ONE SIZE FITS NONE: EXERCISE-ADAPTIVE COGNITIVE ROUTE INDUCTION FOR COGNITIVE DIAGNOSIS WITH LLM SUPERVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Cognitive diagnosis aims to identify students' learning weaknesses, yet most existing models operate at the concept level, representing exercises as item IDs, Q-matrix entries, concept labels, or dense embeddings. Such representations obscure heterogeneous reasoning demands within the same concept: SQL exercises tagged with JOIN may require either simple recall or complex schema inspection, key-relation reasoning, constraint filtering, and query construction. To address this limitation, we propose ***Cognitive Route Induction*** **(CRI)**, which parses exercise text into a Bloom-based cognitive route graph, where nodes denote activated cognitive processes and edges encode prerequisite dependencies. CRI defines the target representation for fine-grained diagnosis: the cognitive operations required by an exercise and their dependencies. Building on CRI, we introduce **ROUTECD**, a route-aware cognitive diagnosis model that projects each exercise route into a shared concept-by-process space and predicts responses from the gap between student cognitive state and exercise demand. LLM-generated analyses provide node- and edge-level graph supervision during training, while inference requires no LLM calls. Experiments show improved response prediction and cognitive-structure modeling, and ablations confirm the diagnostic value of induced routes. Our code is available at: https://anonymous.4open.science/r/ROUTECD.

1 INTRODUCTION

Cognitive diagnosis aims to identify students' learning weaknesses from their responses to exercises. A useful diagnosis should not only predict whether a student will answer an exercise correctly, but also reveal what kind of learning support the student may need. Existing cognitive diagnosis models have made substantial progress by modeling student proficiency, item difficulty, and concept mastery. However, most of them formulate the problem primarily at the concept level: an exercise is represented as an item ID, a Q-matrix entry, a concept label, or a dense embedding associated with one or more knowledge concepts. While such representations are effective for response prediction, they are often too coarse to explain the cognitive source of an error. This limitation becomes evident when exercises sharing the same concept require different reasoning processes. As shown in Figure 1(a), two SQL exercises may share the same concept label, JOIN, while requiring substantially different reasoning processes: one mainly tests recall of the join operation, whereas the other requires schema inspection, key-relation reasoning, filtering, and query construction. A concept-level diagnosis model can therefore identify the involved concept, but cannot determine whether a failed response

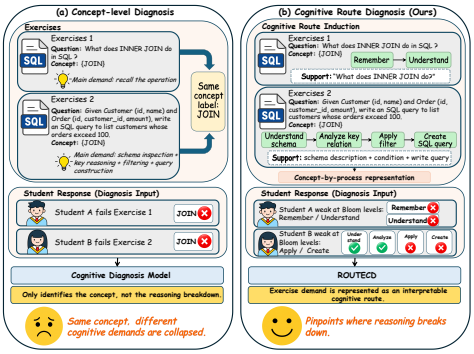


Figure 1: From Concept-Level Diagnosis to Cognitive-Route Diagnosis. Concept-level diagnosis collapses exercises with the same concept label despite different reasoning demands. Cognitive-route diagnosis represents each exercise as an interpretable Bloom-based route and pinpoints where student reasoning breaks down.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

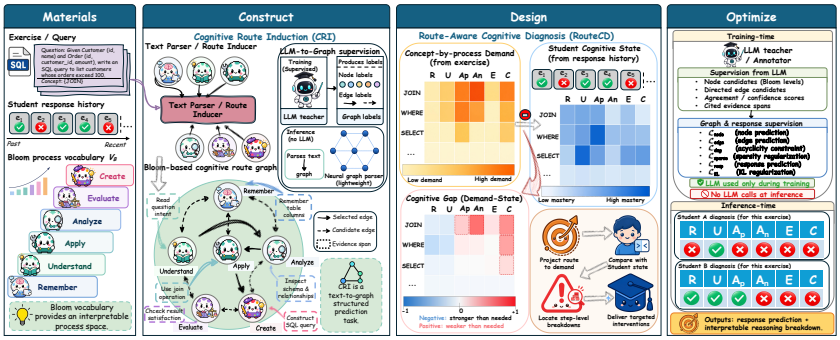


Figure 2: Overview of RouteCD.

reflects weak recall, poor schema understanding, difficulty in applying filters, or failure to construct a query. Figure 1(b) motivates our cognitive-route view. Rather than representing each exercise only by its concept label, we induce an exercise-specific Bloom-based route that makes the required cognitive processes and their dependencies explicit. This route-level representation enables diagnosis in a shared concept-by-process space, allowing the model to compare student cognitive state with exercise demand and pinpoint where reasoning breaks down. We therefore argue that fine-grained cognitive diagnosis requires a shift from concept-level modeling to cognitive-process-level modeling: the model should recover not only the concepts involved in an exercise, but also the cognitive operations required to solve it and the dependencies among them.

In this work, we refer to this structured demand as a cognitive route. A cognitive route describes the set of cognitive operations required by an exercise and their prerequisite dependencies. Solving an exercise may require recalling a definition, understanding a condition, applying a procedure, analyzing constraints, evaluating alternatives, or constructing a solution. These operations are not merely independent labels; they often form a dependency structure. For instance, understanding a database schema may be required before applying a query pattern, and coordinating multiple constraints may be required before constructing a final SQL query. To instantiate cognitive routes, we adopt Bloom's taxonomy (Anderson et al., 2001), which organizes educational objectives into six cognitive processes: *Remember*, *Understand*, *Apply*, *Analyze*, *Evaluate*, and *Create*. We use this taxonomy as a compact, interpretable, and education-grounded vocabulary for describing exercise-level cognitive demands, rather than as a direct measurement of students' internal mental states.

