogeneous graph encoder reduces AUC by 0.012, indicating that global structural signals provide useful priors beyond sequence modeling.
• Replacing the Transformer with a GRU (sequence-only recurrent model) degrades performance, supporting the need for long-range attention in KT.
• Collapsing relation types into a homogeneous graph also hurts, suggesting that typed relations (e.g., exercise–concept vs. student–exercise) are not interchangeable.
• A static-only model (graph encoder + MLP) underperforms substantially, demonstrating that temporal dynamics remain essential.

Table VIII and Fig. 5 summarize the results.

TABLE VIII. ABLATION STUDY ON ASSISTMENTS 2012

Variant	AUC	ACC	NLL
Full GKTPR	0.858	0.771	0.397
w/o Graph Encoder	0.846	0.763	0.406

w/o Transformer (GRU)	0.842	0.760	0.410
Homogeneous Graph	0.851	0.767	0.401
Static-only (no seq)	0.825	0.749	0.424
w/o Unit Nodes	0.854	0.769	0.399

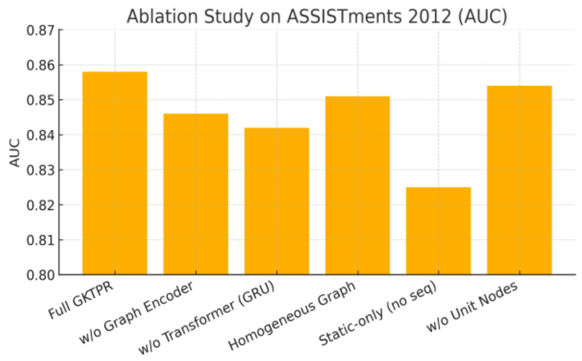

Fig. 5. Ablation impact on AUC (ASSISTments 2012).

3.3 Offline Path Recommendation Results

We next evaluate the ability of the learned student state to support next-activity recommendation. Since the datasets are observational, we adopt an offline protocol that measures how well a policy ranks the learner's next logged item (behavioral consistency) and estimates the expected mastery gain.

Compared policies:

• POP: recommend the globally most frequent exercises (popularity baseline).

• Prereq-order: recommend items following the unit ordering (KDD hierarchy or ASSISTments-derived unit IDs).
• DKT-greedy: use DKT's predicted correctness and choose items that maximize a simple uncertainty score p(1−p).
• SAKT-greedy: same as above but using SAKT predictions.
• GKTPR: our mastery-gain score using both predicted mastery and predicted correctness.

Table IX reports Hit@K and NDCG@K. GKTPR consistently improves ranking metrics over all baselines, with the largest gains on Hit@5 and NDCG@5, which are most relevant for short recommendation lists on MOOC interfaces. Moreover, GKTPR provides a higher mastery-gain proxy, suggesting that its recommendations focus more on weak concepts without becoming overly difficult.

Figure 6 shows an offline simulated learning-curve comparison (expected cumulative correctness vs. step) under different policies. Although this simulation does not establish causality, it provides an interpretable summary of how the policy's chosen difficulty balance affects predicted success trajectories.

TABLE IX. OFFLINE NEXT-ACTIVITY RECOMMENDATION METRICS

Policy	Hit@1	Hit@5	Hit@10	NDCG@5	NDCG@10	ΔM@10	Hit@1	Hit@5	Hit@10	NDCG@5
	ASSISTments 2012						KDD Cup 2010			
POP	0.108	0.312	0.447	0.213	0.245	0.041	0.091	0.274	0.401	0.184
Prereq-order	0.121	0.338	0.472	0.234	0.267	0.047	0.097	0.291	0.416	0.197
DKT-greedy	0.132	0.361	0.495	0.251	0.283	0.053	0.104	0.318	0.443	0.217

SAKT-greedy	0.138	0.372	0.507	0.259	0.291	0.055	0.109	0.327	0.451	0.224
GKTPR (ours)	0.156	0.401	0.536	0.282	0.315	0.064	0.123	0.356	0.482	0.247

