

Article

Design of a Lightweight Adaptive Knowledge Graph Tutoring System for Discrete Mathematics

Hao Wang ¹, Tianrui Li ^{2,*}, Handong Wang ² and Yingjie Zhang ²

¹ School of Science, Xi'an University of Architecture and Technology, Xi'an 710055, China; haowang@xauat.edu.cn

² School of Software, Henan University of Engineering, Zhengzhou 451191, China; 202212211225@haue.edu.cn (H.W.); 202212211151@stu.haue.edu.cn (Y.Z.)

* Correspondence: tianruili827@haue.edu.cn

Abstract

This paper proposes a lightweight adaptive knowledge graph tutoring system for discrete mathematics, addressing the challenges of knowledge fragmentation, homogenized pedagogical paradigms, and insufficient practical support in traditional instruction. The system formalizes disciplinary knowledge points and associated exercise repositories, constructing an integrated architecture that unifies knowledge graph representation, intelligent recommendation, adaptive assessment, and content delivery. By integrating lightweight cognitive assessment models with heuristic recommendation strategies, an adaptive learning path generation mechanism is built to dynamically quantify learner proficiency based on problem solving interaction traces. Diverging from compute-intensive deep learning paradigms, this method requires no large-scale training, making it convenient for users to adjust parameters according to actual conditions. It preserves the interpretability of learning trajectories and real-time responsiveness while enabling seamless deployment across conventional computing environments. Experiments in authentic educational settings indicate that the system facilitates learners' problem solving performance and learning efficiency, offering a scalable and cost-effective solution for personalized supplementary instruction in discrete mathematics and broader undergraduate foundational mathematics courses.

Keywords: new engineering; discrete mathematics; knowledge graph; intelligent tutoring system; personalized learning; teaching reform

1. Introduction

Under the emerging engineering education initiative, teaching reform in fundamental computer science courses faces unprecedented challenges. Discrete mathematics, serving as a cornerstone course for computer science and software engineering majors, encompasses a knowledge spectrum spanning set theory, graph theory, mathematical logic, and algebraic systems, characterized by pronounced conceptual abstraction, intricate interconnections among knowledge units, and stringent formal reasoning requirements. However, current discrete mathematics instruction is beset by knowledge fragmentation, homogenized pedagogical approaches, insufficient practical scaffolding, and unidimensional assessment mechanisms [1]. Conventional classroom paradigms fall short in accommodating individualized learning needs, instructional content delivery remains insufficiently targeted, and formative assessment mechanisms are overly simplistic, significantly hampering improvements in instructional quality and the cultivation of innovative talent.



Academic Editor: George Angelos Papadopoulos

Received: 2 July 2026

Revised: 28 July 2026

Accepted: 29 July 2026

Published: 1 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Intelligent Tutoring Systems (ITS) offer a promising technical avenue for addressing the aforementioned challenges. Zerkouk et al. [1] conducted a systematic review of the evolution and application challenges of AI-based intelligent tutoring systems, noting that the integration of artificial intelligence technologies to deliver personalized instructional support has emerged as a pivotal direction in technology-enhanced education. Al-Cheikh et al. [2] provided a thorough examination of the evolutionary trajectory and adaptive learning contributions of ITS from the perspective of educational artificial intelligence, underscoring that adaptive learning constitutes the central objective of intelligent tutoring systems. In recent years, knowledge graph technology, distinguished by its robust knowledge organization and relational capabilities, has invigorated the intelligent transformation of the education sector [3]. Knowledge graphs enable the structured interconnection of fragmented knowledge units, constructing semantically rich knowledge networks, thereby furnishing a robust knowledge foundation for personalized learning path planning, intelligent resource recommendation, and learning behavior analysis [4].

In the domain of educational knowledge graph construction, researchers have proposed a variety of methodologies tailored for the education sector. Ain et al. [5] introduced a word embedding-based approach for automatically constructing educational knowledge graphs, utilizing SqueezeBERT-enhanced SIFRank for keyphrase extraction and SBERT-based semantic similarity for concept weighting, thereby facilitating accurate concept identification and personalized learning resource recommendation. Su and Zhang [6] implemented automated subject knowledge graph construction based on educational big data, exploring viable pathways to derive knowledge relations from unstructured educational resources. Dang et al. [7] linked Wikipedia entries with course concepts to construct educational knowledge graphs, utilizing open encyclopedia resources to enrich curricular knowledge representation. Chen et al. [8] designed the KnowEdu system, focusing on automatic knowledge graph construction in the education domain and enabling extraction of educational knowledge from diverse sources such as textbooks and assessment items. Furthermore, knowledge graph research methods grounded in collaborative knowledge construction discourse [9] offer novel perspectives on educational knowledge graph construction, dynamically generating and refining knowledge associations through the analysis of learner interaction trajectories.

Knowledge Tracing (KT) stands as one of the core technologies in intelligent tutoring systems, designed to dynamically capture the evolving knowledge states of learners. Abdelrahman et al. [10] conducted a systematic survey of knowledge tracing techniques, elucidating the evolution from traditional probabilistic models to deep learning models. Driven by advances in deep learning, deep learning-based knowledge tracing models [11] leverage the robust representational capacity of neural networks, substantially improving the precision of predicting knowledge states. Subsequently, the integration of graph neural networks with knowledge tracing has emerged as a prominent research frontier. Zheng et al. [12] introduced a graph neural network-based knowledge tracing model integrating forgetting factors with memory gating mechanisms, effectively capturing the forgetting patterns and long-term retention mechanisms inherent in learning processes. Zhang et al. [13] further explored adaptive graph neural network approaches coupled with incremental learning for knowledge graph reasoning, enabling dynamic refinement and incremental evolution of knowledge embeddings. Furthermore, Pavlik and Eglington [14] demonstrated that automated search methods in logical knowledge tracing are capable of matching or exceeding the efficacy of deep learning approaches while concurrently providing enhanced interpretability, presenting innovative perspectives for refining knowledge tracing model architectures.

In the domain of intelligent recommendation, graph neural network (GNN) technology has emerged as a powerful enabler for personalized learning resource recommendation. Chetoui and El Bachari [15] proposed a graph neural network-based personalized learning recommendation approach, leveraging the intricate interconnections among knowledge units to enable precise recommendation. Zhu [4] further integrated knowledge graphs with bidirectional long short-term memory networks (BiLSTM) and conditional random fields (CRFs), achieving the intelligent formulation of adaptive learning frameworks. Mhagama and Garg [16] conducted a systematic survey of the technological evolution of educational recommender systems, target learner demographics, and emerging trends in personalized learning, noting that knowledge graph-enhanced recommendation approaches hold considerable promise for educational applications. Concurrently, privacy-preserving federated graph neural network methods [17] furnish practical solutions for secure learning resource recommendation in federated environments, enabling collaborative personalized services while safeguarding learner privacy. Li et al. [18] designed a knowledge graph-enhanced intelligent tutoring system, integrating exercise representativeness and informativeness metrics to realize refined cognitive diagnosis and optimized learning path generation.