Based on this view, we propose *Cognitive Route Induction* (CRI), a fine-grained task that maps exercise text to a Bloom-based cognitive route graph, where nodes represent activated cognitive processes, directed edges encode prerequisite dependencies, and supporting spans make the induced structure auditable. CRI turns exercise understanding into a text-to-structure problem and defines the target representation for fine-grained cognitive diagnosis. A key challenge is the lack of cognitive-route annotations: standard educational datasets usually provide student responses and concept labels, but not cognitive processes, prerequisite dependencies, or supporting spans. Since manually constructing such structures is costly, we use LLMs as training-time cognitive annotators. Given an exercise, the LLM produces Bloom-process activations and prerequisite dependencies, which are distilled into node- and edge-level graph supervision. Thus, LLMs externalize latent cognitive structure from exercise text during training, while no LLM calls are required at inference time.

Building on CRI, we introduce RouteCD, a route-aware cognitive diagnosis model that first induces a cognitive route for each exercise and then projects it into a shared concept-by-process space, where responses are predicted from the cognitive gap between exercise demand and student state. The induced routes are used as functional diagnostic structures rather than post-hoc explanations. Experiments on multiple datasets show that RouteCD outperforms concept-level and text-enhanced baselines, and ablations confirm that cognitive routes improve both response prediction and step-level diagnosis.

The contributions of this work are threefold:

- **Cognitive route induction for exercise understanding.** We advance cognitive diagnosis from concept-level modeling to cognitive-process-level modeling by introducing Cognitive

Route Induction (CRI), a text-to-graph formulation that parses exercise text into Bloom-based cognitive route graphs.

- **Route-aware cognitive diagnosis.** We propose RouteCD, a cognitive diagnosis model that uses CRI-induced routes to represent exercise demand and compares this demand with student cognitive state in a shared concept-by-process space for response prediction and reasoning-breakdown diagnosis.
- **LLM-to-graph supervision.** We show that LLM-generated cognitive analyses can be distilled into node- and edge-level graph supervision, enabling efficient non-LLM inference while preserving the benefits of structured cognitive reasoning.

2 Related Work

Cognitive Diagnosis and Knowledge Tracing. Classical cognitive diagnosis models, such as IRT (Lord, 1980), MIRT (Reckase, 2009), and DINA (de la Torre, 2009), represent student ability, item difficulty, and concept mastery with compact latent variables. Neural cognitive diagnosis models, such as NCD (Wang et al., 2020a) and AGCDM (Pei et al., 2022), improve response prediction with more expressive interaction functions. Graph-based methods further model student–exercise–concept relations or concept-level dependencies, as in RCD (Gao et al., 2021) and CACD (Wang et al., 2024). More recent cognition-aware and language-enhanced methods, such as CCD (Gu et al., 2025) and EduEmbed (Liu et al., 2026), introduce cognitive signals or semantic embeddings into diagnosis.

Despite these advances, most existing methods still represent exercises mainly by item IDs, concept labels, interaction patterns, or dense embeddings. Even when graph structures are used, they are typically defined over student–exercise–concept interactions or shared concept-level relations, rather than over the cognitive process structure of each exercise. In contrast, CRI induces exercise-specific Bloom-based cognitive route graphs from exercise text, making the required cognitive operations and their prerequisite dependencies explicit; RouteCD then uses these routes as functional diagnostic structures for route-aware response prediction.

Bloom's Taxonomy in Educational NLP. Bloom's taxonomy (Anderson et al., 2001) organizes educational objectives into six cognitive processes. Prior work applies this vocabulary mainly for flat question classification, assigning a single Bloom level per exercise (Li et al., 2022; Hwang et al., 2024). Some knowledge tracing studies have also explored Bloom-level information in student modeling (Lalwani & Agrawal, 2018). Unlike these label- or feature-based uses, CRI induces a directed prerequisite graph over activated Bloom processes to explicitly model the cognitive operations required by an exercise and their dependencies. To learn such structures, LLM-generated analyses are distilled into node- and edge-level graph supervision during training, enabling structured route induction without inference-time LLM calls.

Text-to-Structure Parsing. A broader line of work parses natural language into structured representations, such as AMR parsing via sequence-to-sequence generation (Konstas et al., 2017) or graph-prediction with latent alignment (Lyu & Titov, 2018). These approaches target general-purpose semantic structure rather than education-specific cognitive demand. CRI instead targets a domain-specific structured prediction problem: mapping exercise text to a Bloom-grounded cognitive route graph whose nodes and edges carry pedagogical rather than purely syntactic or semantic meaning.

3 Method

Figure 2 summarizes the RouteCD pipeline. Given exercise text, concept metadata, and student histories, CRI induces Bloom-based cognitive routes with LLM-derived training supervision. RouteCD then maps routes and student states into a shared concept-by-process space, predicts responses from their cognitive gap, and is optimized with response, graph, sparsity, acyclicity, and KL objectives. Inference uses only the learned route inducer and requires no LLM calls.

