CTU Journal of Innovation and Sustainable Development
ISSN 2588-1418 | e-ISSN 2815-6412

DOI:10.22144/ctujoisd.2026.059

A distributed multi-agent framework for adaptive personalized online learning with dynamic knowledge graphs and learner modeling

Nguyen Tran Minh Khue[1,2], Nguyen Gia Nhu[2], and Dac-Nhuong Le[3*]

[1]*Academic Affairs Office, Nguyen Huu Canh College, Viet Nam*

[2]*School of Data and AI, Duy Tan University, Viet Nam*

[3]*Faculty of Information Technology, Haiphong University, Viet Nam*

**Corresponding author (nhuongld@dhhp.edu.vn)*

Article info.

Received 12 Jun 2026
Revised 8 Sep 2026
Accepted 2 Oct 2026

Keywords

Dynamic educational knowledge graph, Explainable artificial intelligence, Knowledge tracing, Multi-agent system, Personalized learning, Reinforcement learning

ABSTRACT

A distributed multi-agent framework is developed to enhance adaptive personalized online learning. The framework integrates Large Language Models (LLMs), a Dynamic Educational Knowledge Graph (DEKG), Knowledge Tracing (KT), Reinforcement Learning (RL), and Explainable Artificial Intelligence (XAI) to support learner modeling, adaptive recommendation, and explanation. The architecture separates learner-facing interaction from distributed domain processing while enabling coordinated knowledge representation and adaptive decision-making. A controlled simulation-based feasibility study evaluates learner-state estimation and personalized recommendation. The results show that graph-enhanced KT provides a modest improvement over the independent KT baseline, while the graph-aware RL policy achieves the highest final mastery and mastery gain among the evaluated strategies, although its advantage does not extend to every evaluation metric. The findings also indicate that recommendation performance is influenced by reward design. Overall, the simulation provides preliminary evidence of the feasibility of the proposed framework. Further evaluation using real educational datasets and learner interactions is required to assess its effectiveness and generalizability.

1. INTRODUCTION

Personalized online learning depends on the ability of a learning system to continuously determine what a learner knows, identify what should be learned next, and adjust instructional support as learning progresses. This goal is difficult because learner knowledge changes over time, and each instructional decision may influence subsequent learning outcomes. An effective adaptive system must therefore maintain an evolving representation of learner knowledge, use educational relationships to identify appropriate learning opportunities, and continuously refine its decisions from learner interactions. These requirements motivate an integrated and distributed approach in which learner modeling, knowledge-based reasoning, adaptive recommendation, and explanation operate together throughout the learning process.

Recent advances in artificial intelligence provide complementary capabilities for addressing this challenge. Large Language Models can support educational content understanding and feedback generation, knowledge graphs can represent conceptual and prerequisite relationships, Knowledge Tracing can estimate evolving learner mastery, Reinforcement Learning can optimize

sequential learning decisions, and Explainable Artificial Intelligence can improve the transparency of recommendations. However, integrating these capabilities remains challenging because errors or uncertainty in one component may influence subsequent decisions, while distributed execution additionally requires effective coordination and information exchange.

To address these challenges, this study develops a distributed multi-agent framework for adaptive personalized online learning. The proposed approach integrates knowledge representation, learner-state estimation, adaptive recommendation, and explainable decision-making within a unified learning process. Its distributed design supports the coordination of specialized learning functions while allowing learner information and educational knowledge to be continuously incorporated into subsequent decisions. A simulation-based feasibility study is further conducted to examine learner-state estimation and personalized recommendation, providing preliminary evidence of the operational feasibility of the proposed framework.

The main contributions are fivefold:

– A deployment-level agent specification that defines the learner-facing and domain-processing roles while allowing multiple domain-processing instances across distributed environments.

– A temporal and provenance-aware DEKG formulation connected to graph-enhanced KT and a graph-constrained personalized reinforcement-learning policy.

– An explicit message contract, confidence handling rule, fallback behavior, and analytical latency hypothesis for distributed coordination.

– A reproducibility-oriented specification that separates configurable methodological choices from quantities that require empirical calibration.

– A simulation-based feasibility evaluation that reports repeatable measurements for graph-enhanced learner-state estimation and adaptive recommendation without treating synthetic outcomes as real-learner evidence.

Section 2 reviews the related literature. Section 3 presents the distributed architecture, formal models, and operational algorithm. Section 4 reports a simulation-based feasibility study and discusses threats to validity. Finally, Section 5 concludes the study and outlines the requirements for future real-data validation.

2. RELATED WORKS

LLM-assisted knowledge-graph construction has evolved from relatively direct concept and relation extraction toward more structured knowledge acquisition, graph completion, and context-aware educational applications. Textbook2KG demonstrates how concepts and semantic relations can be extracted from lengthy educational materials to support the transformation of unstructured learning resources into structured knowledge representations (Wu et al., 2025). Broader studies on LLM-driven knowledge-graph construction further indicate that language models can support several stages of the graph-building process, including entity identification, relation extraction, semantic normalization, and graph enrichment (Bian, 2025). In educational settings, these capabilities are particularly useful because learning materials are heterogeneous and continuously updated, making purely manual knowledge engineering difficult to maintain at scale. LLMs have also been investigated for personalized learning pathways and tutoring, where learner queries and contextual information can be used to generate explanations, feedback, or learning recommendations (Liu et al., 2024). More recent graph-generation approaches have explored dynamic graph expansion and curriculum-oriented knowledge modeling (Abu-Rasheed et al., 2025; Wang et al., 2026), suggesting that educational knowledge structures can evolve as new learning resources and relationships become available. Graph-attention recommendation models have additionally shown how heterogeneous knowledge-graph structure can support personalized course recommendation (Nguyen et al., 2025). Nevertheless, LLM-generated knowledge remains vulnerable to unsupported or inconsistent outputs, and knowledge acquisition alone does not provide an explicit mechanism for estimating how learner knowledge changes over time. Consequently, LLM-assisted graph construction provides an important knowledge layer for personalization, but it does not independently address learner-state tracking, sequential instructional decisions, or long-term adaptation.

A second major research direction concerns Knowledge Tracing and the incorporation of structural information into learner modeling. Traditional KT approaches primarily estimate latent mastery from sequences of learner interactions,

whereas recent studies increasingly exploit relationships among questions, skills, and concepts to obtain richer learner-state representations. KGS-KT combines graph convolution with self-supervised learning to capture structural dependencies among educational concepts and learner interactions (Shen & Chen, 2025). DyGKT extends this perspective by representing learner-question-concept interactions through a dynamic graph, allowing temporal and relational information to contribute jointly to knowledge-state estimation (Cheng et al., 2024). Persistent-memory KT architectures have also been proposed to retain longer-term learner information across interactions (Nguyen & Nguyen, 2026). Such approaches are particularly relevant to personalized learning because mastery of one concept may depend on prerequisite knowledge or influence subsequent concepts. Graph information can therefore provide contextual evidence that is unavailable when skills are modeled independently. However, improved learner-state estimation does not itself determine which learning activity should be delivered next. A KT model can indicate what a learner is likely to know or not know, but an additional decision mechanism is required to translate this state into a sequence of pedagogically appropriate actions.