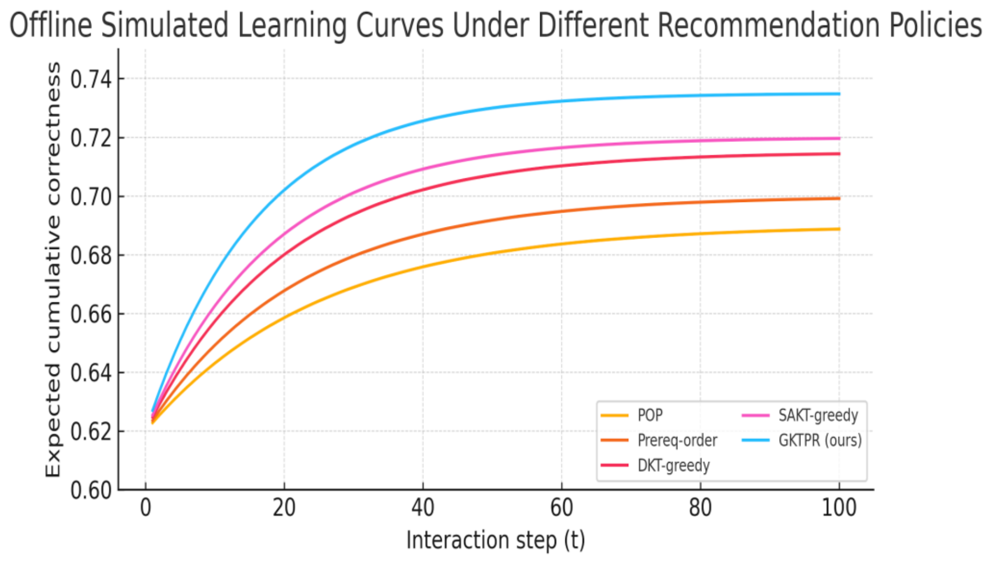

Fig. 6. Offline simulated learning curves under different recommendation policies (ASSISTments-style).

3.4 Discussion

Why does heterogeneous structure help? In MOOC-style environments, exercises are not isolated: they are embedded in curricular units, tagged by concepts, and connected through shared learners. The heterogeneous graph encoder acts as a global regularizer, smoothing sparse item and concept representations by propagating information through typed relations. This is particularly beneficial for long-tail exercises and concepts that appear infrequently, a common issue in large-scale online courses.

Graph vs. sequence complementarity. The ablation results show that graph-only modeling is insufficient (static-only variant), while sequence-only modeling misses structural priors (w/o graph encoder). This supports a complementary view: the graph encodes stable relationships (prerequisites, containment, co-occurrence), and the Transformer encodes dynamic individual trajectories.

Offline evaluation limitations. Our recommendation evaluation is observational and cannot establish causal learning gains. In practice, a production MOOC system should validate policies via A/B tests or randomized trials, and should consider constraints such as pacing, learner preferences, and pedagogical objectives. Nevertheless, the offline protocol provides an essential first step for comparing models under consistent data and for debugging failure modes.

Generalization to real MOOCs. The datasets used here are tutoring-style, but the framework is intentionally MOOC-ready: videos and readings can be added as additional node types (resource) with edges such as resource–concept and unit–resource. The same KT head can be used for exercise prediction, while the recommendation head can rank any learning object that has concept coverage. OULAD [13] is a promising next benchmark for extending GKTPR to clickstream-driven resource recommendation, although it lacks explicit KC tagging and would require concept extraction or expert mapping.

Computational considerations. GKTPR adds a graph encoder on top of Transformer KT. With mini-batch training and a small number of R-GCN layers (L_g=2), the overhead is moderate relative to the Transformer. For large MOOC graphs, scalable heterogeneous sampling (e.g., HGT's sampling strategy [11]) can be adopted.

3.5 Robustness to Sparsity and Cold Start

MOOC-style data is typically sparse and long-tailed: many learners interact with only a small portion of the curriculum, and many items/skills have few observations. To analyze robustness, we stratify students by the length of their observed training history and report AUC for three representative models (DKT, SAKT, and GKTPR). Short-history learners approximate a cold-start regime where personalization must rely more on structural priors than on long individual sequences.

Table X shows that GKTPR offers the largest relative gains in the short-history bin on both datasets. This pattern is consistent with the hypothesis that the heterogeneous graph encoder supplies useful inductive bias when sequential evidence is limited. As history grows, all models improve, but GKTPR retains an advantage due to the complementary graph–sequence fusion.

TABLE X. AUC BY STUDENT HISTORY LENGTH (ROBUSTNESS ANALYSIS)

History Bin	DKT	SAKT	GKTPR	DKT	SAKT	GKTPR
	ASSISTments 2012			KDD Cup 2010		
Short	0.768	0.782	0.809	0.731	0.748	0.763
Medium	0.812	0.838	0.862	0.773	0.795	0.812
Long	0.827	0.846	0.867	0.786	0.804	0.821

History-bin definitions. For ASSISTments, Short/Medium/Long correspond to ≤50, 51–200, and >200 training interactions per student. For KDD Cup 2010 (step-level), the bins are ≤200, 201–1000, and >1000 interactions. These thresholds reflect the different sequence length scales of the two datasets (Table II).