In the domain of intelligent assessment, conventional adaptive testing techniques grounded in Item Response Theory (IRT) dynamically calibrate item difficulty according to individual ability estimates, enabling accurate diagnostic assessment. However, such approaches typically require extensive parameter calibration and computational resources. In contrast, our system adopts a lightweight dynamic assessment strategy based on the Ebbinghaus forgetting curve, which dynamically quantifies learner proficiency using intuitive scoring models without complex pre-training. In recent years, the integration of multimodal knowledge graphs with Retrieval-Augmented Generation (RAG) technologies [19] has further expanded the capability boundaries of intelligent tutoring systems, empowering such systems to handle diverse multimodal educational content spanning textual, visual, and video modalities while furnishing increasingly explainable pedagogical feedback via large language models. Concurrently, concerns regarding methodological soundness have become increasingly prominent in the field of educational data mining. Haim et al. [20] systematically analyzed the replicability concerns prevalent in educational data mining conference proceedings, underscoring the significance of open scientific practices in elevating research quality within the educational technology domain, thereby providing methodological guidance for the experimental design and data analysis of the present study.

Recent advancements in Large Language Models (LLMs) offer promising solutions to the scalability challenges of manual construction. For instance, KG-OBS [21] proposes an adaptive correction method for open knowledge graph construction, and FKQG [22] enables few-shot question generation via in-context learning. While these approaches effectively mitigate the labor costs and scalability issues, they typically rely on substantial computational resources and complex model tuning. In contrast, this paper focuses on a lightweight, dynamic assessment strategy suitable for resource-constrained educational environments where such heavy-duty models may not be feasible.

In summary, existing research has achieved notable advances in knowledge graph construction, knowledge tracing, intelligent recommendation, and adaptive assessment, yet several limitations persist. First, existing research predominantly focuses on general educational contexts, with a notable absence of specialized intelligent tutoring systems designed for discrete mathematics—a foundational course characterized by pronounced theoretical rigor and intricate conceptual interdependencies. Second, most systems architect knowledge graph construction, recommendation engines, and assessment modules as separate components, lacking deep integration and synergistic optimization. Third,

empirical validation of system effectiveness predominantly relies on simulated data, while real-world deployment and impact assessment in genuine instructional environments remain comparatively underdeveloped. In light of these limitations, this study targets the distinctive characteristics and pedagogical pain points of discrete mathematics, designing and implementing a knowledge graph-based intelligent tutoring system for discrete mathematics. The system encompasses four core modules: knowledge graph construction, intelligent recommendation engine, learning behavior analysis, and adaptive assessment, structuring interconnections among 147 knowledge points and achieving personalized recommendation by combining user problem-solving records with proficiency values.

2. Foundational Theories of Discrete Mathematics Knowledge Graphs

2.1. Foundational Theory of Knowledge Graphs

A knowledge graph is fundamentally a semantic network, designed to formally represent real-world entities and their interrelationships in a structured manner. In mathematical terms, a knowledge graph is defined as a directed graph $KG = (E, R)$, where E denotes the set of entities and R represents the set of semantic relationships between entities. Let $E = \{e_1, e_2, \dots, e_m\}$, where m denotes the total number of entities. Each entity e_i serves as a fundamental carrier of knowledge, possessing a unique identifier and a set of attributes. In the context of discrete mathematics instruction, grounded in the intrinsic logic of knowledge organization and the practical requirements of instructional application, core entity types can be categorized into the following five classes:

The knowledge point entity $e_{\text{Knowledge}}$ is formalized as the tuple $\{id, name, category, difficulty, content, related_knowledge\}$, where id denotes a unique identifier, $name$ represents the knowledge point title, $category$ specifies the affiliated classification, $difficulty$ quantifies the complexity level, $content$ encapsulates the knowledge point description in Markdown syntax, and $related_knowledge$ denotes the set of interrelated knowledge points. This entity constitutes the core node within the knowledge graph, bearing the essential pedagogical material of the discrete mathematics curriculum.

The category entity e_{Category} is defined as the tuple $\{id, name, description, sort_order\}$, where $name$ assumes one of the five disciplinary domains: set theory, logic, binary relations, graph theory, and algebraic systems, and $sort_order$ governs the presentation sequence of classifications. The category entity furnishes knowledge points with domain affiliation and structural organization, enabling learners to engage in systematic module-based learning.

The exercise entity e_{Exercise} is formalized as the tuple $\{id, topic_name, content, type, options, answer, difficulty\}$, where $type$ assumes one of the following question formats: multiple-choice, true-or-false, or fill-in-the-blank, $topic_name$ denotes the associated knowledge point, and $answer$ specifies the reference solution. This entity serves as the critical bridge connecting knowledge mastery with competency assessment, furnishing the foundational data for adaptive evaluation.

The difficulty entity $e_{\text{Difficulty}}$ is defined as the tuple $\{topic_name, difficulty_value, difficulty_level\}$, where $difficulty_value \in [0.2, 0.7]$ represents a continuous difficulty metric, and $difficulty_level \in \{\text{introductory, intermediate, advanced}\}$ denotes a discrete difficulty classification. This entity furnishes the quantitative basis for difficulty calibration within personalized recommendation, enabling adaptive alignment between problem complexity and learner proficiency.

The relation set $R = \{r_1, r_2, \dots, r_k\}$ represents semantic associations between entities, with the core relation formalized as a triple $r = (e_s, e_o, \text{label})$, where $e_s \in E$ denotes the subject entity, $e_o \in E$ denotes the object entity, and label specifies the relation type. Within the discrete mathematics knowledge graph, four categories of core semantic relations are defined: the containment relation r_{contain} signifies hierarchical subordination among

knowledge points, for instance, “graph theory” encompasses “shortest path”; the extension relation r_{extend} indicates lateral expansion of knowledge concepts, exemplified by the progression from “propositional logic” to “predicate logic”; the foundational relation $r_{\text{foundation}}$ captures prerequisite dependencies between knowledge points, such as “set operations” serving as the basis for “relational operations”; and the application relation r_{apply} characterizes practical application scenarios of knowledge concepts, as illustrated by “relations” being applied within “graph theory”. Through the formalization of these entities and relations, the discrete mathematics knowledge graph constitutes a structured semantic network, thereby establishing the foundational data infrastructure for subsequent intelligent retrieval and reasoning.

2.2. Principles of Knowledge Graph Construction

To ensure both the accuracy and efficiency of the knowledge graph within lightweight deployment environments, this study adheres to the following four core principles during the construction process.

Remark 1. *In the context of this system, the term “lightweight” refers to the characteristic that the algorithm requires no model training or complex parameter optimization, allowing it to function effectively without large-scale datasets. It is designed to be user-friendly, enabling users to expand and refine the algorithm without needing specialized knowledge in machine learning. This significantly lowers the technical barrier, allowing novice users to quickly master the system logic and perform customized deployments.*

The entity uniqueness principle mandates that each entity possess a globally unique identifier, thereby precluding ambiguity and conflict during inference processes. Formally, for any $e_i, e_j \in E$, if their core attribute sets are identical, then necessarily $e_i = e_j$. In concrete implementation, an auto-incrementing integer id is adopted as the primary key and combined with attributes such as knowledge point name and affiliated category to establish a composite unique index, thereby ensuring consistency between the database layer and the graph layer.