3.1 Problem Formulation and Cognitive Route Definition

We first formalize the inputs and the cognitive route representation used by RouteCD. **Input data.** Let $\mathcal{D} = \{(s, e, y_{s,e})\}$ denote student–exercise responses, where $y_{s,e} \in \{0, 1\}$ indicates whether student s answers exercise e correctly. Each exercise e contains text x_e and concept metadata $q_e \in \{0, 1\}^{|K|}$ over domain concepts $K = \{k_1, \ldots, k_{|K|}\}$. The response history of student s is denoted as $\mathcal{H}_s = \{(e_t, y_{s,e_t})\}_{t<T}$. **Bloom process vocabulary.** We use the Bloom process vocabulary $V_B = \{\mathrm{R}, \mathrm{U}, \mathrm{Ap}, \mathrm{An}, \mathrm{E}, \mathrm{C}\}$, corresponding to Remember, Understand, Apply, Analyze, Evaluate, and Create. This vocabulary provides an interpretable process space for describing exercise-level cognitive demand, rather than directly measuring students' internal mental states. **Cognitive route graph.** For exercise e, a cognitive route is defined as $\mathcal{G}_e = (V_B, z_e, A_e, \mathcal{S}_e)$, where $z_e \in \{0, 1\}^6$ marks activated Bloom processes, $A_e \in \{0, 1\}^{6\times 6}$ marks directed prerequisite dependencies, and $\mathcal{S}_e$ stores supporting spans for positive nodes and edges. This representation makes the diagnostic target explicit: beyond identifying the involved concepts, we model which cognitive operations an exercise requires and how these operations depend on one another. CRI is then formulated as a structured prediction task:

$$f_\theta : (x_e, q_e) \to (\hat{z}_e, \hat{A}_e). \tag{1}$$

where $\hat{z}_e$ identifies required cognitive operations and $\hat{A}_e$ identifies prerequisite dependencies among them. Supporting spans $\mathcal{S}_e$, when provided by LLM supervision, are used as training-time grounding evidence for node and edge labels, but are not predicted at inference time. We allow non-chain routes: $\hat{A}_e$ is not constrained to follow the fixed Remember $\to$ Understand $\to$ Apply $\to$ Analyze $\to$ Evaluate $\to$ Create order. Instead, Bloom-level dependencies are treated as exercise-specific directed relations, consistent with revised Bloom's taxonomy and local prerequisite modeling in cognitive diagnosis; further discussion is provided in Appendix B.

3.2 Construct: CRI — Cognitive Route Induction

After defining the inputs, we describe the Construct stage in Figure 2, where CRI converts raw exercise information into an exercise-specific cognitive route graph. Concept metadata indicates which knowledge components are involved, but it does not specify which cognitive operations are required or how they depend on one another. CRI addresses this gap by inducing a Bloom-based cognitive route graph from exercise text and concept metadata.

Text-conditioned Bloom node initialization. Given exercise text x_e and concept metadata q_e, CRI first builds an exercise representation that combines semantic information from the text with concept-level information from metadata:

$$\mathbf{h}_e = \mathrm{Enc}_{\mathrm{txt}}(x_e), \quad \mathbf{c}_e = W_q q_e, \quad \mathbf{u}_e = [\mathbf{h}_e; \mathbf{c}_e], \tag{2}$$

Here, $\mathbf{h}_e$ encodes the exercise semantics, $\mathbf{c}_e$ projects the concept metadata into the representation space, and $\mathbf{u}_e$ denotes the combined exercise representation.

CRI then conditions each Bloom process on the current exercise. This step converts the fixed Bloom vocabulary into exercise-specific candidate cognitive operations:

$$\mathbf{r}_i^{(0)} = \phi(W_r[\mathbf{p}_i; \mathbf{u}_e] + b_r), \quad i \in V_B, \tag{3}$$

In this equation, $\mathbf{p}_i$ is the initial embedding of Bloom process i, and $\mathbf{r}_i^{(0)}$ is its exercise-conditioned node state. Thus, the same Bloom process can have different representations for different exercises.

Directed route induction. To model prerequisite relations among candidate cognitive operations, CRI scores a directed edge between every pair of Bloom processes:

$$a_{ij} = \sigma\left(\mathbf{w}_a^\top [\mathbf{r}_i^{(0)}; \mathbf{r}_j^{(0)}; \mathbf{u}_e] + b_a\right), \tag{4}$$

$$\alpha_{ij} = \frac{\exp(a_{ij})}{\sum_{k\neq j} \exp(a_{kj})}, \tag{5}$$

Here, a_{ij} estimates whether process i is a prerequisite for process j in the current exercise. The normalized coefficient α_{ij} represents the relative strength of process i as an incoming prerequisite of process j.

Based on these directed edge weights, CRI lets Bloom processes exchange information through prerequisite-aware message passing:

$$\mathbf{m}_j^{(t)} = \sum_{i \neq j} \alpha_{ij} W_m \mathbf{r}_i^{(t)}, \tag{6}$$

$$\mathbf{r}_j^{(t+1)} = \mathrm{GRU}\left(\mathbf{r}_j^{(t)}, \mathbf{m}_j^{(t)}\right), \tag{7}$$

where $t = 0, \ldots, T-1$. The update of process j relies on two sources: its previous state $\mathbf{r}_j^{(t)}$ and the incoming prerequisite message $\mathbf{m}_j^{(t)}$ from other candidate processes. The GRU combines these two sources to refine each Bloom-node representation according to the exercise-specific route structure.

After T propagation layers, CRI predicts the final node and edge structure of the cognitive route graph. The node predictor estimates whether each Bloom process is activated by the exercise:

$$\hat{z}_i = \sigma\left(\mathbf{w}_z^\top [\mathbf{r}_i^{(T)}; \mathbf{u}_e] + b_z\right), \tag{8}$$

Here, $\hat{z}_i \in [0, 1]$ denotes the activation probability of Bloom process i.

