Advances in Vocational and Technical Education (2026)
Clausius Scientific Press, Canada
DOI: 10.23977/avte.2026.080207
ISSN 2523-5834 Vol. 8 Num. 2

A Knowledge-Graph-Constrained Learning Path Recommender for Linux Practical Training in Chinese Higher Vocational Education

Xianjie Xie*

Faculty of Computer Information, Kunming Metallurgy College, Kunming, 650033, China
934363055@qq.com
**Corresponding author*

Keywords: Technical and vocational education and training; Knowledge graph; Q-learning; Learning path recommendation; Linux practical training

Abstract: Linux practical training requires students to coordinate command syntax, file-path reasoning, permission management, service administration, and shell scripting. However, vocational learners often enter laboratory work with markedly different levels of prior knowledge and practical fluency. This study develops a learning-path recommender that combines a 40-point Linux course knowledge graph with a graph-constrained Q-learning policy. The graph encodes prerequisite relations and maps 12 practical tasks to their required knowledge points, while Q-learning prioritizes eligible units according to learner state and task progress. A transparent simulator representing five pedagogically motivated learner profiles was used for pre-deployment evaluation. In a five-strategy comparison, the proposed KG+Q method achieved the highest mean mastery (0.4693) and task-completion rate (0.3744), the lowest error count (26.3533), and zero prerequisite violations. Ablation results indicate that graph constraints preserve instructional order, while modeled adaptive support contributes to mastery improvement and error control. Parameter sensitivity analyses and 20 independent simulation repetitions showed consistent performance patterns. The proposed framework provides an interpretable basis for designing and testing adaptive Linux practical-training paths, although classroom evaluation remains necessary before conclusions about student learning effectiveness can be drawn.

1. Introduction

Linux operating-system training is a core practical component of many vocational computer programs and supports later study in networking, cloud computing, cybersecurity, server maintenance, and software deployment. Competence is demonstrated through operations such as directory management, permission configuration, process monitoring, service control, shell scripting, and troubleshooting. These activities have clear dependencies: learners should understand paths before manipulating files, users and groups before configuring permissions, and redirection before writing scripts.

Learners nevertheless differ in command-line experience and operational fluency. A fixed sequence is easy to administer but may expose weak learners to integrated tasks before prerequisite knowledge is secure, while stronger learners repeat familiar content. Learning-path recommendation has therefore been studied through graph models, optimization, learning analytics, and machine learning [1,2]. Knowledge graphs make concept and prerequisite relations explicit [3,4], whereas reinforcement learning supports sequential decisions whose value depends on later learner responses [5-8]. Computing laboratories have also been used to analyze terminal histories, reinforce operating-systems knowledge, and support assisted exercises [9-11].

Two practical difficulties remain for small vocational courses. First, many adaptive systems assume large historical datasets that are not yet available. Second, an unconstrained optimizer may recommend a locally beneficial but instructionally premature task. This study addresses both issues by separating pedagogical eligibility from adaptive prioritization. A manually inspectable Linux knowledge graph filters the action space, and a Q-learning policy selects among eligible knowledge points. A controlled learner simulator is used to test this mechanism before classroom deployment; no claim is made about measured learning effects.

The study contributes: (1) a course model containing forty Linux knowledge points, explicit prerequisites, and twelve practical tasks; (2) a graph-constrained Q-learning policy that combines structural safety with learner-state adaptation; and (3) a pre-deployment evaluation including baseline comparison, component ablation, statistical testing, sensitivity analysis, and independent simulation repetitions.

2. Related work

Learning-path recommenders select resources or activities according to learner characteristics, goals, and process data. Early graph approaches emphasized competency dependencies [1,2], while later models incorporated multidimensional knowledge graphs [3,4], process mining, knowledge tracing, and reinforcement learning [12,13]. These methods demonstrate the value of dynamic adaptation but often rely on extensive learner logs or online-resource repositories.