The relational integrity principle ensures that closed-loop associations are formed among all entity types within the knowledge graph. Formally, for any knowledge point entity $k \in E_{\text{Knowledge}}$, there must exist a category entity $c \in E_{\text{Category}}$ and a difficulty entity $d \in E_{\text{Difficulty}}$ such that both $r_{\text{belong_to}}(k, c)$ and $r_{\text{has_difficulty}}(k, d)$ hold. This principle guarantees that every knowledge point possesses complete attribute descriptions and domain affiliation, thereby precluding inference discontinuities caused by information deficiency.

The lightweight principle governs system storage and computational complexity, ensuring efficient operation within resource-constrained environments. This system employs a MySQL relational database to simultaneously store the graph structure and detailed attributes, with spatial complexity constrained to the order of $O(|E| + |R|)$. For a data scale comprising 147 knowledge point nodes and 151 relational edges, the memory footprint following graph initialization is rigorously maintained within reasonable bounds, satisfying the deployment requirements of small-to-medium-scale educational institutions.

The extensibility principle supports the dynamic evolution and incremental updating of the knowledge graph. The system adopts a modular architecture, enabling new knowledge points to be automatically integrated into the existing graph structure through standardized interfaces without necessitating the reconstruction of the entire knowledge network. Both entity types and relation types support dynamic expansion, thereby reserving interfaces for subsequent cross-curricular knowledge graph fusion.

The construction of the knowledge graph in this system follows a structured, expert-driven methodology to ensure pedagogical accuracy. Data Source: The knowledge points

(entities) were primarily extracted from the standard course syllabus of our college, ensuring alignment with the formal curriculum requirements.

Relation Annotation: Unlike automated extraction methods, the relations between knowledge points were manually annotated by domain experts (instructors). We adopted this manual strategy for two key reasons.

(1) **Pedagogical Flexibility:** It allows instructors to adjust the knowledge structure dynamically according to specific teaching needs and progress, which is often difficult with automated methods.

(2) **Lightweight Implementation:** This approach eliminates the need for training complex relation-extraction models, aligning with the "lightweight" philosophy defined in Remark 1. To ensure consistency, annotations were cross-validated by multiple experts, with conflicts resolved through discussion.

Graph Application: The knowledge graph serves primarily as a structural visualization tool to help students understand the knowledge architecture. The learning path recommendation module functions independently, relying on learner proficiency values rather than complex graph semantic reasoning. This decoupling ensures that the system can generate effective recommendations even with a relatively sparse graph structure.

2.3. Knowledge Graph and Discrete Mathematics Retrieval Association

Traditional discrete mathematics instructional resource retrieval predominantly relies on keyword matching mechanisms, inherently constrained to superficial string-level similarity comparisons that fail to capture deep semantic associations and logical dependencies among knowledge concepts. For instance, when a learner queries for graph theory applications related to propositional logic, conventional keyword-based retrieval cannot identify the indirect associative pathways between propositional logic and graph theory, frequently returning fragmented or irrelevant results.

Within the discrete mathematics instructional context, the concrete implementation of semantic retrieval encompasses two categories of core inference paradigms. Intra-category inference leverages the subordination relationships of classification entities to retrieve all knowledge points within the same category, thereby supporting systematic learning organized by domain modules. Intra-difficulty inference operates upon the attribute values of difficulty entities to achieve aggregated recommendation of knowledge points at comparable complexity levels, furnishing the evidentiary basis for personalized learning path planning. Through these semantic inference mechanisms, the knowledge graph weaves discrete mathematical knowledge points into an organic knowledge network, enabling learners to comprehend the discrete mathematics knowledge architecture from a macro-structural perspective rather than memorizing fragmented content in isolation.

3. Intelligent Recommendation Engine

Existing knowledge graph-enhanced online assessment system exercise recommendation frameworks predominantly employ static evaluation methods when constructing user-exercise matching matrices. Building upon this foundation, the present system refines the recommendation engine by introducing memory decay mechanisms and multi-dimensional feature fusion computation, thereby accommodating the dynamic characteristics inherent in learning processes.

3.1. Dynamic Competence Assessment Model

The overall architecture of the intelligent recommendation engine in this system is illustrated in Figure 1. The data input layer ingests user problem-solving records, exercise data, and knowledge point data. The feature computation layer separately calculates

dynamic proficiency values, interest degrees, and vector embedding similarities. The fusion computation layer subsequently computes difficulty adaptation degrees and forgetting enhancement factors based on these intermediate features. Finally, the scoring output layer aggregates these results to generate a Top-N recommendation list.

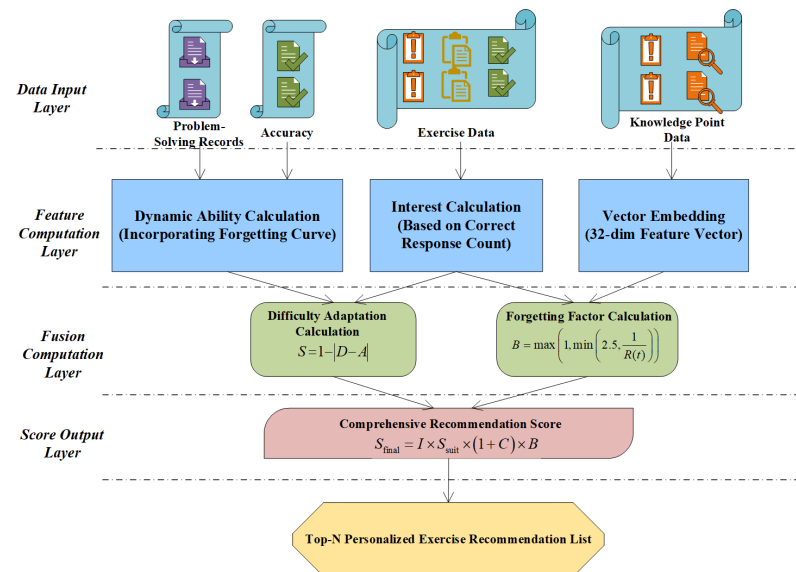


Figure 1. Personalized Recommendation Algorithm Recommendation Diagram.

In the original UEMD-GA algorithm [23] (User-Exercise Matching Degree-Generate Algorithm), user proficiency values are typically either directly provided by the online assessment system or estimated through the ELO rating algorithm based on historical records. Difficulty adaptation degrees are subsequently computed from the discrepancy between this static proficiency value and exercise difficulty. Such approaches omit user response time and memory decay factors from the computational process.

To address this limitation, the present system incorporates the Ebbinghaus forgetting curve to dynamically compute user proficiency values. Initially, a cumulative scoring model is adopted wherein correct responses accrue points while incorrect responses incur no penalty, with the per-problem score formulated as $S_u = 10 + D \times 10$ (where D denotes the problem difficulty). Subsequently, the memory retention rate is defined as $R(t) = \exp(-\frac{t}{T})$, where t represents the elapsed days since the last correct response and T denotes the configured decay period (defaulting to seven days). The user's effective score S_{eff} is derived from the product of the cumulative score and the memory retention rate. Finally, the ultimate proficiency value is computed through a nonlinear transformation normalized against the global maximum score, formulated as $P = \frac{S_{\text{eff}}}{S_{\text{max}}} \times 100$.

$$A = \left(\frac{S_{\text{eff}}}{S_{\text{max}}} \right)^{0.7} \quad (1)$$

This computational approach induces a natural temporal decay in proficiency values while constraining their numerical range within the interval $[0.05, 0.95]$. To ensure system stability and prevent the model from becoming overconfident in its assessment, the computed proficiency value A is constrained within the interval $[0.05, 0.95]$ via a clipping function. Specifically, values calculated below 0.05 are set to 0.05, and values above 0.95 are set to 0.95. This range ensures that the system maintains a minimal probability of recommending exercises for underperforming students and retains a minimal review probability for high-performing students.