The edge predictor estimates whether a directed prerequisite relation should be selected between two Bloom processes:

$$\hat{A}_{ij} = \sigma\left(\mathbf{w}_A^\top [\mathbf{r}_i^{(T)}; \mathbf{r}_j^{(T)}; \mathbf{u}_e] + b_A\right), \tag{9}$$

Here, $\hat{A}_{ij} \in [0, 1]$ denotes the probability that process i is a prerequisite for process j. The selected nodes and edges form the cognitive route graph $\mathcal{G}_e$. At inference time, CRI predicts activated Bloom processes and prerequisite edges without calling the LLM.

LLM-to-graph supervision. Standard response datasets provide student answers and concept labels, but not cognitive route annotations. To obtain scalable supervision, we use an LLM teacher only during training. For each exercise, the teacher returns candidate Bloom nodes, directed prerequisite edges, confidence scores, and evidence spans. We query the teacher M times with shuffled Bloom label orders and paraphrased instructions, aggregate the outputs by agreement, and convert them into weighted node- and edge-level supervision. The formal agreement aggregation and loss definitions are provided in Appendix C, while prompt templates, multi-round querying, evidence-span handling, and dataset-specific annotation details are provided in Appendix F. At inference time, the LLM teacher is removed entirely.

Graph regularization. To encourage concise and valid prerequisite routes, CRI applies sparsity and acyclicity regularization to the predicted adjacency matrix. The details of the regularization terms are provided in Appendix D.

3.3 Design: RouteCD — Route-Aware Cognitive Diagnosis

As shown in the Design part of Figure 2, RouteCD converts the CRI-induced route $\mathcal{G}_e$ into response prediction through three steps: exercise demand construction, student state estimation, and cognitive gap modeling.

Concept-by-process demand. RouteCD first projects the induced route into a concept-by-process demand matrix $\mathbf{D}_e \in \mathbb{R}^{|K| \times |V_B|}$:

$$\mathbf{r}_e = \sum_i \frac{\hat{z}_i}{\sum_k \hat{z}_k + \epsilon} \mathbf{r}_i^{(T)}, \tag{10}$$

$$\mathbf{D}_e = \sigma\left(\mathrm{reshape}_{|K| \times |V_B|}(W_D[\mathbf{r}_e; \mathbf{u}_e] + b_D)\right), \tag{11}$$

Here, $\mathbf{r}_e$ aggregates final Bloom-node states according to their activation probabilities, and $\mathbf{D}_e[k, v]$ estimates how strongly exercise e requires Bloom process v under concept k.

Student cognitive state. Given a student's response history, RouteCD estimates a student state matrix $\mathbf{S}_s \in [0,1]^{|K|\times|V_B|}$ in the same concept-by-process space:

$$\mathbf{h}_s = \mathrm{HistEnc}(\{[\mathbf{D}_{e_t}; y_{s,e_t}]\}_{t<T}), \tag{12}$$
$$\boldsymbol{\mu}_s = W_\mu \mathbf{h}_s, \tag{13}$$
$$\log \boldsymbol{\sigma}_s^2 = W_\sigma \mathbf{h}_s, \tag{14}$$
$$\mathbf{z}_s = \boldsymbol{\mu}_s + \boldsymbol{\sigma}_s \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, I), \tag{15}$$
$$\mathbf{S}_s = \mathrm{sigmoid}\Big(\mathrm{reshape}_{|K|\times|V_B|}\left(W_S \mathbf{z}_s + b_S\right)\Big), \tag{16}$$

The history encoder observes not only whether past exercises were answered correctly, but also which concept-by-process demands those exercises imposed. The variational parameterization allows the model to represent uncertainty in the inferred student state.

Cognitive gap and response prediction. RouteCD compares exercise demand and student state by computing:

$$\mathbf{G}_{s,e} = \mathbf{D}_e - \mathbf{S}_s, \tag{17}$$
$$\mathbf{g}_{s,e} = \mathrm{Pool}([\mathbf{G}_{s,e}; \mathbf{D}_e; \mathbf{S}_s]), \tag{18}$$

Positive entries in $\mathbf{G}_{s,e}$ indicate concept-process positions where exercise demand exceeds estimated student mastery, which provides a reasoning-breakdown hypothesis.

The pooled gap representation is then used for response prediction:

$$\hat{y}_{s,e} = \mathrm{sigmoid}(\mathrm{MLP}_{pred}([\mathbf{g}_{s,e}; \mathbf{h}_e; \mathbf{h}_s])), \tag{19}$$

Thus, RouteCD outputs both the response probability $\hat{y}_{s,e}$ and the interpretable cognitive gap $\mathbf{G}_{s,e}$.

3.4 Optimize: Training Objective

The model is trained with response prediction, route supervision, graph regularization, and variational regularization:

$$\begin{aligned}\mathcal{L} =& \mathcal{L}_{resp} + \lambda_z \mathcal{L}_{node} + \lambda_A \mathcal{L}_{edge} \\ &+ \lambda_d \mathcal{L}_{dag} + \lambda_s \mathcal{L}_{sparse} + \lambda_{KL} \mathcal{L}_{KL}.\end{aligned} \tag{20}$$

Here, $\mathcal{L}_{resp}$ trains response prediction, $\mathcal{L}_{node}$ and $\mathcal{L}_{edge}$ supervise route induction, $\mathcal{L}_{dag}$ and $\mathcal{L}_{sparse}$ regularize the induced graph, and $\mathcal{L}_{KL}$ regularizes the variational student state. Detailed loss definitions are provided in Appendix E. The full training and inference procedure is summarized in Algorithm 1 in Appendix A. During inference, the LLM teacher is removed entirely.