Reinforcement Learning provides a natural mechanism for this sequential decision problem because recommendation can be formulated as repeated interaction between the learning system and the learner. Instead of optimizing only the immediate relevance of an individual resource, an RL policy can select actions according to the current learner state and update its behavior from subsequent outcomes. RL-DKT links evolving learner states with sequential learning-path optimization (Fu, 2025), illustrating the potential of combining knowledge-state estimation with policy learning. Knowledge-graph-based deep RL has likewise been investigated for educational recommendation (Zhou & Wang, 2025), where graph structure can constrain or inform the selection of learning resources. Related graph-based reinforcement-learning work in recommendation further demonstrates how structured relational information can guide sequential decision-making (Magar et al., 2025). These studies suggest an important transition from static recommendation toward adaptive learning policies that consider longer-term educational outcomes. However, several challenges remain. Learner behavior is non-stationary, the observable state may only partially represent actual knowledge, and the set of appropriate learning actions can vary according to prerequisite mastery and changes in the underlying educational graph. Moreover, an optimized policy does not automatically guarantee pedagogically meaningful recommendations. This motivates tighter integration between graph-based knowledge constraints, learner-state estimation, and sequential decision-making rather than treating them as isolated components.

Knowledge graphs can contribute to this integration beyond their role as repositories of educational concepts. In a personalized learning system, graph relations can be used to identify prerequisites, retrieve relevant resources, trace dependencies among concepts, and restrict candidate learning activities according to the learner's current mastery. They can also provide explicit reasoning paths that help explain why a particular concept or remedial activity has been recommended. Previous work has demonstrated the value of knowledge graphs for supplying contextual information and interpretable reasoning paths (Rasheed et al., 2024). This capability is complementary to feature-level explainability techniques, which seek to identify the learner characteristics or interaction variables that contribute most strongly to a model decision. Feature attribution methods have accordingly been explored in educational AI to improve the interpretability of predictive and recommendation models (Shiferaw et al., 2024). Combining these two forms of explanation is potentially valuable: feature attribution can indicate which learner-state variables influenced a decision, while graph reasoning can reveal the conceptual or prerequisite path supporting the recommended activity. Nevertheless, explainability is often investigated as an additional component rather than being incorporated into the complete adaptive decision cycle.

Another related research stream concerns distributed and multi-agent architectures. Personalized learning involves several computationally different operations, including content interpretation, knowledge retrieval, learner-state estimation, recommendation, feedback generation, and explanation. Performing all of these operations within a single centralized component may create tightly coupled processing and make individual functions difficult to update or distribute. Multi-agent approaches provide an alternative in which processing responsibilities can be coordinated across autonomous or semi-

autonomous computational entities. Surveys of multi-agent deep reinforcement learning emphasize communication, coordination, distributed execution, and non-stationarity as fundamental issues in such environments (Gronauer & Diepold, 2022; Wong et al., 2023). These principles are relevant to educational systems because learner interactions generate continuously changing information that may need to be processed by different learning and knowledge services. Distributed execution may also support separation of computational responsibilities and deployment near different knowledge or data resources. However, introducing multiple agents does not automatically provide an effective architecture. Agent responsibilities, exchanged information, orchestration rules, and boundaries between agents and internal computational modules must be explicitly specified if the system is to be reproducible and experimentally evaluated.

A further limitation in the existing literature is that these research directions are frequently investigated independently or connected only partially. LLM-based educational systems may generate personalized explanations without maintaining an explicit evolving knowledge structure; graph-based systems may provide semantic relationships without learning a sequential recommendation policy; KT models may accurately estimate mastery without determining the most appropriate next learning action; and RL-based recommenders may optimize actions without sufficiently transparent reasoning. Similarly, multi-agent studies often emphasize communication and distributed intelligence but do not necessarily integrate dynamic educational knowledge, learner-state estimation, adaptive recommendation, and explainability into the same operational workflow. As a result, the principal research gap is not simply the absence of any individual AI technique, but the lack of a clearly specified mechanism through which these complementary capabilities can exchange information and jointly support continuous personalization.

The framework proposed in this study is positioned within this integration gap. The proposed architecture distinguishes deployment-level agents from specialized logical agents. The UserAgent and ProcessAgent operate at the deployment level, whereas specialized logical agents within the ProcessAgent perform particular educational and knowledge-processing functions. Knowledge representation provides prerequisite and semantic context, KT maintains an evolving representation of learner mastery, RL supports sequential recommendation, and explanation mechanisms provide interpretable evidence for selected actions. Information generated at one stage can subsequently influence later stages, forming a closed adaptive cycle from learner observation to recommendation, feedback, and state updating. This design also makes the interfaces among the major computational functions explicit, which is important for evaluating both learning behavior and distributed-system characteristics.

Importantly, the proposed integration should not itself be interpreted as evidence that a distributed framework necessarily produces better learning outcomes than centralized or component-specific alternatives. Architectural integration establishes the mechanisms through which knowledge, learner states, recommendations, and explanations can interact, whereas educational effectiveness requires empirical measurement. For this reason, the present study separates the architectural contribution from the empirical claim. The simulation-based feasibility study evaluates whether the specified learner-modeling and recommendation components generate measurable adaptive behavior under controlled conditions, while validation using real learner interaction data remains necessary to determine effectiveness and generalizability. This distinction provides a more cautious basis for evaluating the proposed framework and avoids assuming that architectural complexity alone results in improved personalized learning.

3. PROPOSED FRAMEWORK

3.1. Deployment architecture and implementation specification

The architecture contains two deployment-level agent roles. The UserAgent maintains the learner-facing conversation context, classifies incoming requests, and forwards a typed request to a ProcessAgent. A ProcessAgent coordinates domain processing and invokes internal modules for LLM-assisted knowledge processing, DEKG retrieval, graph-enhanced KT, personalized reinforcement learning, and explanation. Figure 1 illustrates this distinction. In a distributed deployment, Place 1 hosts the UserAgent and web interface, while Places 2 and 3 host ProcessAgent instances associated with different knowledge domains. “Place” therefore denotes a logical execution environment rather than a physical site or an experimentally benchmarked node.