The detailed computational workflow for dynamic proficiency values is illustrated in Figure 2. This workflow comprises four sequential stages: cumulative score computation, memory retention rate calculation, effective score derivation, and nonlinear transformation.

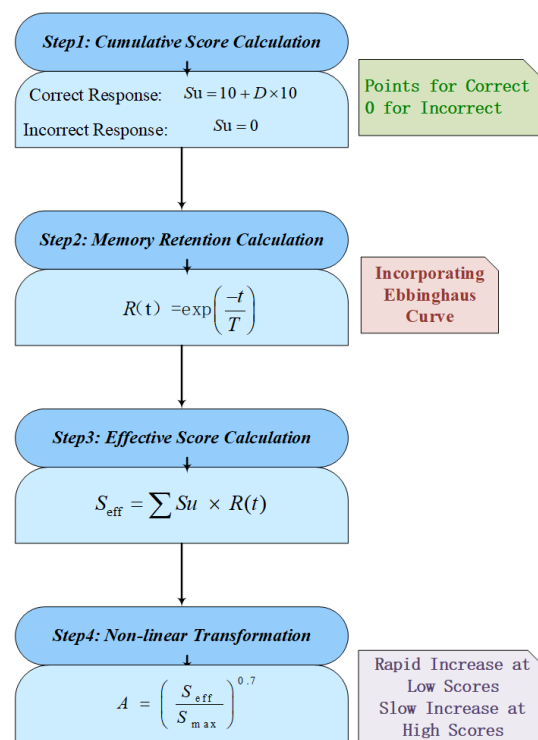


Figure 2. Dynamic Proficiency Assessment Model Workflow.

3.2. Exercise Recommendation Scoring Algorithm with Integrated Forgetting Mechanism

In conventional exercise recommendation algorithms, the matching degree between users and exercises is typically derived directly from the product of interest degree and difficulty adaptation, wherein the interest degree is predominantly computed through normalization based on knowledge point occurrence frequency. This strategy exhibits limited capability to guide review of previously learned knowledge points under scenarios such as prolonged user inactivity.

The present system restructures the exercise recommendation scoring model, partitioning the recommendation score into two components: a base score and a forgetting enhancement factor. The base score S_{base} is computed as follows:

$$S_{\text{base}} = I \times S_{\text{suit}} \times (1 + C) \quad (2)$$

In this formulation, S_{suit} denotes the difficulty adaptation degree, and C represents the cross-compression similarity between the problem feature vector and the user vector, with the system constructing 32-dimensional vectors to characterize features such as problem accuracy rate, knowledge point quantity, and difficulty level. Specifically, the user feature vector is currently composed of the proficiency value. The problem feature vector consists of three key features: error rate, knowledge point indicator (0/1), and knowledge point difficulty, with the remaining dimensions padded using knowledge point difficulty to meet the 32-dimensional requirement. The dimensionality is set to 32 to reserve space for future feature expansion. I signifies the refined interest degree, whose computation incorporates the frequency of correct responses by the user on related knowledge points, formulated as:

$$I = \sum_{i=1}^n \frac{1 - \frac{1}{1+c_i}}{n} \quad (3)$$

In this formulation, c_i denotes the count of correct responses, and n represents the number of knowledge points encompassed by the problem.

The Interest Score is designed to model learner preference. In real-world scenarios, a high frequency of practice on a specific knowledge point indicates the user's strong interest or willingness to focus on that area. By assigning higher weights to these points, the system aligns recommendations with the user's current learning intent, thereby maintaining engagement and motivation. Unlike automated estimation methods, the difficulty level of each exercise is manually set by instructors during the course construction phase. This ensures that the difficulty labels align with the pedagogical goals. The Difficulty Suitability (S_{suit}) is then calculated based on the match between this manual difficulty and the learner's current proficiency state.

Building upon this foundation, the system introduces a forgetting enhancement factor B to strengthen the review mechanism, formulated as:

$$B = \max\left(1, \min\left(2.5, \frac{1}{R(t)}\right)\right) \quad (4)$$

This factor moderately elevates the weight of exercises that have remained unanswered for extended periods, thereby facilitating review guidance while preserving the probability of new problem recommendation. The final recommendation score is computed as $S_{final} = S_{base} \times B$.

4. System Architectural Framework

4.1. System Overview

This system adopts a microservices architectural paradigm, integrating a front-end and back-end decoupled design pattern to construct a knowledge graph-based discrete mathematics learning recommendation system. The system is hierarchically organized into four tiers: the presentation layer, service layer, business logic layer, and data layer, with each tier maintaining clearly defined responsibilities and loose coupling achieved through standardized interfaces, thereby ensuring system scalability and maintainability. The overall architecture is illustrated in Figure 3.

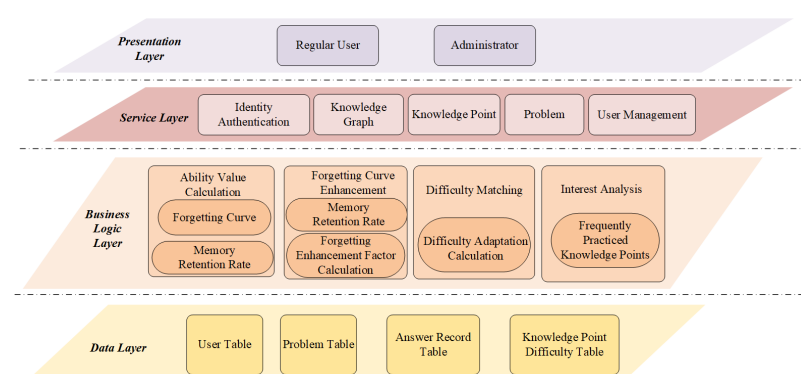


Figure 3. System Architecture Diagram.

The presentation layer is responsible for user interaction, adopting a front-end and back-end decoupled design while supporting multi-terminal access. The Web client is developed upon the Vue framework, furnishing functionalities including user registration, authentication, problem-solving, and recommendation result presentation. The administra-

tive backend provides operational interfaces for exercise management, user administration, and data analytics.

The service layer serves as the unified system entry point, implementing request routing and identity authentication through a gateway component. This tier encompasses five functional modules: authentication, knowledge graph, knowledge point, exercise, and user management. The authentication module handles user registration, login, and privilege verification. The knowledge graph module furnishes data interfaces to support knowledge point relationship queries. The knowledge point module manages concept labels and their associative relationships. The exercise module implements create, retrieve, update, and delete operations alongside difficulty management. The user module processes user information and proficiency value query requests.

The business logic layer constitutes the concrete implementation tier for core system functionalities, primarily encompassing the recommendation engine and other critical modules. The recommendation engine comprises four submodules: the proficiency calculation module, which leverages the Ebbinghaus forgetting curve to compute dynamic user proficiency values through cumulative scores and memory retention rates; the forgetting curve enhancement module, which adjusts recommendation weights according to forgetting degrees to prevent excessive knowledge decay; the difficulty matching module, which computes the adaptation degree between exercise difficulty and user proficiency; and the interest degree analysis module, which achieves personalized weight allocation by statistically analyzing user accuracy rates across individual knowledge points.