Knowledge tracing estimates mastery from interaction sequences [14,15], and educational data mining provides broader methods for learner modeling and intervention [16]. Such approaches are valuable after sufficient observations are available. In an earlier deployment stage, however, a transparent simulator can expose design assumptions and test whether a policy respects course structure. This is particularly relevant to Linux laboratories, where prerequisite relations are observable and terminal activity can later provide calibration evidence [9]. The present work therefore uses the knowledge graph as an explicit instructional constraint rather than as an opaque learned feature, and uses tabular Q-learning to retain inspectability for instructors.

3. Proposed method

3.1 Course knowledge graph and practical-task mapping

The target setting is a 48-64-hour Linux course for second-year vocational computer students. Seven modules cover fundamentals, files and directories, users and permissions, text processing and pipelines, packages and services, processes and resources, and shell scripting. Forty knowledge points and their directed prerequisites were constructed from the syllabus, textbook organization, practical tasks, and Linux laboratory teaching experience. The resulting graph is a revisable course model rather than a universal Linux ontology.

Each point records its module, name, difficulty, practice weight, and error category. Twelve tasks, including project-directory organization, text search, shared-directory permission

configuration, service diagnosis, process control, log backup, and integrated troubleshooting, are mapped to the knowledge points required for completion. Figure 1 summarizes the module-level prerequisites and task mapping. It illustrates why mean mastery alone is insufficient: an integrated task can remain unready when one required point is missing.

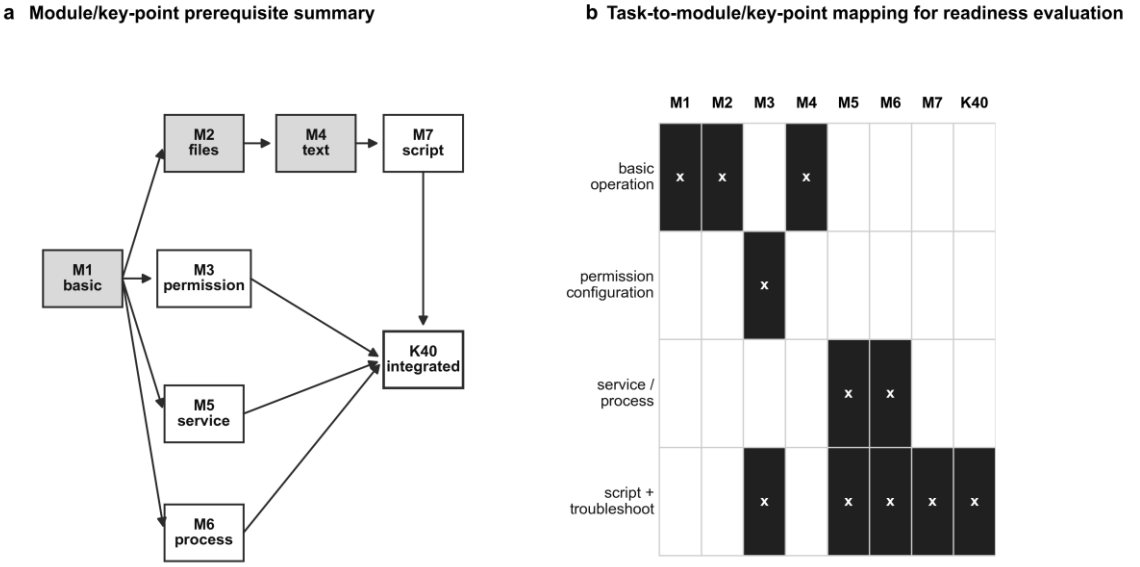

Figure 1. Simplified module/key-point prerequisite summary and practical task mapping.

3.2 Learner state and simulated response

Learner state is represented by a mastery vector $\mathbf{M}_t = [m_1(t), \ldots, m_{40}(t)]$, where $0 \leq m_i(t) \leq 1$. A profile also specifies learning ability, practical ability, weakness category, and initial mastery. Table 1 defines five profiles chosen to represent common instructional patterns rather than population estimates.

Table 1. Simulated learner profile parameters.