4 Experiments

Dataset	#Students	#Exercises	#Concepts	#Interactions
FrcSub	536	20	8	10,720
ASSIST	2,493	17,671	123	267,415
DBE-KT22	1,101	212	93	157,557
MOOC	11,020	2,488	190	884,132

Table 1: Dataset statistics.

We evaluate RouteCD along four axes: **(Q1)** response prediction against classical, neural, cognition-aware, and language-enhanced cognitive diagnosis baselines; **(Q2)** component contributions via ablation; **(Q3)** sensitivity to text encoders and LLM teachers; and **(Q4)** data efficiency, hyperparameter robustness, and cross-domain generalisation. Overall, the experiments examine whether structured cognitive routes improve prediction, whether each component contributes to performance, and whether RouteCD remains robust across model and data settings.

Model	FrcSub			ASSIST			DBE-KT22			MOOC		
	AUC↑	ACC↑	RMSE↓	AUC↑	ACC↑	RMSE↓	AUC↑	ACC↑	RMSE↓	AUC↑	ACC↑	RMSE↓
IRT	0.8398	0.7696	0.4038	0.7417	0.7100	0.4468	0.8098	0.7966	**0.3733**	0.6893	0.8133	0.3772
MIRT	0.8554	0.7826	0.3971	0.6604	0.6440	0.5684	0.6875	0.7247	0.4849	0.5364	0.5397	0.6610
DINA	0.7280	0.6250	0.5089	0.6958	0.6381	0.4905	0.7177	0.6153	0.4636	0.6804	0.6745	0.4395
NCD	0.8501	0.7920	0.4004	0.7528	**0.7312**	0.4355	0.7929	0.7912	0.3808	0.6806	0.8305	0.3672
AGCDM	0.8664	0.8022	0.3900	0.7651	0.7265	0.4270	0.8079	0.7975	0.3751	0.7613	0.8266	0.3614
RCD	0.8041	0.8024	0.3812	**0.7680**	0.7290	**0.4247**	0.8086	0.7970	0.3741	0.5174	0.4569	0.7190
CACD	0.8528	0.7920	0.3918	0.7563	0.7271	0.4277	0.8041	0.7927	0.3775	0.7730	0.8340	0.3506
CCD	0.8695	0.7938	0.3850	0.7565	0.7216	0.4332	0.8050	0.7973	0.3738	0.7759	0.8336	0.3499
EduEmbed	0.8719	0.8025	0.3811	0.7561	0.7179	0.4301	0.8056	0.7978	0.3745	0.7516	0.8256	0.3717
LLM emb.+head	0.8701	0.8012	0.3810	0.7514	0.7213	0.4345	0.8068	0.7960	0.3765	0.7705	0.8339	0.3519
RouteCD (ours)	**0.8862**	**0.8207**	**0.3690**	**0.7680**	0.7296	0.4251	**0.8099**	**0.7983**	0.3740	**0.7768**	**0.8351**	**0.3498**

Table 2: Main response-prediction results. **Bold**: best per column; underline: second best. Row shading: classical , neural CD , text-enhanced , 2026 SOTA , ours .

4.1 Experimental Setup

Datasets. We use four real-world response-prediction datasets with different scales and levels of text richness: FrcSub[1], ASSISTments0910[2], DBE-KT22[3], and MOOC[4]. FrcSub (DeCarlo, 2011) is a small fraction-subtraction dataset with expert Q-matrix annotations; ASSISTments0910 (Corbett & Anderson, 2001) provides large-scale student response logs from an online tutoring system; DBE-KT22 (Abdelrahman et al., 2022) is a text-rich SQL exercise benchmark; and MOOC (Huang et al., 2020) is a large-scale online-course dataset.

We apply dataset-specific preprocessing: for FrcSub, ASSISTments0910, and DBE-KT22, we remove students with fewer than 15 interactions; for MOOC, we retain exercises appearing in student interactions, remove concepts with frequency below 5, and keep students with at least 10 interactions. All datasets are split by student into training, validation, and test sets with a 7:1:2 ratio, and results are averaged over five random seeds. Dataset statistics are reported in Table 1.

Baselines and Metrics. We compare RouteCD with representative baselines from four categories: classical cognitive diagnosis models, including IRT (Lord, 1980), MIRT (Reckase, 2009), and DINA (de la Torre, 2009); neural and graph-based cognitive diagnosis models, including NCD (Wang et al., 2020a), AGCDM (Pei et al., 2022), RCD (Gao et al., 2021), and CACD (Wang et al., 2024); cognition-aware cognitive diagnosis, including CCD (Gu et al., 2025); and language-enhanced cognitive diagnosis, including EduEmbed (Liu et al., 2026). Unlike embedding-based semantic baselines, RouteCD converts exercise text into explicit Bloom-based route graphs for route-aware diagnosis. All baselines are rerun under the same preprocessing, data splits, random seeds, and evaluation protocol.

Following prior cognitive diagnosis studies (Gao et al., 2021; Wang et al., 2024), we evaluate the overall diagnostic performance through student response prediction. We report AUC (Fawcett, 2006), ACC (Powers, 2011), and RMSE (Chai & Draxler, 2014). Higher AUC and ACC indicate better classification performance, while lower RMSE indicates smaller discrepancy between predicted and observed response scores.