The data layer is responsible for data storage and management, with the system employing a relational database as the underlying storage medium. This tier encompasses four core data tables: the user table, which stores fundamental user information and proficiency values; the exercise table, which preserves exercise content and knowledge point associations; the response record table, which documents user answering behaviors and outcomes; and the knowledge point difficulty table, which maintains difficulty assessment data for individual knowledge points.

4.2. Core Functional Modules

The core functional modules are constructed upon the overall architecture, primarily responsible for implementing concrete business logic and user services. Each module maintains independent operation while conducting data interchange through standardized interfaces, collectively sustaining stable system execution.

The user management module handles foundational functionalities including user registration, login authentication, personal information management, and proficiency value queries. This module employs password hashing storage mechanisms to safeguard account security while supporting user information retrieval and modification. Upon completion of user problem-solving activities, this module triggers real-time proficiency value computation and aggregates user activity metrics, thereby furnishing the data foundation for subsequent personalized recommendations.

The exercise management module is responsible for maintaining the system exercise repository data, supporting create, retrieve, update, and delete operations alongside the association binding between exercises and knowledge points. Administrators can perform multi-level classification of exercises according to knowledge point labels, while dynamically managing exercise difficulty through the system-embedded difficulty assessment mechanism. Furthermore, this module supports batch import of exercises via standard format files, substantially enhancing the efficiency of repository maintenance.

The recommendation engine module serves as the core functional component of this system, primarily responsible for generating personalized exercise recommendations.

This module integrally invokes the proficiency value calculation model, the forgetting curve enhancement factor computation model, and the difficulty adaptation algorithm to perform multi-dimensional feature scoring on candidate exercises within the repository. By computing final recommendation scores and arranging them in descending order, the system can precisely deliver exercises aligned with the user's current learning state and knowledge deficiency points. The specific algorithmic principles and implementation details are presented in Section 3.

The response record module is responsible for comprehensively documenting user problem-solving behaviors within the system, furnishing underlying data support for the continuous refinement of recommendation algorithms. This module persistently stores user response time, correctness, and submission frequency in real time, while dynamically updating cumulative user scores based on answering outcomes. Concurrently, the system conducts in-depth analysis of these recorded data to generate accuracy statistical reports and track user mastery levels across individual knowledge points.

The knowledge graph module is designed to visualize the structural architecture of discrete mathematics knowledge, assisting users in constructing systematic knowledge networks. This module renders logical relationships among knowledge points through nodes and edges, supporting associative retrieval by name or difficulty level. Furthermore, the system recommends knowledge path navigation based on user learning progress, while providing interactive graph browsing functionalities to facilitate incremental task completion.

4.3. Database Design

This system employs the MySQL relational database management system, ensuring data consistency, integrity, and scalability through rational table partitioning, primary and foreign key constraint definitions, and appropriate data type specifications, while concurrently minimizing data redundancy. The database architecture is organized around core modules encompassing user information, exercise data, response records, and knowledge point difficulty, with inter-table data linkage achieved through associated fields. The principal database table designs are presented as follows.

Table 1 stores user problem-solving records, which facilitate both the recommendation of relevant exercises and the comprehensive understanding of individual learning progress.

Table 1. User exercise record table schema.

Field	Data Type	Description
id	INT	Primary key, unique record identifier
user_id	INT	User identifier, NOT NULL
question_id	INT	Exercise identifier, NOT NULL
is_correct	TINYINT(1)	Correctness flag (1/0), DEFAULT '0'
submit_count	INT	Submission count, DEFAULT '1'
time_order	INT	Problem sequence number, DEFAULT '1'
created_at	TIMESTAMP	Response timestamp, CURRENT_TIMESTAMP

Table 2 stores difficulty levels for individual knowledge points, enabling flexible adjustment of difficulty-related computations through differentiated difficulty value assignments.

Table 2. Knowledge point difficulty table schema.

Field	Data Type	Description
id	INT	Primary key, unique knowledge point identifier
topic_name	VARCHAR(255)	Knowledge point name, NOT NULL
difficulty	DECIMAL(3,2)	Knowledge point difficulty value, DEFAULT '0.50'
level	VARCHAR(20)	Knowledge point proficiency classification

Tables 3 and 4 store the nodes and edges of the knowledge graph, respectively. The separation of edges and vertices into distinct tables facilitates content addition and management within the graph structure.

Table 3. Knowledge graph node table schema.

Field	Data Type	Description
id	INT	Primary key, unique node identifier
text	VARCHAR(255)	Knowledge point name, NOT NULL
created_at	TIMESTAMP	Knowledge point creation time

Table 4. Knowledge graph edge table schema.

Field	Data Type	Description
id	INT	Primary key, unique edge identifier
from_node	VARCHAR(50)	Source node reference, NOT NULL
to_node	VARCHAR(50)	Target node reference, NOT NULL
text	VARCHAR(255)	Edge description, NOT NULL
created_at	TIMESTAMP	Edge creation time, CURRENT_TIMESTAMP

5. Data Analysis

The experimental data were derived from the problem-solving activities of students within our college. Prior to formal system usage, a privacy statement was explicitly displayed to all users, ensuring informed consent. The deployment involved approximately 500 participants and spanned a duration of about six weeks.

5.1. Learner Behavior Analysis

To investigate user learning behavior patterns, this study conducted statistical analysis of user response time periods during system operation. As shown in Figure 4, there is a pronounced temporal concentration in user answering behavior: the 14:00–18:00 interval constitutes the peak learning period, accounting for 21.7% of total responses, followed by 09:00–12:00 (16.2%) and 18:00–21:00 (13.8%), while the 21:00–24:00 interval exhibits the lowest proportion at merely 1.0%. This distribution pattern reflects a higher level of student engagement during the afternoon hours.

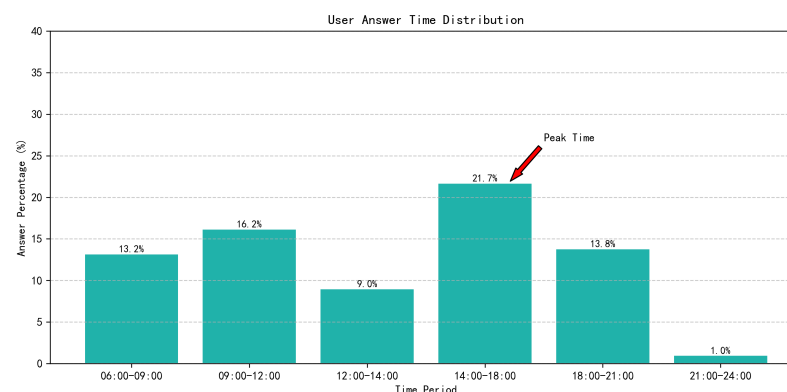


Figure 4. User Response Time Distribution.

As illustrated in Figures 5 and 6, users demonstrate higher mastery levels for fundamental concept categories, such as basic set concepts, propositional logic, and fundamental graph concepts, whereas performance on complex application categories, including planar graph coloring, mathematical logic, and graph theory, remains comparatively weaker. Accuracy rates for medium-difficulty knowledge points fall within the range of 69.2% to 69.7%.