Profile	Learning ability	Practical ability	Weakness category	Initial mastery
Strong foundation	1.18	1.10	None	0.32
Weak foundation	0.82	0.88	Basic	0.16
Limited practical fluency	0.98	0.72	Integrated	0.24
Path-and-permission difficulty	0.92	0.86	Permission	0.22
Shell-scripting difficulty	0.94	0.82	Shell	0.23

Initial mastery varies with profile, point difficulty, weakness penalty, and a small random term:

$$m_i(0) = \text{clip}\{m_0 - 0.04(d_i - 1) - \pi_i + U(-0.03,0.03),0.02,0.70\}.$$

Here, $\pi_i = 0.06$ when the profile has a general basic weakness or its weakness category matches the point's error category; otherwise, $\pi_i = 0$.

After point i is studied, mastery gain and state transition are:

$$g_i = 0.42af_df_rf_pf_wf_{\text{int}}f_s\epsilon_i, m_i(t+1) = \min\{1, m_i(t) + g_i[1 - m_i(t)]\}.$$

Let $\rho_i(t)$ be the mean mastery of point i's prerequisites, or 1 when none are defined. The factors

are $f_d = (0.78 + 0.18d_i)^{-1}$, $f_r = 0.55 + 0.45\rho_i(t)$, and $f_p = 0.70 + 0.30pw_i$, where p is practical ability and w_i is practice weight. The weakness and integrated-task multipliers are 0.78 and 0.88 when applicable, and 1 otherwise; $\epsilon_i \sim U(0.90,1.10)$. For methods with adaptive support, $f_s = 1.16$ and error risk is multiplied by 0.82; otherwise, $f_s = 1$. Candidate scoring separately rewards a match between the point's error category and the learner profile. Error probability depends on difficulty, prerequisite satisfaction, practical ability, and profile weakness, while time cost increases with point difficulty and generated errors. These equations are controlled behavioral abstractions, not fitted classroom models.

3.3 Graph-constrained Q-learning policy

The tabular state is a five-interval discretization of mean mastery. Candidate ranking also uses point mastery, prerequisite satisfaction, task progress, and weakness category; errors and time cost enter the reward. The reward is

$$R_t = 8G_t + 10P_t + 1.2S_t - 0.35E_t - 0.06C_t - 0.8V_t,$$

where G_t is mastery gain, P_t is the change in task-completion rate, and S_t indicates that the selected point reaches the mastery threshold after the action; E_t, C_t, and V_t are error count, time cost, and prerequisite violations. Q-values are updated by

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \mu[R_t + \lambda \max_{a} Q(s_{t+1}, a') - Q(s_t, a_t)],$$

with learning rate $\mu = 0.18$ and discount factor $\lambda = 0.88$. The graph retains an unfinished point as an action only when prerequisite satisfaction reaches 0.50:

$$A_t = \{i \in \mathcal{K} \mid m_i(t) < 0.82,\ \rho_i(t) \geq 0.50\}.$$

For eligible point i, the priority score adds to $Q(s_t, i)$ a mastery-target term $0.35[1 - |0.84 - m_i(t)|]$, prerequisite term $0.24\rho_i(t)$, task-progress term $0.75\tau_i(t)$, and difficulty penalty $0.025d_i$. It adds 0.18 when the profile weakness matches the point's error category and 0.10 for a high-difficulty point under an integrated-task weakness. Here, $\tau_i(t)$ is the largest proportion of already-mastered companion points among incomplete tasks containing point i. Training uses an ε-greedy policy; evaluation fixes $\varepsilon = 0.02$. At each step, the system constructs A_t, selects a point, simulates the learner response, updates mastery and task progress, computes the reward, and updates the Q-table during training. A task is complete only when every mapped knowledge point reaches the mastery threshold. Figure 2 shows the complete information flow.