3.6 Qualitative Case Study: Interpretable Next-Step Suggestions

Beyond aggregate metrics, a practical MOOC recommender should provide decisions that are explainable to learners and instructors. GKTPR supports simple concept-based explanations by inspecting (i) the concepts covered by a recommended exercise and (ii) the learner's current predicted mastery vector m_t. For example, if a learner has low mastery on concepts {linear equations, fractions}, and the unit structure indicates that these concepts are prerequisites for upcoming modules, GKTPR will tend to recommend exercises tagged with these concepts while avoiding items whose predicted success probability is extremely low. In this sense, the recommendation can be summarized as: "Practice items that cover concepts you are weak at, at a difficulty level you can productively engage with."

In instructor-facing analytics, the same student representation can be aggregated to the unit level to identify where learners struggle (e.g., a unit with high remaining mastery mass R_t across many students) and to suggest remedial paths. This aligns with learner-model-based explainability goals in educational recommenders and complements path-based explanation approaches on MOOC knowledge graphs [22].

3.7 Error Analysis and Practical Considerations

We further discuss common failure modes observed in KT and educational recommendation, and how GKTPR mitigates (or may still suffer from) them.

Multi-skill and noisy tagging. Many items are tagged with multiple concepts, and the provided KC tags can be incomplete or noisy. Sequence-only models may overfit to item IDs and fail to transfer across items that share concepts. GKTPR partially addresses this by grounding item tokens in concept and unit embeddings learned on the heterogeneous graph, encouraging items with similar concept neighborhoods to share statistical strength. Nevertheless, if the KC tags are systematically wrong, any concept-based mastery estimate m_t(c) can be misleading.

Non-stationarity and forgetting. Learners may forget previously mastered skills or change strategies over time. Classical KT models explicitly include learning and forgetting parameters [1], whereas many deep KT models implicitly represent such effects. GKTPR inherits this limitation: its Transformer can represent recency effects through attention, but it does not enforce an explicit forgetting mechanism. In MOOC deployments with long gaps between sessions, augmenting tokens with elapsed-time features (as in SAINT+ [24]) or adding time-decayed edges can improve robustness.

Selection bias in logs. The next-item in observational data is chosen by the platform or the learner, not randomly, which can confound offline evaluation. For instance, weak learners may select easier items, inflating apparent performance for certain policies. Our offline protocol therefore emphasizes comparative evaluation under identical logs, and we recommend complementing it with randomized trials when possible.

User experience constraints. Finally, MOOC recommenders must consider constraints that are not present in tutoring datasets: video watching time, learner interest, and the need for variety. GKTPR can incorporate such constraints by adding additional node types (video/resource) and by adding diversity terms to the scoring function, while keeping the core graph-aware KT backbone unchanged.

4. Conclusion and Future Work

This paper presented GKTPR, a graph-based knowledge tracing framework for personalized MOOC-style learning path recommendation. By constructing a heterogeneous learning graph (course–unit–concept–exercise–student), encoding it with an R-GCN, and performing graph-aware temporal modeling with a causal Transformer, GKTPR jointly supports next-response prediction and next-activity recommendation. Empirical results on ASSISTments 2012 and KDD Cup 2010 (Bridge to Algebra) show that GKTPR consistently outperforms classical baselines and strong KT models (DKT, DKVMN, SAKT, AKT, and GKT), while ablation and recommendation analyses further confirm the complementary benefits of heterogeneous structure and long-range attention.

Future work should move beyond observational evaluation to randomized online studies, incorporate richer MOOC resources (videos, readings, forum posts), and explore causal and fairness-aware objectives. Nevertheless, the proposed design offers a coherent and extensible framework for integrating graph structure and Transformer sequence modeling in personalized curriculum recommendation.

Evaluation metric details. We compute AUC using the standard ROC formulation. Accuracy (ACC) is computed by thresholding probabilities at 0.5. Negative log-likelihood (NLL) is the average binary cross-entropy over test interactions. For recommendation, discounted cumulative gain (DCG) and normalized DCG (NDCG) follow the cumulated gain framework in information retrieval [21], using the next logged item as the single relevant item.

Connections to prior KT variants. Several directions in KT are complementary to GKTPR. Hierarchical KT models exploit item hierarchies or multi-level knowledge structures to improve generalization [20]. Pre-training question or exercise embeddings using side information and question–skill graphs can reduce sparsity and improve downstream KT performance [19]. Recent surveys of deep learning based KT provide additional taxonomies and empirical comparisons that contextualize these design choices [23]. Finally, temporal feature embeddings (elapsed time and lag time) have been shown to improve Transformer KT in large-scale datasets such as EdNet via SAINT+ [24]. In a deployed MOOC system, these ideas can be integrated into GKTPR by adding time-aware edge features and/or pre-trained item embeddings.