This disparity furnishes personalized grounding for the recommendation engine, enabling the system to deliver targeted exercises addressing individual knowledge deficiencies.

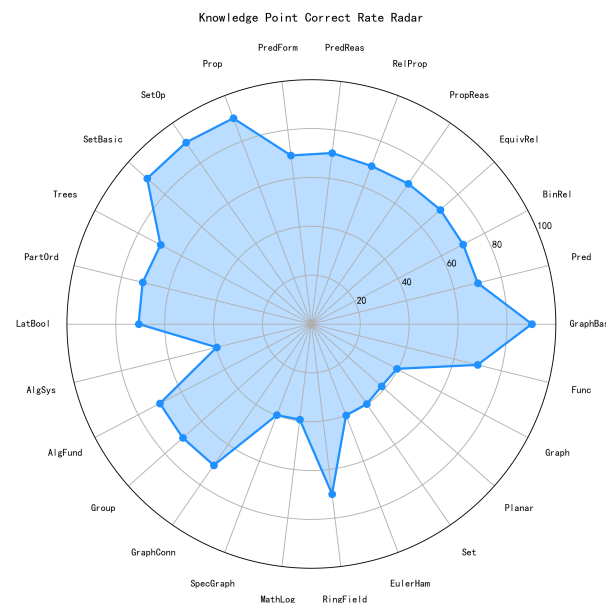


Figure 5. Knowledge Point Proficiency Radar Chart.

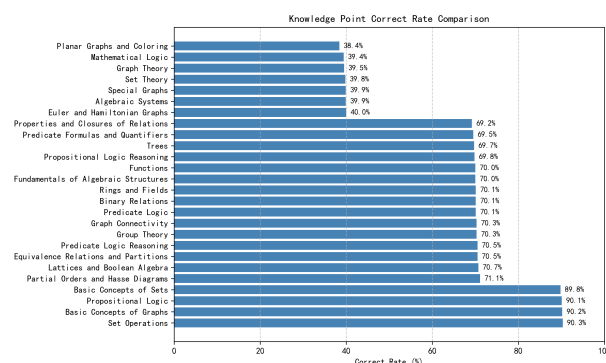


Figure 6. Knowledge Point Accuracy Comparison.

To gain deeper insights into users' genuine feedback regarding the system, we conducted a questionnaire survey. The results indicate that both learning satisfaction and academic performance improved after using the system.

5.2. Exercise Quality Analysis

As depicted in Figure 7, introductory-level exercises constitute the largest proportion within the repository, followed by intermediate-level exercises, while advanced-level exercises represent the smallest share. Figure 8 reveals that regarding exercise discrimination, excellent items account for 30.7% and good items for 30.8%, yielding a combined proportion of 61.5%. This indicates that high-discrimination items occupy a substantial proportion within the repository, effectively differentiating users across varying proficiency levels and thereby furnishing quality assurance for recommendation algorithm accuracy.

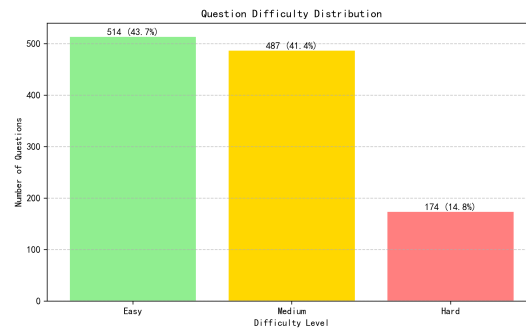


Figure 7. Problem Difficulty Distribution Chart.

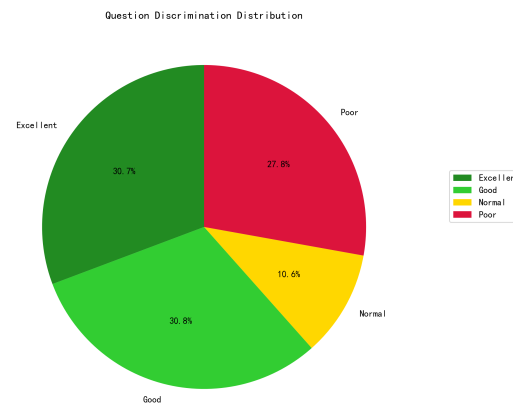


Figure 8. Item Discrimination Distribution Chart.

5.3. Algorithm Parameter Analysis

As illustrated in Figure 9, memory decay curves exhibit pronounced divergence across varying decay-period configurations. With $T = 3$ days, the memory retention rate declines rapidly to approximately 37% within seven days. At $T = 7$ days, the retention rate after seven days approximates 37%. For $T = 14$ days, the retention rate after seven days reaches approximately 60%. Balancing learning efficiency against memory preservation, $T = 7$ days is selected as the recommended value, ensuring adequate review frequency without excessively increasing learning burden.

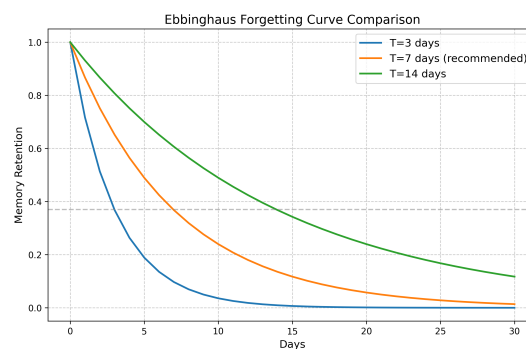


Figure 9. Comparison chart of Ebbinghaus forgetting curve.

Figure 10 illustrates the variation of forgetting enhancement factors under different upper bound configurations. When $\max_boost = 2.5$, the forgetting enhancement factor reaches its upper bound after seven days and subsequently stabilizes, thereby preserving appropriate weighting for previously encountered exercises while precluding excessive recommendation of outdated items. This parameter configuration maintains learning freshness while ensuring effective review of critical knowledge points.

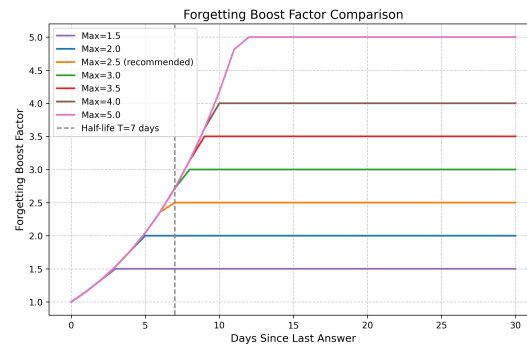


Figure 10. Upper-Bound Variation in Forgetting Enhancement Factor Comparison.

5.4. System Efficiency Analysis

To evaluate the performance of the proposed recommendation algorithm, we conducted a quantitative evaluation of its response time and concurrency performance.

Test Environment: The experiments were performed on a machine with an Intel Core i7 processor and 16GB of RAM, running Python 3.10. The test data was extracted from the real system, comprising 523 users, 1175 exercises, and 370,631 response records.