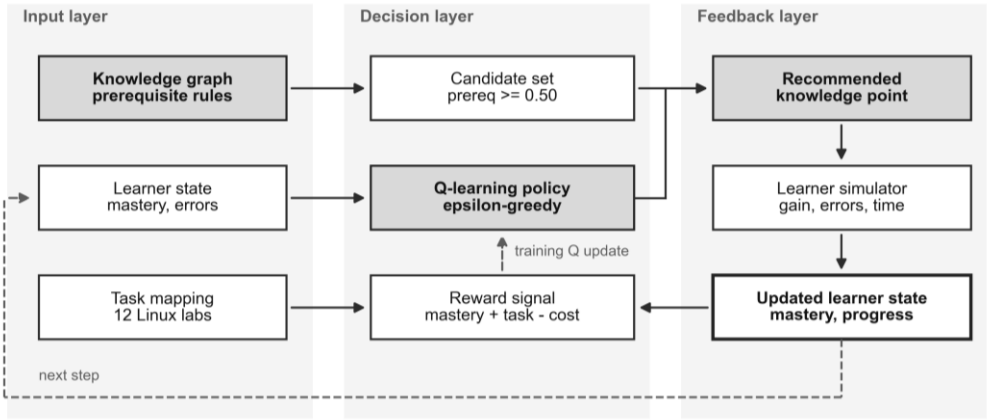

Figure 2. Knowledge-graph-constrained Q-learning recommendation workflow.

4. Simulation design

The main experiment uses five profiles with thirty virtual learners each, giving 150 learners per method. The Q-table is trained for 180 episodes, with a profile sampled for each episode, and then fixed during evaluation to avoid learning from test episodes. Table 2 summarizes the principal settings. Reward coefficients are given in Section 3.3.

Table 2. Main simulation parameters.

Parameter	Value
Knowledge points / practical tasks	40 / 12
Learner profiles / learners per profile	5 / 30
Maximum learning steps	90
Mastery / prerequisite threshold	0.82 / 0.50
Q-learning training episodes	180
Learning rate / discount factor	0.18 / 0.88
Training exploration	$\max(0.05, 0.35e^{-t/32})$
Evaluation exploration	0.02
Adaptive-support gain	1.16

KG+Q is compared with fixed, teacher-authored, difficulty-increasing, and graph-rule sequences. Metrics include mean mastery, task completion, errors, time cost, and prerequisite violations; Mann-Whitney U tests and Cliff's delta compare KG+Q with each baseline. In Table 2, t is the within-episode decision step. Step sensitivity uses 15 learners per profile and 90 training episodes; reward sensitivity uses 12 learners per profile and 140 episodes. Twenty independent-seed repetitions use 12 learners per profile and 180 training episodes each.

5. Results

5.1 Main comparison

Figure 3 compares (a) mean mastery, (b) task completion, (c) errors, and (d) prerequisite violations. KG+Q has the highest mean mastery (0.4693) and task-completion rate (0.3744), the fewest errors (26.3533), and zero violations. The fixed sequence yields 0.4212 mean mastery, 0.3339 task completion, and 35.2000 errors. Although the difficulty sequence has the lowest time cost (197.5768 versus 211.1182 for KG+Q), its task-completion rate is also the lowest (0.1667), suggesting that repeated prioritization of easier points can leave integrated tasks unfinished. Markers at zero in Figure 3(d) are observed values, not missing data.

The fixed, teacher-authored, graph-rule, and KG+Q sequences produce no violations because each follows dependency-compatible ordering or prerequisite filtering. KG+Q's advantage therefore lies in prioritization within the allowable action space. Profile-level comparisons also show higher mastery and fewer errors than the fixed sequence for each simulated profile; these are properties of the specified simulator, not observations of individual students.