Table 1. Proposed Agents and Modules of the framework

Agents and Modules	Multiplicity	Primary responsibility	Principal outputs
User Agent	One service; one session context per active learner	Receive/query classification, profile/preferences, route requests and return answers	Typed request, learner/session context, final response
Process Agent	One or more instances; typically one per domain execution environment	Orchestrate LLM, DEKG, KT, reinforcement-learning and XAI modules; merge evidence and manage fallbacks	Selected action, explanation, confidence, audit trace
LLM module	Internal to ProcessAgent	Extract/normalize concepts; interpret natural-language request; generate grounded content	Candidate concepts/relations, normalized query, generated artefact
DEKG module	Shared/domain-local service	Store concepts, prerequisites, resources, confidence and provenance	Subgraph, candidate constraints, reasoning paths
KT module	Internal learning service	Estimate mastery and uncertainty from interaction history and graph information	Mastery vector, hidden state, uncertainty
RL policy module	Internal decision service	Rank feasible pedagogical actions under prerequisite and pedagogical constraints	Action values, selected action, policy confidence
XAI module	Internal explanation service	Explain selected action using feature attribution and DEKG paths	Feature attribution, graph path, explanation record

Table 1 summarizes the deployment-level agents and principal computational modules, whereas the seven logical agents coordinated within the ProcessAgent are described below.

For reproducibility, every inter-agent message is specified by the tuple <request_id, message_type, sender, receiver, learner_alias, concept_ids, payload, confidence, provenance, timestamp>. Raw personal identifiers need not be exchanged across domain environments. Extraction messages should use schema-constrained structured output; deterministic or near-deterministic decoding is recommended for knowledge extraction, whereas a low-temperature configuration can be used for learner-facing generation. Any empirical implementation must report the exact model name/version, decoding settings, prompt template, maximum context, retry policy, and API date because these settings can materially affect results.

The intended software stack remains OpenAI Agents SDK for orchestration, PyTorch for learning components, the OpenAI API/GPT-4 family for language processing, and SHAP for feature attribution. These names specify the reference stack rather than proof that all modules have already been integrated and benchmarked. ASSISTments 2012 is designated as the planned benchmark for learner-modeling and recommendation experiments, not as a dataset already used to produce results in this paper. Place 1 hosts the learner-facing Web interface and UserAgent, whereas Places 2 and 3 illustrate separate logical execution environments hosting ProcessAgent instances associated with different knowledge domains. The numbering of these Places is illustrative and does not restrict the architecture to exactly three execution environments.

At the deployment level, the proposed architecture consists of two principal agents, namely the UserAgent and the ProcessAgent. The UserAgent handles learner-facing interactions and communicates learner requests and contextual information, whereas the ProcessAgent coordinates the domain-processing functions required for adaptive personalized learning. Within the proposed architecture, the ProcessAgent internally coordinates seven specialized logical agents, each responsible for a distinct function in the personalized learning process. These logical agents are not treated as independent deployment-level agents; rather, they represent specialized functional entities operating within the ProcessAgent.

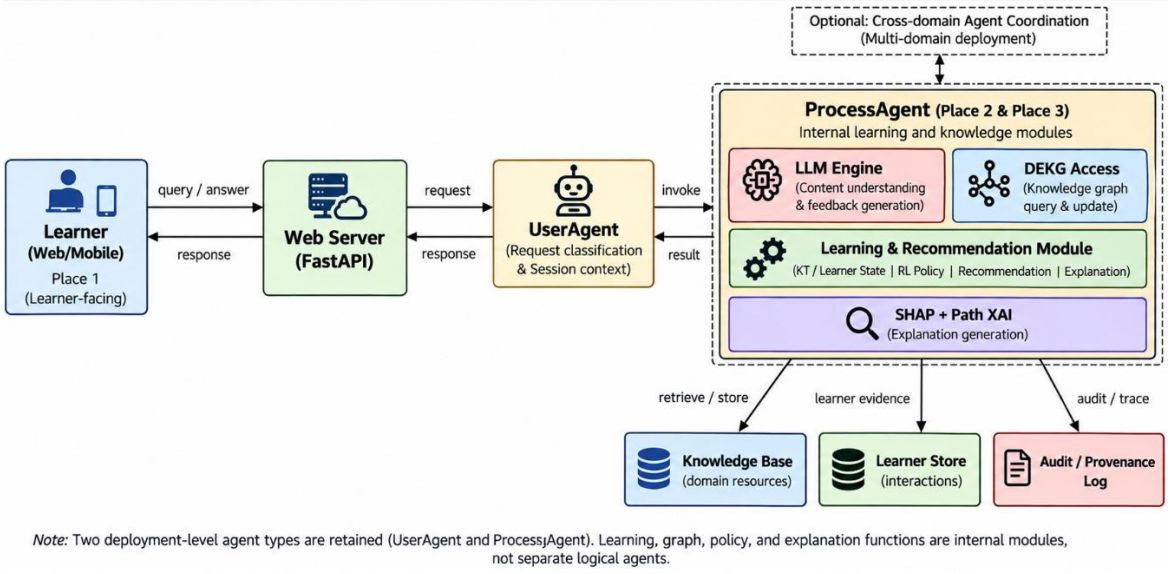


Figure 1. Deployment-level distributed architecture and internal learning modules

The Knowledge Acquisition Agent uses schema-constrained LLM prompts to extract concepts, semantic relations, learning objectives, and resource metadata from educational documents. Its outputs are represented as candidate triples accompanied by source spans, confidence values, and model-version information to support subsequent verification and traceability. The Graph Validation Agent evaluates these candidates through entity resolution, duplicate detection, cycle detection, and consistency checking, while uncertain cases can be escalated for human review. The validated results include accepted or rejected triples together with provenance information, validity intervals, and quality scores. The Learner Modeling Agent continuously updates graph-enhanced Knowledge Tracing estimates using assessment results, learner behavior, and temporal context. It produces an evolving mastery vector as well as uncertainty estimates and hypotheses concerning misconceptions and knowledge gaps.

Based on the estimated learner state, the Pedagogical Policy Agent constructs candidate actions that satisfy prerequisite constraints and applies the RL policy to determine an appropriate next learning activity. Its outputs include ranked candidate actions, Q-values, policy confidence, and expected learning utility. The Tutor and Content Agent retrieves, adapts, or generates appropriate explanations, hints, examples, exercises, and feedback while maintaining alignment with the relevant concepts and difficulty levels. The Assessment Agent evaluates learner responses, estimates item difficulty, and transforms interaction evidence into observations and reward components. Its outputs include learning outcomes, completion status, engagement, interaction time, and satisfaction measures, which can then inform learner-state and policy updates. Finally, the Explanation and Safety Agent supports transparent and reliable decision-making by generating SHAP-based feature explanations and DEKG reasoning paths, checking policy constraints, and identifying low-confidence decisions. Its outputs include feature attributions, reasoning paths, counterfactual conditions, and safety decisions. Together, these seven logical agents operate within the ProcessAgent as a coordinated functional layer, while the ProcessAgent remains the deployment-level entity responsible for orchestration and communication with other components of the distributed architecture.