Test Methodology: To ensure objectivity and reproducibility, the following procedures were implemented:

- **Warm-up:** Three pre-runs were executed before formal testing to eliminate cold-start effects.
- **Multi-round Repetition:** Each test was repeated five times, with the arithmetic mean taken as the final result.
- **Data Preloading:** All data, including user ability values and exercise attributes, were preloaded into memory.
- **Real Algorithm Invocation:** The core function “compute_final_score()” from the production environment was directly invoked. The pseudocode for this function is shown in Function 1.
- **Recommendation Source Control:** The recommendation candidate set was limited to exercises not previously answered by the user.

Function 1: compute_final_score

```
def compute_final_score_real(self, question, user_id, user_ability):
    topic_name = question.get('topic_name', '')
    topics = [topic_name] if topic_name else []
    difficulty = self.test_data['difficulty_map'].get(topic_name, 0.5)
    I = self.compute_interest_real(topic_name, user_id)
    S = 1.0 - abs(difficulty - user_ability)
    B = I * S
    q_vec = self.get_question_embedding_real(question)
    user_vec = [user_ability] * 32
    C = self.compute_cross_compression_real(user_vec, q_vec)
    base_score = B * (1.0 + C)
    forgetting_boost = self.compute_forgetting_boost_real(user_id, topic_name)
    final_score = base_score * forgetting_boost
    return final_score
```

We evaluated the response time under different batch sizes (1, 10, 50, 100, and 500 exercises). For each batch size, 50 real users were used to measure the average and P95 response times, with results shown in Figure 11.

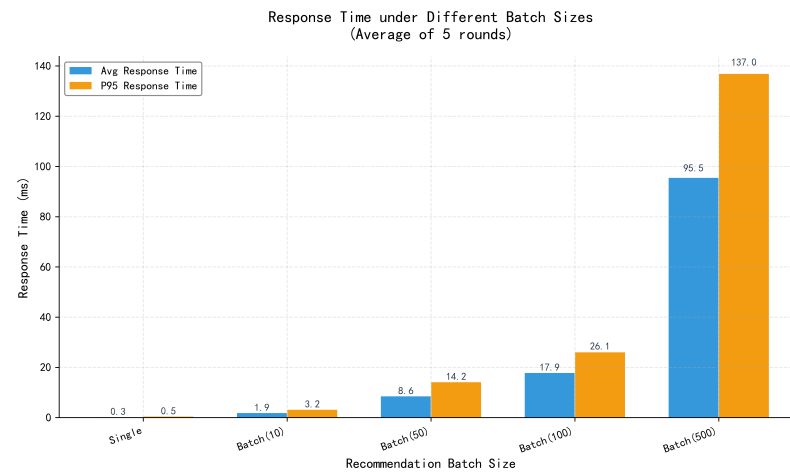


Figure 11. Response-Time Comparison under Different Batch Sizes.

The results demonstrate the algorithm's excellent response performance across all batch sizes. The average response time for a single recommendation is only 0.3 ms, with a P95 of 0.5 ms, meeting the sub-millisecond requirements of real-time systems. For a batch of 500 exercises, the average response time is 95.5 ms and the P95 is 137.0 ms, showing a near-linear growth trend. The stable ratio of P95 to average response time indicates consistent performance.

These findings confirm that the core computations of the algorithm—including interest evaluation, difficulty adaptation, and vector compression—are lightweight mathematical operations, providing inherent advantages in response performance.

To assess concurrency performance, we tested the system with 5 to 100 concurrent users (in increments of 5). For each user count, we measured per-user response time and system throughput, as shown in Figure 12.

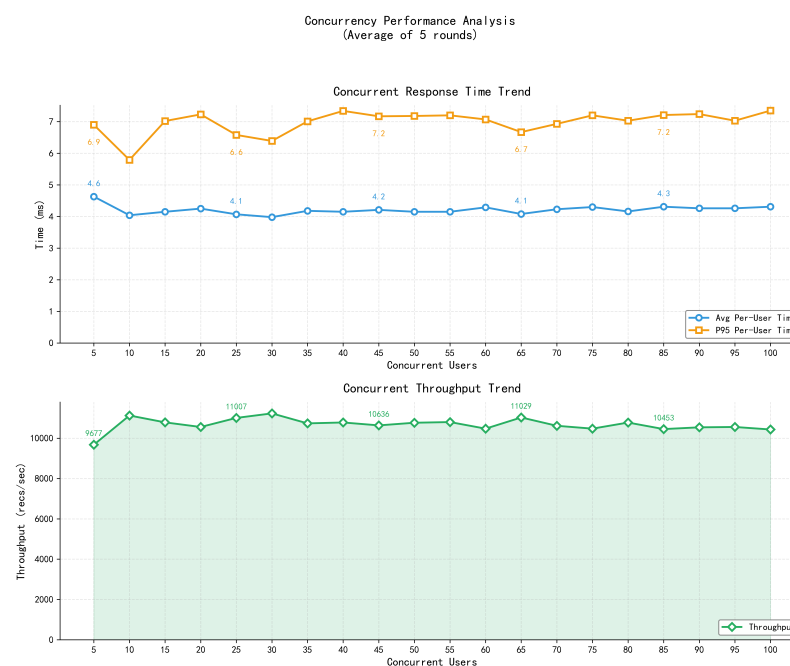


Figure 12. Response Time and Throughput Analysis under Different Concurrency Levels.

The analysis reveals stable performance under concurrency. Within the 5–100 user range, the average per-user response time remains between 4.0 ms and 4.6 ms, and the P95 time is between 6.6 ms and 7.2 ms, showing no significant degradation with increased load.

The system throughput increases from 9677 recs/s at 5 users to 10,453 recs/s at 100 users. This demonstrates that a single-instance deployment is sufficient to meet the performance demands of typical online education scenarios.

It should be noted that the test was implemented using Python multi-threading, which is subject to the Global Interpreter Lock (GIL). In a production environment, performance could be further enhanced by adopting a multi-process architecture or using a GIL-free Python implementation like PyPy.

To position the proposed recommendation algorithm within the broader landscape of educational recommendation systems, we conduct a comparative analysis with representative baseline models, focusing on key design dimensions. We primarily contrast our approach with Deep Knowledge Tracing (DKT) models and traditional Collaborative Filtering (CF) methods, as summarized in Table 5.

Table 5. Comparison of different recommendation approaches.

Dimension	Proposed Algorithm	Deep Knowledge Tracing	Collaborative Filtering
Core Mechanism	Statistical computation	Deep neural networks (RNN/LSTM)	Matrix factorization
Training Phase	Not required (Online update)	Required (Iterative backpropagation)	Required (Periodic retraining)
Data Dependency	Functions with limited data	Relies on large-scale datasets	Depends on interaction density
Deployment Expertise	General programming	ML/Hardware tuning required	Moderate algorithmic knowledge

The comparison highlights the distinct design philosophy of the proposed algorithm. In terms of model implementation, while DKT utilizes deep neural networks to capture complex learning sequences, it typically involves an offline training phase and parameter tuning. In contrast, the proposed method adopts a statistical computation approach, allowing model parameters to be updated online based on user behavior, which removes the need for an explicit training phase. Regarding deployment requirements, Collaborative Filtering methods often require matrix reconstruction when new users or items are introduced. The proposed algorithm, however, maintains a fixed computational structure independent of dataset size variations. These characteristics suggest that the algorithm offers a viable alternative for scenarios where model interpretability and ease of deployment are prioritized alongside recommendation performance.