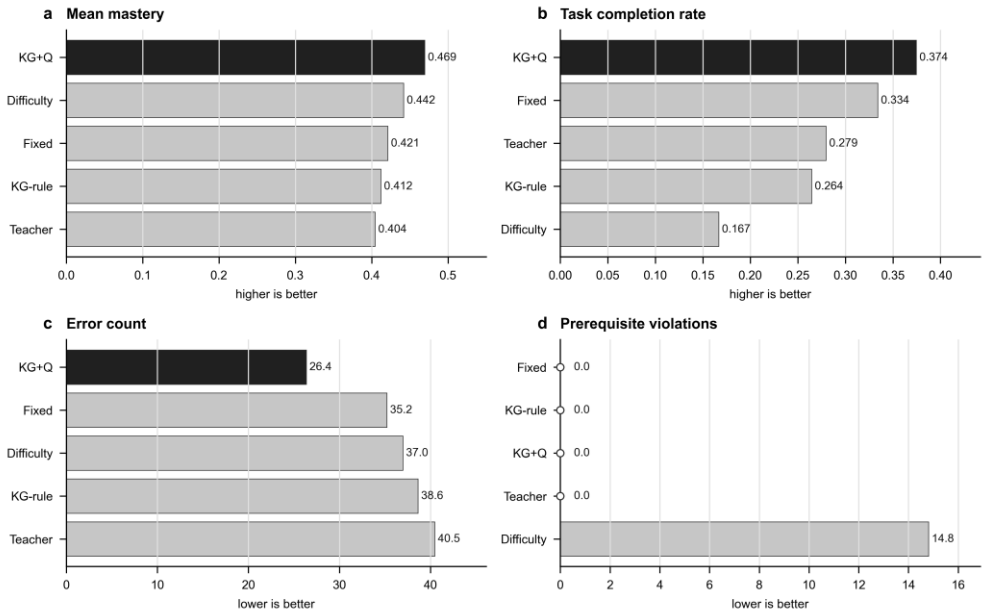

Figure 3. Main simulation comparison across recommendation methods.

5.2 Component, statistical, and sensitivity analyses

Figures 4(a) and 4(b) report ablation results. Q-learning without the graph attains a mean mastery of 0.4463 and 29.4533 errors but produces 20.1200 prerequisite violations. Removing the modeled adaptive-support response preserves zero violations but lowers mean mastery from 0.4644 to 0.4241 and increases errors from 27.3067 to 33.8933. Thus, graph constraints protect instructional order, while modeled adaptive support contributes to mastery and error control.