3.2. Formal problem definition and notation

Let U be the set of learners, C the set of educational concepts, L the set of learning resources and activities, and R the set of semantic relation types. For learner u at decision time t, the system observes interaction evidence o_t, maintains a mastery probability $m_{t,C} \in [0,1]$ for each concept $c \in C$, and selects a pedagogical action a_t from a dynamically constrained candidate set. The learner's target is represented by a goal vector g_u

over concepts, while the profile p_u contains relatively stable attributes such as preferences, accessibility requirements, and prior educational background. The DEKG is a temporal, attributed, multi-relational graph in which vertices represent concepts, resources, assessments, learning objectives, and optionally learners; edges represent prerequisite, part-of, similar-to, assesses, covers, and misconception relations. Each edge carries a confidence score, provenance, creation time, and validity interval.

$$G_t = (V_t, E_t, R, X_t) \quad (1)$$

$$E_t \subseteq V_t \times \mathcal{R} \times V_t \quad (2)$$

where G_t denotes the Dynamic Educational Knowledge Graph (DEKG) at time t; V_t and E_t denote the sets of vertices and edges at time t, respectively; R denotes the set of semantic relation types; and X_t denotes the node and edge attributes, including confidence, provenance, creation time, and validity information.

Personalization is formulated as a constrained long-horizon optimization problem. The policy must maximize expected educational utility while satisfying prerequisite, safety, privacy, and workload constraints.

$$\pi^* = arg\ max\ over\ \pi\ E_\pi[\sum_{t=0}^{T-1} \gamma^t r_t] \quad (3)$$

subject to $a_t \in A_t^{safe}$ and $P(dropout \mid \pi) \leq \delta$; where δ denotes the maximum acceptable dropout-risk threshold. In which, $G_t, V_t, E_t, \mathcal{R}, X_t$ are temporal DEKG, vertices, edges, relation types, and node/edge attributes; m_t is the learner's mastery probabilities over concepts; o_t, h_t: Observed interaction at time t and interaction history/hidden state; s_t or b_t:

Observable state approximation or belief over the latent learner state; a_t, A_t^{safe}: Selected pedagogical action and graph-/safety-constrained candidate set; r_t, γ: Multi-objective reward and discount factor; $Q(s,a), \pi(a|s)$: Action-value function and pedagogical policy; $c(e), prov(e)$: Confidence and provenance attached to graph edge e.

3.3. Distributed knowledge bases, temporal updates, and provenance

The distributed knowledge base is partitioned across K repositories containing learning materials, assessments, interaction logs, and contextual metadata. Logical distribution permits data locality and independent administration, but a global catalogue maintains identifiers, schemas, access policies, and provenance links. Each proposed graph edge e is supported by one or more evidence items. If $c_j(e)$ is the confidence assigned by evidence source j and ω_j is its reliability weight, a simple normalized fusion rule is:

$$c(e) = \frac{\Sigma_j\, \omega_j\, c_j(e)}{\Sigma_j\, \omega_j}, \quad 0 \leq c(e) \leq 1 \quad (4)$$

Source reliability weights can be calibrated on expert-validated candidate relations. In a future empirical implementation, the reliability weight for source $j$can be estimated as: $\omega_j = N_j^{\text{valid}} / N_j^{\text{evaluated}}$; where N_j^{valid} is the number of relations from source confirmed as valid by domain experts, and $N_j^{\text{evaluated}}$ is the total number of relations from that source evaluated during calibration. The present simulation does not perform this expert-calibration procedure; therefore, $\omega_j$is treated as a configurable design parameter. An edge is admitted automatically only when c(e) exceeds a validation threshold, and no hard schema or prerequisite constraint is violated. Borderline cases are queued for human review. Dynamic updates are versioned as additions ΔE_t^+, retractions $\Delta E_t^-$and attribute changes; therefore, historical recommendations remain reproducible even after the graph evolves. The graph update is $E_{t+1} = (E_t \setminus \Delta E_t^-) \cup \Delta E_t^+$, with every change linked to its source document, extraction model, validator, and timestamp.

3.4. Knowledge graph embedding and graph-enhanced knowledge tracing

The LLM-based extraction process is constrained by an ontology schema rather than permitted to generate arbitrary triples. Given a source segment x, the Knowledge Acquisition Agent produces a structured set of candidates (h, r, t), evidence spans, confidence, and normalization proposals. Entity resolution maps synonymous mentions to canonical concept identifiers; the Graph Validation Agent then checks domain/range constraints, duplicate relations, prerequisite cycles, contradictions, and source consistency. Retrieval of relevant definitions and existing graph neighborhoods before extraction reduces hallucination and improves relation disambiguation.

The LLM may also generate learning content, but generation is conditioned on a validated concept set, target difficulty, pedagogical strategy, and cited source subgraph. Generated items pass answerability, concept-alignment, toxicity, and leakage checks before delivery. Hence, LLM output is treated as a probabilistic proposal whose

acceptance is governed by graph and safety constraints, not as authoritative curriculum knowledge.

Knowledge Graph Embedding maps every entity v and relation r to low-dimensional vectors e_v and e_r. Two compatible scoring functions are *TransE* (Bordes et al., 2013) and *RotatE* (Sun et al., 2019):

$$f_{TransE}(h,r,t) = ||e_h + e_r - e_t||_p \quad (5)$$
$$f_{RotatE}(h,r,t) = ||e_h \odot e_r - e_t||_2 \quad (6)$$

Lower scores indicate more plausible triples. With positive triples E^+ and corrupted negative triples E^-, a negative-sampling objective can be written as:

$$L_{KGE} = -\Sigma_{(h,r,t)\in E^+} log\ \sigma(\gamma_{KGE} - f(h,r,t)) - \Sigma_{(h\prime,r,t\prime)\in E^-} log\ \sigma(f(h',r,t') - \gamma_{KGE}) \quad (7)$$

where p specifies the norm order, $\odot$ denotes element-wise multiplication in the complex space, $\sigma(\cdot)$ is the sigmoid function, and γ_{KGE} is the margin parameter used in the embedding loss. A graph-enhanced KT model combines the embedding of the practiced concept with messages from its prerequisite and semantic neighborhood. One generic relational aggregation is:

$$z_{c,t} = \sigma(W_0 e_c + \Sigma_{(j,r,c)\in N(c)} \alpha_{jrc} W_r\ e_j) \quad (8)$$

where $W_0$and $W_r$are learnable transformation matrices for the target concept and relation type r, respectively; $\alpha_{jrc}$is the normalized importance weight of neighboring concept j, and $N_c$denotes the relational neighborhood of concept c. The interaction encoder then updates a hidden learner representation and converts it into mastery probabilities:

$$h_t = F_\theta(h_{t-1}, [z_{c,t}; e_{q_t}; y_t; \Delta t_t]) \quad (9)$$
$$m_t = \sigma(W_m h_t + b_m) \quad (10)$$