6. Conclusions

This paper addresses the distinctive characteristics of the discrete mathematics knowledge architecture, systematically elucidating and synthesizing logical interconnections among knowledge concepts to construct a personalized recommendation mechanism grounded in user learning behaviors. The system is designed to enhance user mastery of core discrete mathematics concepts and problem-solving capabilities through intelligent exercise recommendation strategies. Leveraging historical user response data and ability values, the algorithm specifically identifies and recommends knowledge points aligned with user preferences for targeted practice. This personalized recommendation mechanism effectively addresses user-specific needs, facilitating efficient knowledge consolidation and capability enhancement. The system has been successfully deployed in discrete mathematics learning assistance scenarios. Future work will extend this technical framework to additional disciplinary domains, leveraging cross-curricular knowledge graph construction and recommendation algorithm optimization to furnish more comprehensive learning efficiency enhancement solutions.

Author Contributions: Conceptualization, H.W. (Hao Wang) and T.L.; methodology, H.W. (Hao Wang) and T.L.; validation, H.W. (Handong Wang) and Y.Z.; formal analysis, T.L. and Y.Z.; investigation, H.W. (Hao Wang) and Y.Z.; resources, H.W. (Handong Wang) and Y.Z.; data curation, H.W. (Handong Wang) and Y.Z.; writing—original draft preparation, T.L. and Y.Z.; writing—review and editing, T.L. and Y.Z.; visualization, H.W. (Hao Wang) and Y.Z.; supervision, T.L. and Y.Z.; project administration, T.L. and Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Higher Education Key Scientific Research Program of Henan Province under Grant 24A520008, as well as by the Natural Science Foundation of Henan Province under Grant 252300421507.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zerkouk, M.; Mihoubi, M.; Chikhaoui, B. A comprehensive review of AI-based intelligent tutoring systems: Applications and challenges. *arXiv* **2025**, arXiv:2507.18882.
2. Al-Cheikh, C.; Tahiri, I.; Khaldi, M. Educational artificial intelligence and intelligent tutoring systems: Foundations, evolution, and contributions to adaptive learning. *Glob. J. Eng. Technol. Adv.* **2025**, *25*, 240–246. [\[CrossRef\]](#)
3. Abu-Salih, B.; Alotaibi, S. A systematic literature review of knowledge graph construction and application in education. *Heliyon* **2024**, *10*, e25383. [\[CrossRef\]](#) [\[PubMed\]](#)
4. Zhu, Y. A knowledge graph and BiLSTM-CRF-enabled intelligent adaptive learning model and its potential application. *Alex. Eng. J.* **2024**, *91*, 305–320. [\[CrossRef\]](#)
5. Ain, Q.U.; Chatti, M.A.; Bakar, K.G.C.; Joarder, S.; Alatrash, R. Automatic construction of educational knowledge graphs: A word embedding-based approach. *Information* **2023**, *14*, 526. [\[CrossRef\]](#)
6. Su, Y.; Zhang, Y. Automatic construction of subject knowledge graph based on educational big data. In Proceedings of the 3rd International Conference on Big Data and Education (ICBDE '20), London, UK, 1–3 April 2020; pp. 30–36. [\[CrossRef\]](#)
7. Dang, F.R.; Tang, J.T.; Pang, K.Y.; Wang, T.; Li, S.S.; Li, X. Constructing an educational knowledge graph with concepts linked to Wikipedia. *J. Comput. Sci. Technol.* **2021**, *36*, 1200–1211. [\[CrossRef\]](#)
8. Chen, P.; Lu, Y.; Zheng, V.W.; Chen, X.; Yang, B. KnowEdu: A system to construct knowledge graph for education. *IEEE Access* **2018**, *6*, 31553–31563. [\[CrossRef\]](#)
9. Yang, Y.; Lai, S.N. Research on knowledge graphs based on collaborative knowledge construction discourse. *Mod. Educ. Technol.* **2025**, *35*, 97–106. [\[CrossRef\]](#)
10. Abdelrahman, G.; Wang, Q.; Nunes, B. Knowledge Tracing: A Survey. *ACM Comput. Surv.* **2023**, *55*, 224. [\[CrossRef\]](#)
11. Zhou, X.; Zhang, Z.; Xie, X.; Zhang, J. Deep learning based knowledge tracing in intelligent tutoring systems. *Sci. Rep.* **2025**, *15*, 21395. [\[CrossRef\]](#) [\[PubMed\]](#)
12. Zheng, H.; Ma, H.; Xie, Y.; Tang, W. Knowledge tracing model based on graph neural network blending with forgetting factors and memory gate. *J. Comput. Appl.* **2023**, *43*, 2747–2752. [\[CrossRef\]](#)
13. Zhang, J.; Zan, H.; Wu, S.; Zhang, K.; Huo, J. Adaptive Graph Neural Network with Incremental Learning Mechanism for Knowledge Graph Reasoning. *Electronics* **2024**, *13*, 2778. [\[CrossRef\]](#)
14. Pavlik, P.I., Jr.; Eglington, L.G. Automated search improves logistic knowledge tracing, surpassing deep learning in accuracy and explainability. *J. Educ. Data Min.* **2023**, *15*, 58–86.
15. Chetoui, I.; El Bachari, E. Personalized learning recommendations based on graph neural networks. *Int. J. Electr. Comput. Eng.* **2025**, *15*, 3246–3256. [\[CrossRef\]](#)
16. Mhagama, J.T.; Garg, K. A systematic review of educational recommender systems: Techniques, target users, and emerging trends in personalized learning. *Int. J. Technol. Educ. Sci.* **2025**, *2*, 79–98.
17. Liu, Y.; Li, H.; Hao, M. Personalized and privacy-preserving federated graph neural network. *Front. Phys.* **2024**, *12*, 1383276. [\[CrossRef\]](#)
18. Li, L.; Wang, Z. Knowledge graph enhanced intelligent tutoring system based on exercise representativeness and informativeness. *arXiv* **2023**, arXiv:2307.15076.
19. Deng, C.; Yuan, B. Research on an intelligent tutoring system based on automatic construction of multimodal knowledge graphs and retrieval-augmented generation. *Front. Comput. Sci.* **2026**, *8*, 1777749. [\[CrossRef\]](#)

20. Haim, A.; Gyurcsan, R.; Baxter, C.; Shaw, S. T.; Heffernan, N. T. How to Open Science: Debugging Reproducibility within the Educational Data Mining Conference. In Proceedings of the 16th International Conference on Educational Data Mining (EDM 2023), Bengaluru, India, 11–14 July 2023; pp. 114–124.
21. Yuan, J.; Luo, X.; Yin, Y. Optimizing knowledge bias: An adaptive correction method for open knowledge graph construction. *Neurocomputing* **2026**, *667*, 132262. [[CrossRef](#)]
22. Liu, R.; Xie, S.; Wang, X.; Luo, X.; Yu, H. FKQG: Few-shot question generation from knowledge graph via large language model in-context learning. *Data Knowl. Eng.* **2025**, *161*, 102528. [[CrossRef](#)]
23. Ye, J.; Song, J.; Zhang, K.; Li, M.; Wang, H. Research on exercise recommendation algorithm for online judge system enhanced by knowledge graph. *J. Chin. Comput. Syst.* **2023**, *44*, 2558–2565.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.