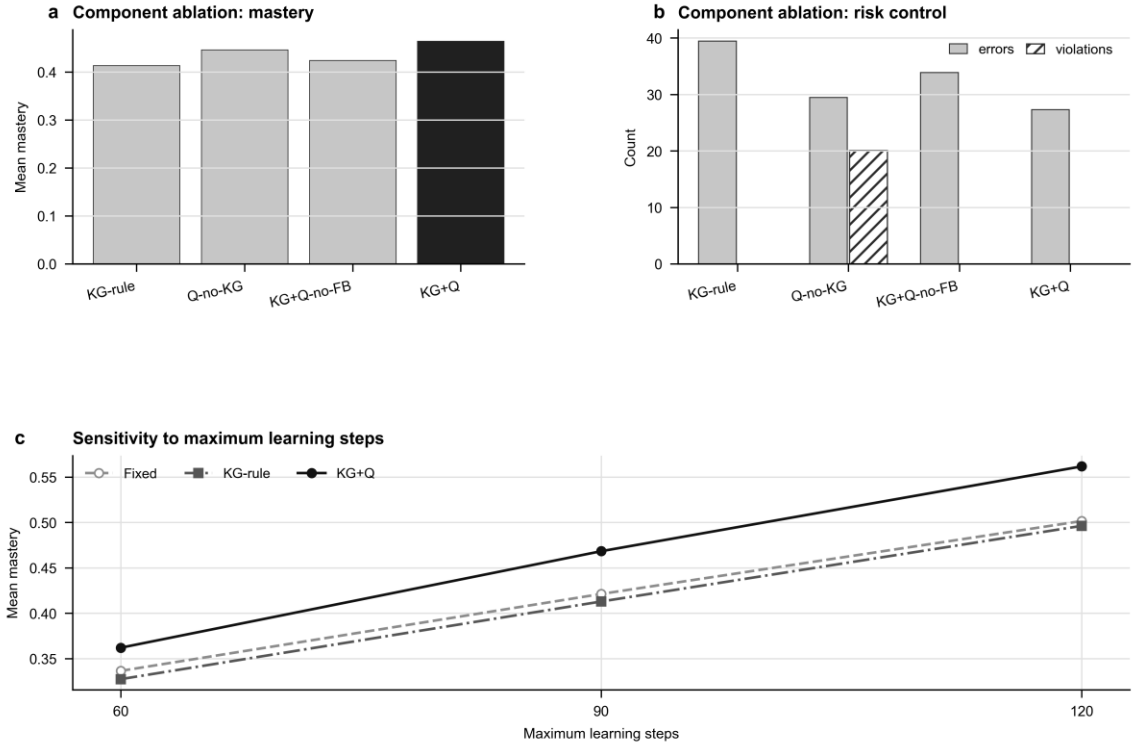

Figure 4. Ablation and robustness analysis.

Mann-Whitney U tests show higher mean mastery and task completion and fewer errors for KG+Q than for each baseline (all $p < 0.001$). Cliff's delta ranges from 0.2828 to 0.4400 for mean mastery, from 0.3505 to 1.0000 for task completion, and from -0.8081 to -0.9300 for error count; negative error effects indicate fewer errors under KG+Q. These tests describe separation among simulated learner episodes and do not support population inference about real students. The step-sensitivity curves in Figure 4(c) retain the mean-mastery advantage at 60, 90, and 120 steps. Across tested reward settings, the mean-mastery advantage over the fixed sequence ranges from 0.0235 to 0.0711, and error reduction is retained. Task completion is less stable because it depends on whether reward design emphasizes immediate task progress or prerequisite strengthening.

5.3 Independent-seed repeatability

Table 3 summarizes twenty full repetitions. KG+Q retains the highest mean mastery and task completion and the lowest error count. Its bootstrap 95% interval is 0.4648-0.4667 for mean mastery and 27.3883-28.0175 for errors. These intervals quantify variation within the simulator rather than uncertainty in a real student population.

Table 3. Multi-seed results across 20 independent simulation repetitions (mean ± standard deviation).

Method	Mean mastery	Task completion	Errors	Violations
Fixed	0.4213 ± 0.0004	0.3357 ± 0.0014	36.1283 ± 0.6539	0.0000
Teacher	0.4041 ± 0.0004	0.2808 ± 0.0017	40.5525 ± 0.6341	0.0000
Difficulty	0.4415 ± 0.0005	0.1667 ± 0.0000	37.0967 ± 0.9065	15.3975
KG-rule	0.4127 ± 0.0011	0.2688 ± 0.0068	38.8000 ± 0.8273	0.0000
KG+Q	0.4657 ± 0.0022	0.3623 ± 0.0251	27.7150 ± 0.7297	0.0000

6. Discussion

The results support the separation of explicit course structure from adaptive decision making. A graph-only policy preserves prerequisites but cannot respond dynamically to modeled weaknesses; Q-learning without the graph can improve selected outcomes while recommending premature units. KG+Q combines both roles and keeps its decisions inspectable because instructors can review the course graph, task mapping, and priority features.

This design is relevant to vocational laboratories where historical data are limited but instructional dependencies are well understood. Before deployment, simulation can reveal premature recommendations and reward trade-offs. During later implementation, mastery estimates could be updated from quizzes, terminal histories, laboratory submissions, and learning-management-system logs. The current evidence does not establish classroom effectiveness.

Three limitations delimit the conclusions. First, simulated profiles cannot reproduce the full variability of learner behavior, instructor intervention, or task assessment. Second, profile parameters and reward weights are transparent design assumptions rather than estimates fitted to student data. Third, tabular Q-learning favors interpretability but scales poorly as state features increase. A subsequent classroom study should preregister outcomes such as task quality, command-level errors, time on task, post-test performance, and instructor workload, and should calibrate model parameters from observed interactions.

7. Conclusion

This study developed a knowledge-graph-constrained learning-path recommender for Linux practical training in Chinese higher vocational education. The graph enforces course prerequisites, while Q-learning adapts selection among eligible knowledge points. Under the controlled simulator, KG+Q improves mastery and error control while preserving prerequisite consistency across baseline, ablation, sensitivity, and independent-seed analyses. The contribution is an interpretable pre-deployment framework; classroom evaluation remains necessary before claims about student learning can be made.