Here q_t is the item/resource, c_t is its concept, y_t is the observed outcome, and Δt_t captures elapsed time or forgetting. F_θ may be a Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU), Transformer, or continuous-time graph model; the recurrent formulation is consistent with Deep Knowledge Tracing (Piech et al., 2015). W_m and $b_m$denote the learnable projection matrix and bias used to map the hidden learner representation to concept-level mastery probabilities. The model is trained to predict the next observed outcome with a regularized cross-entropy loss:

$$L_{KT} = -\Sigma_t[y_{t+1} log\ \hat{y}_{t+1} + (1 - y_{t+1}) log(1 - \hat{y}_{t+1})] + \lambda||\theta||_2^2 \quad (11)$$

where $\hat{y}_{t+1}$is the predicted probability of the next observed outcome, and $\lambda$is the regularization coefficient. Because mastery is latent, calibration is as important as predictive accuracy. The Learner Modeling Agent should report both m_t and uncertainty, enabling the policy to request a diagnostic assessment when uncertainty is too high. A practical learner state is $s_t = [m_t; h_t; p_u; g_u; engagement_t; uncertainty_t]$.

3.5. Personalized reinforcement learning

Personalized learning is intrinsically partially observable because the learner's true knowledge state cannot be measured directly. The decision problem is therefore formulated as a Partially Observable Markov Decision Process (POMDP) (Kaelbling et al., 1998), where denotes the learner-state space, the action space, the observation space, the learner-state transition model, the observation model, the reward function, and the discount factor.

Since the true learner state is latent, KT-derived learner-state estimates are used to approximate the belief state from observed learner interactions:

$$b_t(s) = P(s_t = s \mid o_{1:t}, a_{1:t-1}),\ \hat{s}_t = E_{b_t}[s] \quad (12)$$

The DEKG and pedagogical constraints mask infeasible actions before RL scoring:

$$A_t^{safe} = \{a \in A : m_t, c \geq \theta_c \forall c \in Pre(a), d(a) \in [d_t^-, d_t^+], Risk(a) \leq \rho\} \quad (13)$$

The safe action set $A_t^{safe}$is defined as the subset of feasible pedagogical actions that satisfies prerequisite, difficulty, privacy, workload, and safety constraints. Here, $Pre(a)$denotes the prerequisite concept set of action a, $d(a)$is its difficulty level, and $[d_t^-, d_t^+]$is the admissible difficulty interval for the current learner state. The prerequisite threshold θ_c prevents premature advancement, the difficulty interval approximates the learner's zone of proximal development, and ρ bounds pedagogical or safety risk. Candidate masking is crucial: without it, exploratory RL may recommend conceptually invalid or excessively difficult actions.

The immediate reward is a normalized multi-objective signal:

$$r_t = \lambda_g \Delta M_t + \lambda_e Eng_t + \lambda_c C_t + \lambda_S S_t - \lambda_\tau T_t - \lambda_o O_t \quad (14)$$

ΔM_t denotes mastery gain, Eng_t engagement, C_t completion, S_t satisfaction, T_t time cost, and O_t cognitive overload or dropout risk. Because the

relative weights directly determine the trade-offs among learning effectiveness, engagement, learner satisfaction, efficiency, and cognitive risk, they are treated as tunable hyperparameters rather than fixed heuristic constants. The six reward components are first normalized to a common scale to prevent differences in magnitude from dominating the reward function. The weight vector $\boldsymbol{\lambda} = (\lambda_g, \lambda_e, \lambda_c, \lambda_S, \lambda_\tau, \lambda_o)$. Candidate weight combinations are evaluated according to the resulting cumulative reward and learning effectiveness, with primary emphasis on mastery improvement while maintaining acceptable engagement, completion, satisfaction, time cost, and cognitive overload. For real-data experiments, the reward weights should be selected on a validation set using grid search or another predefined optimization procedure. In the present synthetic feasibility study, fixed coefficients are used to examine policy behavior; these coefficients are not claimed to be empirically optimal. Such a procedure would provide an explicit and reproducible mechanism for selecting reward weights in future real-data experiments and reduce dependence on arbitrary manual assignment. For a finite discretized state-action space, Q-learning (Watkins & Dayan, 1992) updates the action value by:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t[r_t + \gamma \max_{a \in A_{t+1}^{safe}} Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (15)$$

where $\alpha_t$is the learning rate at time t. The greedy personalized policy is $\pi(s_t) = arg\,max_{a \in A_t^{safe}} Q(s_t, a)$, with masked ε-greedy exploration during training. Because the proposed state contains continuous mastery and embedding vectors, tabular Q-learning is practical only after coarse discretization. A more scalable implementation uses a deep Q-network $Q_\theta(s, a)$ (Mnih et al., 2015) and a target network θ^-:

$$L_Q(\theta) = E[(r_t + \gamma \max_{a' \in A_{t+1}^{safe}} Q_{\theta^-}(s_{t+1}, a') - Q_\theta(s_t, a_t))^2] \quad (16)$$

where $Q_\theta$denotes the online Q-network with parameters θ, while Q_{θ^-}denotes the target Q-network with parameters θ^-.

Offline pretraining on logged trajectories, conservative policy constraints and limited online exploration is recommended because unconstrained exploration with real learners is ethically inappropriate. Importance weighting or doubly robust evaluation can be used to estimate a new policy from historical data before deployment.

3.6. Distributed agent coordination and latency hypothesis

Let the agent set be $A_g = \{A_1, \ldots, A_N\}$. Each agent A_i is characterized by its observation space O_i, local knowledge K_i, decision procedure π_i, and message interface M_i. For a candidate pedagogical action named a, an agent may return a local utility $q_i(a)$, confidence c_i, evidence, expected latency, and risk estimate. The "Process Agent", acting as the orchestrator, aggregates compatible proposals rather than blindly concatenating natural-language outputs:

$$Score(a) = \Sigma_i \omega_i c_i q_i(a) - \lambda_{risk} Risk(a) - \lambda_{cost} Cost(a) \quad (17)$$

where $q_i(a)$ and c_i denote the local utility and confidence assigned by agent i to candidate action a, respectively; λ_{risk} and λ_{cost} control the penalties for pedagogical risk and execution cost, respectively. The reliability weight ω_i may be estimated from validation performance or assigned by domain experts in a future empirical deployment; in the present study, it remains a configurable coordination parameter. Hard constraints—such as a missing prerequisite or a prohibited content category—override the aggregate score. When two agents disagree, the orchestrator can request additional evidence, invoke a validator, or select a conservative fallback. Thus, coordination is evidence-aware and risk-sensitive.