Implementation details. Unless otherwise specified, exercise text is encoded with all-MiniLM-L6-v2 (Reimers & Gurevych, 2019; Wang et al., 2020b) as the default text encoder (see Section 4.4 for a comparison of frozen and fine-tuned variants, as well as alternative encoders). The route GNN uses $T=2$ message-passing layers, and the student profile is modeled by a one-layer GRU with variational reparameterization. We set $\tau_z=\tau_A=0.5$ and use $M=5$ LLM annotation rounds. The loss coefficients are $\lambda_z=\lambda_A=0.5$, $\lambda_d=0.1$, $\lambda_s=0.01$, and $\lambda_{KL}=0.001$. Models are optimized with Adam (Kingma & Ba, 2015) using a learning rate of 5×10^{-4} and batch size 256. All experiments are conducted on a server with two NVIDIA A100-SXM4-40GB GPUs.

[1] https://wenchao-ma.github.io/GDINA/reference/frac20.html

[2] https://sites.google.com/site/assistmentsdata/home/2009-2010-assistment-data/skill-builder-data-2009-2010

[3] https://dataverse.ada.edu.au/dataset.xhtml?persistentId=doi:10.26193/6DZWOH

[4] https://bhpan.buaa.edu.cn/link/AAE5A458AFE444474EBEC928F98986E5B8

Variant	Text Enc.	CRI Route	LLM Sup.	AUC	ACC	RMSE	ΔAUC
Base CD backbone	✗	✗	✗	0.7826	0.7864	0.3961	−0.0273
Text only	✓	✗	✗	0.7952	0.7920	0.3846	−0.0147
CRI only	✗	✓	✗	0.7983	0.7939	0.3822	−0.0116
LLM sup. only	✗	✗	✓	0.7969	0.7933	0.3839	−0.0130
Text + CRI	✓	✓	✗	0.8095	0.7978	0.3787	−0.0004
Text + LLM sup.	✓	✗	✓	0.8056	0.7965	0.3801	−0.0043
CRI + LLM sup.	✗	✓	✓	0.8058	0.7967	0.3818	−0.0041
RouteCD (full)	✓	✓	✓	**0.8099**	**0.7983**	**0.3740**	—

Table 3: Ablation study on DBE-KT22. ✓/✗indicates whether each key component is used. When CRI Route is absent, LLM Sup. is applied as auxiliary semantic supervision to the prediction head. ΔAUC is computed as $\text{AUC}_{\text{variant}} - \text{AUC}_{\text{full}}$.

4.2 Main Response-Prediction Results (Q1)

Table 2 reports AUC, ACC, and RMSE on the four datasets, and Figure 3 visualizes the accuracy–error landscape.

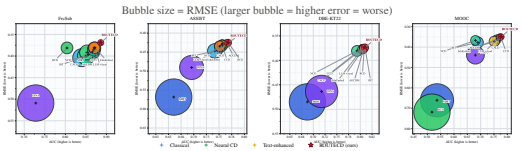


Figure 3: Accuracy–error landscape across four datasets. Each panel plots AUC (x-axis, higher is better) against RMSE (y-axis, lower is better); bubble size also encodes RMSE, so a smaller, higher-right bubble is better. RouteCD (red star) achieves the smallest RMSE and among the highest AUC on every dataset.

Overall performance. RouteCD achieves the best or competitive response-prediction performance across the four datasets, consistently outperforming classical, neural, cognition-aware, and language-enhanced baselines on most metrics. The gains are more evident on text-rich datasets such as DBE-KT22 and MOOC, where exercise descriptions provide useful evidence for inducing structured Bloom route graphs. On FrcSub, RouteCD achieves a clear improvement despite limited item coverage, suggesting that route induction provides useful cognitive signals even in small-scale settings. On ASSISTments0910, the gains are relatively smaller because exercise text is limited, reducing the benefit of text-based route induction. These results suggest that the advantage of RouteCD does not come merely from adding semantic embeddings. Although language-enhanced baselines can improve response prediction by incorporating exercise semantics, they still compress exercise information into dense representations. In contrast, RouteCD organizes exercise semantics into explicit Bloom-based route graphs, modeling both the required cognitive operations and their prerequisite dependencies. This structured representation provides a more informative basis for response prediction than concept labels or dense embeddings alone.

4.3 Ablation Study (Q2)

Table 3 isolates the contributions of three key components on DBE-KT22: the text encoder, the CRI route module, and LLM-derived supervision. The full visualization is provided in Appendix G.1. The base CD backbone obtains an AUC of 0.7826 and performs worst among the variants, while adding any single component improves performance. Among single-component variants, the CRI-only model yields the largest gain, suggesting that route-based cognitive structure provides useful diagnostic signal beyond the backbone. Combining text encoding with CRI further improves AUC to 0.8095 and substantially improves ACC and RMSE over the backbone. The complete RouteCD achieves the best AUC and ACC, confirming that text encoding, route induction, and LLM-derived supervision are complementary. Overall, these results support the core design of RouteCD: response prediction benefits from explicit Bloom-level route modeling rather than relying only on concept labels or dense embeddings.