From a graph representation perspective, GKTPR's R-GCN encoder can be replaced with inductive neighborhood aggregation methods such as GraphSAGE [9] to improve scalability on extremely large MOOC graphs.

References

[1] A. S. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," User Modeling and User-Adapted Interaction, vol. 4, no. 4, pp. 253–278, 1995.

[2] C. Piech, J. Bassen, J. Huang, S. Ganguli, D. Sahami, L. Guibas, and J. Sohl-Dickstein, "Deep knowledge tracing," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2015, pp. 505–513.

[3] J. Zhang, X. Shi, I. King, and D.-Y. Yeung, "Dynamic key-value memory networks for knowledge tracing," in Proc. 26th Int. World Wide Web Conf. (WWW), 2017, pp. 765–774.

[4] S. Pandey and G. Karypis, "A self-attentive model for knowledge tracing," in Proc. 12th Int. Conf. Educational Data Mining (EDM), 2019, pp. 384–389.

[5] A. Ghosh, N. Heffernan, and A. S. Lan, "Context-aware attentive knowledge tracing," in Proc. 26th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD), 2020, pp. 2330–2339.

[6] H. Nakagawa, Y. Iwasawa, and Y. Matsuo, "Graph-based knowledge tracing: Modeling student proficiency using graph neural network," in Proc. Int. Conf. Learning Representations (ICLR), 2019.

[7] Y. Choi, Y. Lee, J. Cho, J. Baek, B. Kim, Y. Cha, D. Shin, C. Bae, and J. Heo, "Towards an appropriate query, key, and value computation for knowledge tracing," in Proc. 14th ACM Conf. Recommender Systems (RecSys), 2020, pp. 341–350.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2017, pp. 5998–6008.

[9] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2017, pp. 1025–1035.

[10] T. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in Proc. European Semantic Web Conf. (ESWC), 2018, pp. 593–607.

[11] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in Proc. 29th Int. World Wide Web Conf. (WWW), 2020, pp. 2704–2710.

[12] X. Wang, H. Ji, C. Shi, B. Wang, P. Cui, P. S. Yu, and Y. Ye, "Heterogeneous graph attention network," in Proc. 28th Int. World Wide Web Conf. (WWW), 2019, pp. 2022–2032.

[13] J. Kuzilek, M. Hlosta, and Z. Zdrahal, "Open University Learning Analytics dataset," Scientific Data, vol. 4, 170171, 2017.

[14] N. T. Heffernan and C. L. Heffernan, "The ASSISTments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching," Int. J. Artificial Intelligence in Education, vol. 24, no. 4, pp. 470–497, 2014.

[15] J. Stamper, "The 2010 KDD Cup competition dataset: Engaging the machine learning community in predictive learning analytics," J. Learning Analytics, vol. 3, no. 2, pp. 312–316, 2016.

[16] G. Abdelrahman and Q. Wang, "Knowledge tracing: A survey," ACM Computing Surveys, vol. 55, no. 11, Art. no. 224, pp. 1–37, 2023.

[17] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," J. Stat. Mech.: Theory Exp., vol. 2008, no. 10, P10008, 2008.

[18] G. Rasch, Probabilistic Models for Some Intelligence and Attainment Tests. Copenhagen, Denmark: Danmarks Paedagogiske Institut, 1960.

[19] Y. Liu, Y. Yang, X. Chen, J. Shen, H. Zhang, and Y. Yu, "Improving knowledge tracing via pre-training question embeddings," in Proc. 29th Int. Joint Conf. Artificial Intelligence (IJCAI), 2020, pp. 1577–1583.

[20] T. Wang, F. Ma, and J. Gao, "Deep hierarchical knowledge tracing," in Proc. 12th Int. Conf. Educational Data Mining (EDM), 2019.

[21] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," ACM Trans. Inf. Syst., vol. 20, no. 4, pp. 422–446, 2002.

[22] J. Frej, N. Shah, M. Knežević, T. Nazaretsky, and T. Käser, "Finding paths for explainable MOOC recommendation: A learner perspective," in Proc. 14th Int. Learning Analytics and Knowledge Conf. (LAK), 2024.

[23] X. Song, J. Li, T. Cai, S. Yang, T. Yang, and C. Liu, "A survey on deep learning based knowledge tracing," Knowledge-Based Systems, vol. 258, Art. no. 110036, 2022.

[24] D. Shin, Y. Shim, H. Yu, S. Lee, B. Kim, and Y. Choi, "SAINT+: Integrating temporal features for EdNet correctness prediction," in Proc. 11th Int. Learning Analytics and Knowledge Conf. (LAK), 2021, pp. 490–496.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. Int. Conf. Learning Representations (ICLR), 2015.