If n independent tasks execute in parallel, the critical-path latency is approximately:

$$T_{parallel} \approx T_{orch} + max_i T_i + T_{merge} \quad (18)$$

whereas

$$T_{seq} \approx T_{orch} + \Sigma_i T_i + T_{merge}. \quad (19)$$

where $T_{orch}$is the orchestration overhead, $T_i$is the execution time of the agent i, and $T_{merge}$is the time required to aggregate agent outputs. Equations (18) and (19) express the potential latency benefit of parallel agent execution, but actual speedup depends on communication overhead, contention, and task independence. Distributed execution is therefore an architectural hypothesis, not a verified performance claim. Future experiments should compare centralized and distributed configurations using throughput, p50/p95 latency, communication volume, resource utilization, and speedup under increasing concurrent learners. The framework remains a cooperative MAS unless multiple agents independently learn interacting policies, which would require MARL-specific credit assignment or centralized training with decentralized execution.

3.7. Explainable decision-making

For a function approximator Q_θ, or a local surrogate $\hat{Q}$ when tabular Q-learning is used, the SHAP value of feature j (Lundberg & Lee, 2017) is:

$$\varphi_j = \sum_{S \subseteq F \setminus \{j\}} \frac{|S|!(M-|S|-1)!}{M!} [\widehat{Q}(S \cup \{j\}, a) - \widehat{Q}(S, a)] \quad (20)$$

where F is the complete feature set, $S \subseteq F \setminus \{j\}$ is a feature subset excluding feature j, and $M = | F |$is the total number of features. The top positive and negative φ_j values identify the mastery deficits, engagement signals, time constraints, or preferences that increased or decreased the selected action's value.

For graph reasoning, let $P(c_{gap} \to a)$ be the set of admissible DEKG paths from a deficient concept to the concepts or resource associated with action a. A confidence-weighted concise path is selected by:

$$p* = argmax_{p \in P(cgap \to a)} [\Sigma_{e \in p} logc(e) - \eta |p|] \quad (21)$$

where $\eta > 0$ is the path-length penalty coefficient controlling the trade-off between path confidence and explanation conciseness. The path penalty discourages long explanations, while edge confidence and provenance preserve evidential quality.

SHAP should not be interpreted as a causal explanation; rather, it provides feature attribution with respect to the learned action-value function and the selected background distribution. Accordingly, attribution results should be interpreted together with model uncertainty and the supporting DEKG evidence rather than as independent causal claims.

The two explanation channels can also be adapted to different stakeholders. A learner-facing explanation can emphasize the immediate prerequisite, the principal learner-state factors, and the expected educational benefit of the recommended action. A teacher-facing explanation can additionally expose the complete reasoning path, uncertainty estimates, alternative candidate actions, and counterfactual conditions such as the mastery threshold at which the recommendation would change. Explanation quality should therefore be evaluated in terms of fidelity, stability, sparsity, comprehensibility, and human usefulness, complementing predictive and policy-performance metrics.

3.8. Adaptive recommendation operational algorithm

The operational cycle is illustrated in Figure 2. Knowledge-graph maintenance can run asynchronously, whereas learner-state update and action selection occur for each interaction.

3.9. Reproducibility requirements for real-data extension

– Dataset: exact ASSISTments 2012 release, preprocessing rules, removed records, concept mapping, sequence construction, and learner-level split.

– KT: architecture, embedding size, hidden size, optimizer, learning rate, batch size, stopping criterion, random seeds, and calibration procedure.

– RL policy: state representation, action catalogue, feasibility thresholds, reward weights, discount factor, exploration schedule, replay settings, and target-network update rule when applicable.

– LLM: model/version, prompt schemas, decoding parameters, maximum context, retry policy, and date of API access.

– DEKG: ontology/schema, source documents, entity resolution, edge-validation threshold, expert-calibration protocol, provenance fields, and graph version.

– System: number of ProcessAgent instances, hardware, network setting, concurrent request levels, throughput, p50/p95 latency, communication volume, and failure scenarios.

Additional system experiments can compare one domain-processing instance with multiple distributed instances under equivalent workloads. These comparisons focus on learner-state estimation, personalized decision-making, and distributed coordination.