Node-versus-edge decomposition. The ablation above isolates text encoding, route induction, and LLM-derived supervision as three components, but does not by itself show whether the gain from CRI comes from activating Bloom-process nodes, from the prerequisite edges between them, or both. Table 4 decomposes the CRI route module itself on DBE-KT22. Adding Bloom-process nodes to the text-only backbone improves AUC from 0.7952 to 0.8001 and reduces RMSE from 0.3846 to 0.3783, while ACC decreases slightly. Further adding prerequisite edges on top of the

Variant	Bloom Nodes	Prerequisite Edges	AUC↑	ACC↑	RMSE↓
Text-only backbone	✗	✗	0.7952	0.7920	0.3846
CRI nodes only	✓	✗	0.8001	0.7906	0.3783
CRI nodes + prerequisite edges	✓	✓	**0.8035**	**0.7928**	**0.3769**

Table 4: Controlled decomposition of Bloom-process nodes and prerequisite edges on DBE-KT22.

node-only variant improves ACC from 0.7906 to 0.7928, reduces RMSE to 0.3769, and increases AUC to 0.8035, achieving the best mean performance across all three metrics. This confirms that prerequisite-edge modeling contributes gains beyond node activation alone, supporting the design choice of representing cognitive routes as directed graphs rather than unordered node sets.

4.4 Encoder and Teacher Sensitivity (Q3)

Text encoder comparison. Figure 9 (Appendix G) compares different text encoders on DBE-KT22 and FrcSub, including frozen and fine-tuned all-MiniLM-L6-v2 (Reimers & Gurevych, 2019; Wang et al., 2020b), DeBERTa-v3-base (He et al., 2023), BGE-small-zh-v1.5 (Xiao et al., 2023), and TF-IDF+SVD (Salton & Buckley, 1988; Deerwester et al., 1990). Across encoder choices, RouteCD consistently outperforms CCD, indicating that its gains are not tied to a specific encoder backbone. Fine-tuned all-MiniLM-L6-v2 achieves the best AUC on both DBE-KT22 and FrcSub, reaching 0.8099 and 0.8862, respectively, despite using only 22M parameters. Larger encoders such as DeBERTa-v3-base do not necessarily improve performance, suggesting that encoder capacity alone is not the dominant factor. On DBE-KT22, both the GNN variant without LLM supervision and the LLM emb.+head baseline remain below the full RouteCD model. This indicates that semantic embeddings alone are insufficient, while LLM-derived supervision helps learn more reliable cognitive route structures.

LLM teacher sensitivity. Figure 10 (Appendix G) evaluates the effect of teacher choice and annotation rounds M on DBE-KT22. All LLM-supervised settings outperform the no-LLM backbone, which obtains 0.7826 AUC. With GPT-4o as the teacher, AUC improves from 0.8042 at $M{=}1$ to 0.8099 at $M{=}5$, but further increasing M to 10 yields only a marginal gain to 0.8102. This suggests that agreement aggregation is useful, but its benefit saturates after a moderate number of annotation rounds. Teacher quality also affects performance. At $M{=}5$, GPT-4o achieves the best AUC of 0.8099, outperforming Claude-3.5-Sonnet (0.8079), GPT-4o-mini (0.8057), and GPT-3.5-Turbo (0.8033). These results suggest that a stronger teacher with moderate agreement aggregation is more effective than simply increasing annotation rounds with a weaker teacher.

Figure 7 (Appendix G) shows that with only 30% training data, RouteCD (AUC about 0.8000) already outperforms NCD trained on 100% data, and with 70% it surpasses full-data CCD, CACD, and NCD, reaching 0.8099 with all data.

Hyperparameter robustness. Figure 8 (Appendix G) shows that $T{=}2$ layers give the best trade-off between AUC (0.8099) and training time, and $\lambda_z{=}0.5$ is optimal; removing LLM supervision ($\lambda_z{=}0$) drops AUC to 0.7969. Further analyses are provided in Appendix G.

5 Conclusion

This paper presents RouteCD, a route-aware cognitive diagnosis framework that induces Bloom-based cognitive route graphs from exercise text and projects them into a shared concept-by-process space. By comparing exercise demand with student cognitive state, RouteCD supports both response prediction and reasoning-breakdown diagnosis. Experiments on four datasets show that RouteCD achieves the best or competitive performance against classical, neural, graph-based, cognition-aware, and language-enhanced baselines. Ablation and sensitivity analyses confirm the value of explicit route induction and LLM-derived graph supervision, while inference requires no LLM calls. Overall, our results suggest that cognitive diagnosis should model not only *which concepts* are involved, but also *which cognitive operations* are required and *how* they depend on one another.

AI USE STATEMENT

We have not used generative AI tools for any of the tasks requiring mandatory disclosure under the ICLR 2027 AI Policy for Authors (e.g., generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating or proving mathematical claims, proposing or refining hypotheses, designing research methodology or experiments, implementing methods, translation, dataset cleaning/reformatting, or interpreting results); all such work was performed solely by the authors. We used generative AI tools for tasks under the recommended-disclosure category, specifically to draft parts of the manuscript text and to edit the manuscript for readability, consistency, and correctness (e.g., proofreading, and reconciling inconsistencies between the narrative text and reported tables/figures). All AI-assisted text was reviewed and verified by the authors against the underlying experimental results before inclusion. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Ghodai Abdelrahman, Sherif Abdelfattah, Qing Wang, and Yu Lin. DBE-KT22: A knowledge tracing dataset based on online student evaluation. *arXiv preprint arXiv:2208.12651*, 2022. doi: 10.48550/arXiv.2208.12651.

Lorin W. Anderson, David R. Krathwohl, Peter W. Airasian, Kathleen A. Cruikshank, Richard E. Mayer, Paul R. Pintrich, James Raths, and Merlin C. Wittrock. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. Longman, New York, 2001.

Tianfeng Chai and Roland R. Draxler. Root mean square error (rmse) or mean absolute error (mae)? – arguments against avoiding rmse in the literature. *Geoscientific Model Development*, 7 (3):1247–1250, 2014. doi: 10.5194/gmd-7-1247-2014.