Acknowledgements

This work was supported by the Kunming Metallurgy College Research Fund Project, "AI-Enabled Intelligent Scheduling and Governance of OpenStack Private-Cloud Resources in Higher Vocational Institutions" (Grant No. 2026xjy02).

References

[1] Nabizadeh AH, Leal JP, Rafsanjani HN, Shah RR. Learning path personalization and recommendation methods: A survey of the state-of-the-art. Expert Systems with Applications. 2020;159:113596. doi: 10.1016/j.eswa.2020.113596.

[2] Durand G, Belacel N, Laplante F. Graph theory based model for learning path recommendation. Information Sciences. 2013;251:10-21. doi: 10.1016/j.ins.2013.04.017.

[3] Shi D, Wang T, Xing H, Xu H. A learning path recommendation model based on a multidimensional knowledge graph framework for e-learning. Knowledge-Based Systems. 2020;195:105618. doi: 10.1016/j.knosys.2020.105618.

[4] Zhang S, Hui N, Zhai P, Xu J, Cao L, Wang Q. A fine-grained and multi-context-aware learning path recommendation model over knowledge graphs for online learning communities. Information Processing & Management. 2023;60(5):103464. doi: 10.1016/j.ipm.2023.103464.

[5] Tan C, Han R, Ye R, Chen K. Adaptive learning recommendation strategy based on deep Q-learning. Applied Psychological Measurement. 2020;44(4):251-266. doi: 10.1177/0146621619858674.

[6] Rafferty AN, Brunskill E, Griffiths TL, Shafto P. Faster teaching via POMDP planning. Cognitive Science. 2016;40(6):1290-1332. doi: 10.1111/COGS.12290.

[7] Watkins CJCH, Dayan P. Q-learning. Machine Learning. 1992;8(3-4):279-292. doi: 10.1007/BF00992698.

[8] Sutton RS, Barto AG. Reinforcement Learning: An Introduction. 2nd ed. Cambridge, MA: MIT Press; 2018.

[9] Mirkovic J, Aggarwal A, Weinman D, Lepe P, Mache J, Weiss R. Using terminal histories to monitor student progress on hands-on exercises. In: Proceedings of the 51st ACM Technical Symposium on Computer Science Education; 2020. p. 866-872. doi: 10.1145/3328778.3366935.

[10] Freedman R. Using an operating systems class to strengthen students' knowledge of C++. In: Proceedings of the 51st ACM Technical Symposium on Computer Science Education; 2020. p. 947-953. doi: 10.1145/3328778.3366936.

[11] Chu ETH, Fang CW. CALEE: A computer-assisted learning system for embedded OS laboratory exercises. Computers & Education. 2015;84:36-48. doi: 10.1016/j.compedu.2015.01.006.

[12] Zhang F, Feng X, Wang Y. Personalized process-type learning path recommendation based on process mining and deep knowledge tracing. Knowledge-Based Systems. 2024;303:112431. doi: 10.1016/j.knosys.2024.112431.

[13] Ma D, Zhu H, Liao S, Chen Y, Liu J, Tian F, Chen P. Learning path recommendation with multi-behavior user modeling and cascading deep Q networks. Knowledge-Based Systems. 2024;294:111743. doi: 10.1016/j.knosys.2024.111743.

[14] Corbett AT, Anderson JR. Knowledge tracing: Modeling the acquisition of procedural knowledge. User Modeling and User-Adapted Interaction. 1995;4:253-278. doi: 10.1007/BF01099821.

[15] Piech C, Bassen J, Huang J, Ganguli S, Sahami M, Guibas LJ, Sohl-Dickstein J. Deep knowledge tracing. In: Advances in Neural Information Processing Systems. 2015;28:505-513.

[16] Romero C, Ventura S. Educational data mining and learning analytics: An updated survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2020;10(3):e1355. doi: 10.1002/widm.1355.