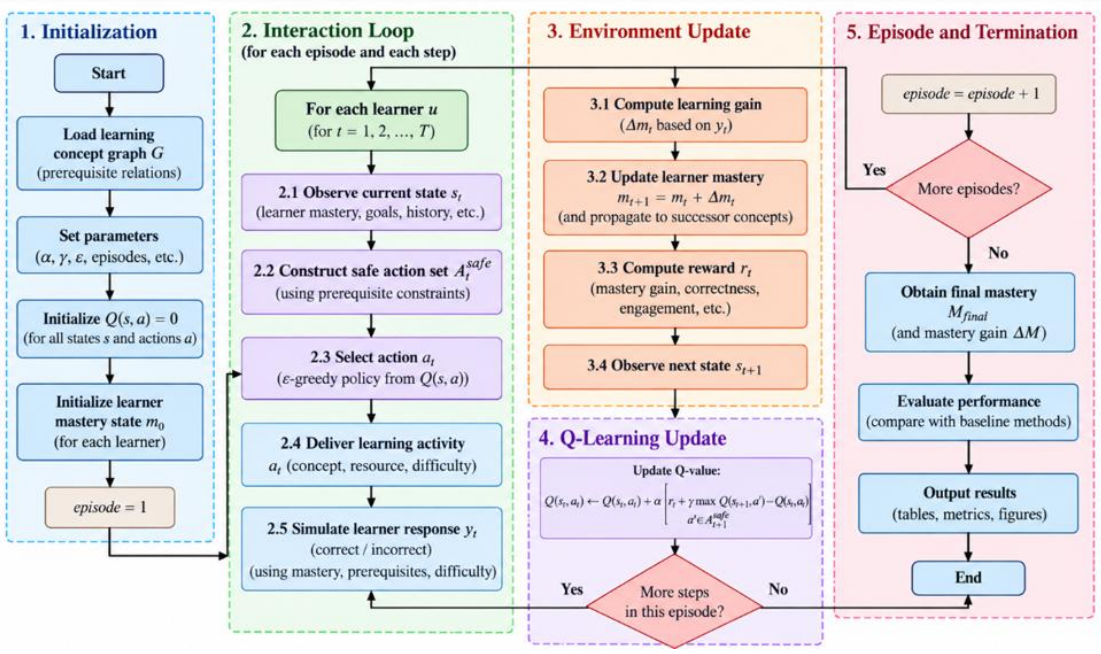


Figure 2. Operational flow of the graph-aware Q-learning recommendation cycle

4. SIMULATION-BASED FEASIBILITY STUDY

4.1. Synthetic environment and experimental protocol

The simulator contains 18 concepts arranged in a three-level prerequisite directed acyclic graph. Initial learner mastery is sampled from a beta distribution and correlated along prerequisite relations. At each interaction, response correctness is generated from a logistic function of concept mastery, prerequisite mastery, and item difficulty. Practice updates the attempted concept and produces a small transfer to successor concepts. This construction intentionally makes prerequisite information relevant, so the experiment tests internal coherence of the proposed graph-aware pipeline rather than external educational validity.

For learner-state estimation, each seed contains 800 synthetic learners with 50 interactions per learner (40,000 interactions). Learners are split 70/30 into train and test groups so that no learner appears in both sets. An independent KT proxy uses smoothed concept history, learner-level history, item difficulty, interaction count, and concept depth. The graph-enhanced variant uses the same estimator and adds prerequisite- and successor-state features. Both variants are trained with logistic regression to isolate the contribution of graph information. Performance is measured using AUC (Area Under the ROC Curve), log loss, Brier score, and accuracy. Five random seeds are reported.

For personalized recommendation, the simulator compares three policies: a static prerequisite rule, a KT-only adaptive policy that selects the lowest estimated mastery concept without graph filtering, and a graph-aware reinforcement-learning policy. The RL state discretizes mean mastery at the three graph levels into 27 states and chooses among remedial, target-practice, and challenge action categories. Training uses 1,200 synthetic learners per seed, 45 interactions per learner, learning rate α=0.15, discount factor γ=0.90, and ε-greedy exploration decreasing from 0.28 to 0.05. Evaluation uses 250 unseen synthetic learners per seed over 45 interactions. Reported outcomes are cumulative reward, final mastery, advanced-concept mastery, mean mastery gain, and prerequisite-violation rate.

4.2. Learner-state estimation results

Table 2 reports the mean and standard deviation across five seeds. Graph-enhanced KT produced a small but consistent improvement in ranking and probabilistic calibration: mean AUC increased from 0.6429 to 0.6454, log loss decreased from 0.6340 to 0.6330, and Brier score decreased from 0.22187 to 0.22142. Accuracy was essentially unchanged (0.63742 versus 0.63745). The magnitude of the change is deliberately interpreted as modest; the

simulation supports only the claim that prerequisite-state features can be incorporated coherently, not that they will produce the same improvement on real educational data.

Table 2. Simulation-based learner-state estimation results (mean ± SD over five seeds).

Model	AUC	Log loss	Brier	Accuracy
Independent KT	0.6429 ± 0.0046	0.6340 ± 0.0051	0.22187 ± 0.00228	0.63742 ± 0.00520
Graph-enhanced KT	0.6454 ± 0.0054	0.6330 ± 0.0057	0.22142 ± 0.00252	0.63745 ± 0.00529

4.3. Personalized recommendation results

The recommendation experiment produced a more differentiated result, as presented in Table 3. The graph-aware RL policy achieved the highest final mean mastery (0.3879) and mean mastery gain (0.1018), exceeding the static prerequisite rule (0.3798 and 0.0938) and the KT-only policy (0.3489 and 0.0628). Both the static rule and graph-aware RL policy produced zero prerequisite violations under the observable-state constraint, whereas the KT-only policy violated the prerequisite condition in approximately 84% of recommendations. However, cumulative reward for graph-aware RL (13.61) was not consistently higher than the static rule (13.75). Thus, the simulation does not justify a claim of universal RL superiority; instead, it highlights the dependence of policy behavior on reward design and state abstraction.

Table 3. Simulation-based recommendation outcomes (mean ± SD over five seeds)

Policy	Cum. reward	Final mastery	Advanced mastery	Mean gain	Prereq. violation
Static prerequisite rule	13.751 ± 0.054	0.3798 ± 0.0017	0.2863 ± 0.0023	0.0938 ± 0.0003	0.000
KT-only adaptive	1.099 ± 0.081	0.3489 ± 0.0021	0.3581 ± 0.0019	0.0628 ± 0.0004	0.840 ± 0.004
Graph-aware RL	13.611 ± 0.752	0.3879 ± 0.0060	0.3053 ± 0.0042	0.1018 ± 0.0046	0.000

As shown in Figure 3, the graph-aware RL policy achieved the highest final mastery and mastery gain among the evaluated strategies. The improvement was more pronounced relative to the KT-only policy, whereas the difference from the static prerequisite rule was smaller. These results provide preliminary evidence that incorporating graph information into adaptive decision-making can improve learner-mastery outcomes in the simulated environment.

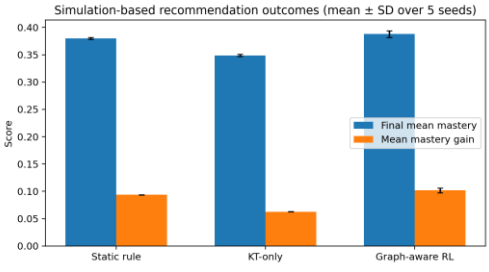


Figure 3. Final mastery and mastery gain in the synthetic recommendation experiment

4.4. Interpretation and threats to validity

These measurements address feasibility rather than external validity. First, the prerequisite graph is part of the data-generating process, so graph-aware methods are evaluated in an environment where graph structure is genuinely informative. Second, the KT models used here are lightweight proxies designed to isolate graph features rather than full neural implementations. Third, the recommendation simulator defines its own learning dynamics and reward function, and the mixed cumulative-reward result shows that conclusions depend on those choices. Fourth, no LLM quality, real-agent latency, teacher/learner explanation study, or real-world learning gain is measured in this simulation. Consequently, ASSISTments 2012 or another real interaction dataset remains necessary for learner-state validation, while a controlled system deployment is required for latency and orchestration claims. The present simulation should therefore be read as reproducible proof-of-concept evidence that the specified interfaces can generate measurable behavior, not as a substitute for empirical validation with real learners. The current study provides baseline comparisons but not a component-level ablation study. A full ablation should separately remove prerequisite masking, graph-derived state features, and successor-concept transfer to quantify the contribution of each mechanism.

5. CONCLUSION

This study specifies a distributed multi-agent framework for adaptive personalized online

learning in which a provenance-aware DEKG supports learner-state estimation, adaptive pedagogical decisions, and explainable recommendations. The architecture separates learner-facing interaction from distributed domain processing and defines explicit message contracts, configurable reliability terms, feasible-action construction, and an analytical coordination model without treating those design elements as already validated performance claims.

A simulation-based feasibility study provides the first quantitative evidence associated with the framework. Graph-enhanced KT achieved slightly better predictive performance than the independent KT proxy, with a higher AUC and lower log loss and Brier score. In the recommendation simulation, graph-aware reinforcement learning achieved the highest final mastery and mastery gain while respecting prerequisite constraints, although cumulative reward remained comparable to the static prerequisite rule. This mixed result is important because it indicates that integration alone does not guarantee superior performance and that reward specification remains a central empirical question.

The reported measurements are synthetic and therefore do not establish external validity on real learner data. Future work will reproduce the learner-state experiment on ASSISTments 2012, compare stronger KT baselines, tune the recommendation policy on held-out trajectories, and measure distributed-agent latency under controlled workloads. Real learner or teacher studies will additionally be required to evaluate educational effectiveness and explanation usefulness.

CONFLICT OF INTEREST

We declare that we have no conflicts of interest regarding the publication of this manuscript.

REFERENCES

Abu-Rasheed, H., Jumbo, C., Al Amin, R., Weber, C., Wiese, V., Obermaisser, R., & Fathi, M. (2025). LLM-assisted knowledge graph completion for curriculum and domain modelling in personalized higher education recommendations. In *2025 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1–5). IEEE. https://doi.org/10.1109/EDUCON62633.2025.11016377

Bian, H. (2025). *LLM-empowered knowledge graph construction: A survey*. ICAIS 2025. https://arxiv.org/abs/2510.20345

Bordes, A., Usunier, N., García-Durán, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems* (Vol. 26, pp. 2787–2795). https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html

Cheng, K., Peng, L., Wang, P., Ye, J., Sun, L., & Du, B. (2024). DyGKT: Dynamic graph learning for knowledge tracing. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (pp. 409–420). Association for Computing Machinery. https://doi.org/10.1145/3637528.3671773

Fu, Z. (2025). Integrating reinforcement learning with dynamic knowledge tracing for personalized learning path optimization. *Scientific Reports, 15*(1), Article 40202. https://doi.org/10.1038/s41598-025-23900-4

Gronauer, S., & Diepold, K. (2022). Multi-agent deep reinforcement learning: A survey. *Artificial Intelligence Review, 55*, 895–943. https://doi.org/10.1007/s10462-021-09996-w

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence, 101*(1–2), 99–134. https://doi.org/10.1016/S0004-3702(98)00023-X

Liu, F., Wang, Y., Feng, Q., Zhu, L., & Li, G. (2024). Optimizing e-learning environments: Leveraging large language models for personalized education pathways. In *Proceedings of the 5th International Conference on Education, Knowledge and Information Management* (pp. 811–817). Atlantis Press. https://doi.org/10.2991/978-94-6463-502-7_86

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems* (Vol. 30, pp. 4765–4774). https://papers.nips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html

Magar, S. P., Kumar, M. V., Manjula, B. M., & Gudala, L. (2025). Reinforcement learning with knowledge graphs for personalized news recommendation. *ITM Web of Conferences, 79*, Article 01039. https://doi.org/10.1051/itmconf/20257901039

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533. https://doi.org/10.1038/nature14236

Nguyen, T., & Nguyen, N. G. (2026). Enhancing knowledge tracing with persistent long-term memory via the Model Context Protocol. In *Lecture Notes in Networks and Systems* (Vol. 1988, pp. 400–410). Springer. https://doi.org/10.1007/978-3-032-26898-3_36

Nguyen, T., Do, D., Vi, H., Vo, K. T., Ta, T.-T., Nguyen, P., Nguyen, H.-T., & Nguyen-Hoang, T.-A. (2025). HetGNN-KGAT: Enhancing personalized course recommendation in MOOCs with knowledge graph attention networks. *IEEE Access, 13*, 192045–192068. https://doi.org/10.1109/ACCESS.2025.3630894

Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L. J., & Sohl-Dickstein, J. (2015). Deep knowledge tracing. In *Advances in Neural Information Processing Systems* (Vol. 28, pp. 505–513). https://proceedings.neurips.cc/paper/2015/hash/bac9162b47c56fc8a4d2a519803d51b3-Abstract.html

Rasheed, H. A., Weber, C., & Fathi, M. (2024). *Knowledge graphs as context sources for LLM-based explanations of learning recommendations*. arXiv. https://arxiv.org/abs/2403.03008

Shen, Y., & Chen, Y. (2025). Knowledge tracing enhanced by graph convolutional networks with self-supervised learning. *International Journal of Information and Communication Technology Education, 21*(1), 1–22. https://doi.org/10.4018/IJICTE.385130

Shiferaw, N. A., Leandre, S. H., Sinha, A., & Rout, D. (2024). *BERT-based approach for automating course articulation matrix construction with explainable AI*. arXiv. https://doi.org/10.48550/arXiv.2411.14254

Sun, Z., Deng, Z.-H., Nie, J.-Y., & Tang, J. (2019). RotatE: Knowledge graph embedding by relational rotation in complex space. In *7th International Conference on Learning Representations (ICLR 2019)*. https://openreview.net/forum?id=HkgEQnRqYQ

Wang, Y., Wei, T., Li, Q., & Zeng, L. (2026). *Beyond static question banks: Dynamic knowledge expansion via LLM-automated graph construction and adaptive generation*. arXiv. https://arxiv.org/abs/2602.00020

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning, 8*, 279–292. https://doi.org/10.1007/BF00992698

Wong, A., Bäck, T., Kononova, A. V., & Plaat, A. (2023). Deep multiagent reinforcement learning: Challenges and directions. *Artificial Intelligence Review, 56*, 5023–5056. https://doi.org/10.1007/s10462-022-10299-x

Wu, H., Zhong, X., & Yuan, J. (2025). Textbook2KG: Decoding knowledge graph from very long textual content via prompt engineering. In *2025 International Conference on Artificial Intelligence and Digital Ethics (ICAIDE)*. IEEE. https://doi.org/10.1109/ICAIDE65466.2025.11189379

Zhou, L.-Y., & Wang, Y.-Y. (2025). Simulation of personalized English learning path recommendation system based on knowledge graph and deep reinforcement learning. *Scientific Reports, 15*(1), Article 34554. https://doi.org/10.1038/s41598-025-17918-x