Albert T. Corbett and John R. Anderson. Locus of feedback control in computer-based tutoring: Impact on learning rate, achievement and attitudes. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 245–252, 2001. doi: 10.1145/365024.365111.

Jimmy de la Torre. DINA model and parameter estimation: A didactic. *Journal of Educational and Behavioral Statistics*, 34(1):115–130, 2009. doi: 10.3102/1076998607309474.

Lawrence T. DeCarlo. On the analysis of fraction subtraction data: The DINA model, classification, latent class sizes, and the q-matrix. *Applied Psychological Measurement*, 35(1):8–26, 2011. doi: 10.1177/0146621610377081.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41 (6):391–407, 1990. doi: 10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9.

Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006. doi: 10.1016/j.patrec.2005.10.010.

Weibo Gao, Qi Liu, Zhenya Huang, Yu Yin, Haoyang Bi, Mu-Chun Wang, Jianhui Ma, Shijin Wang, and Yu Su. RCD: Relation map driven cognitive diagnosis for intelligent education systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 501–510, 2021. doi: 10.1145/3404835.3462932.

Hengnian Gu, Guoqian Luo, Xiaoxiao Dong, Shulin Li, and Dongdai Zhou. Revisiting cognition in neural cognitive diagnosis. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 402–412, 2025. doi: 10.1145/3690624.3709319.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. In *International Conference on Learning Representations*, 2023.

Zhenya Huang, Qi Liu, Yuying Chen, Le Wu, Keli Xiao, Enhong Chen, Haiping Ma, and Guoping Hu. Learning or forgetting? a dynamic approach for tracking the knowledge proficiency of students. *ACM Transactions on Information Systems*, 38(2):1–33, 2020. doi: 10.1145/3379507.

Kevin Hwang, Kenneth Wang, Maryam Alomair, Fow-Sen Choa, and Lujie Karen Chen. Towards automated multiple choice question generation and evaluation: Aligning with Bloom's taxonomy. In *Artificial Intelligence in Education*, volume 14830 of *Lecture Notes in Computer Science*, pp. 389–396, 2024. doi: 10.1007/978-3-031-64299-9_35.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017. doi: 10.18653/v1/P17-1014.

David R. Krathwohl. A revision of Bloom's taxonomy: An overview. *Theory Into Practice*, 41(4): 212–218, 2002. doi: 10.1207/s15430421tip4104_2.

Amar Lalwani and Sweety Agrawal. Validating revised bloom's taxonomy using deep knowledge tracing. In *Artificial Intelligence in Education*, pp. 225–238. Springer, 2018. doi: 10.1007/978-3-319-93843-1_17.

Tori M. Larsen, Bianca H. Endo, Alexander T. Yee, Trang Do, and Stanley M. Lo. Probing internal assumptions of the revised Bloom's taxonomy. *CBE—Life Sciences Education*, 21(4):ar66, 2022. doi: 10.1187/cbe.20-08-0170.

Jacqueline P. Leighton, Mark J. Gierl, and Steve M. Hunka. The attribute hierarchy method for cognitive assessment: A variation on tatsuoka's rule-space approach. *Journal of Educational Measurement*, 41(3):205–237, 2004. doi: 10.1111/j.1745-3984.2004.tb01163.x.

Yuheng Li, Mladen Raković, Boon Xin Poh, Dragan Gašević, and Guanliang Chen. Automatic classification of learning objectives based on Bloom's taxonomy. In *Proceedings of the 15th International Conference on Educational Data Mining*, pp. 530–537, 2022. doi: 10.5281/zenodo.6853191.

Yuanhao Liu, Zihan Zhou, Kaiying Wu, Shuo Liu, Yiyang Huang, Jiajun Guo, Aimin Zhou, and Hong Qian. Embedding enhancement via fine-tuned language models for learner-item cognitive modeling. In *Proceedings of the ACM Web Conference 2026*, pp. 6517–6528, 2026. doi: 10.1145/3774904.3792542. URL https://github.com/BW297/EduEmbed.

Frederic M. Lord. *Applications of Item Response Theory to Practical Testing Problems*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1980.

Chunchuan Lyu and Ivan Titov. AMR parsing as graph prediction with latent alignment. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018. doi: 10.18653/v1/P18-1037.

Xiaohuan Pei, Shuo Yang, Jiajun Huang, and Chang Xu. Self-attention gated cognitive diagnosis for faster adaptive educational assessments. In *2022 IEEE International Conference on Data Mining*, ICDM, pp. 408–417, 2022. doi: 10.1109/ICDM54844.2022.00051.

David M. W. Powers. Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.

Mark D. Reckase. *Multidimensional Item Response Theory*. Springer, New York, NY, 2009. doi: 10.1007/978-0-387-89976-3.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pp. 3982–3992, 2019. doi: 10.18653/v1/D19-1410.

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988. doi: 10.1016/0306-4573(88)90021-0.

Dongbo Tu, Shiyu Wang, Yan Cai, Jeff Douglas, and Hua Hua Chang. Cognitive diagnostic models with attribute hierarchies: Model estimation with a restricted q-matrix design. *Applied Psychological Measurement*, 43(4):255–271, 2019. doi: 10.1177/0146621618765721.

Fei Wang, Qi Liu, Enhong Chen, Zhenya Huang, Yuying Chen, Yu Yin, Zai Huang, and Shijin Wang. Neural cognitive diagnosis for intelligent education systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 6153–6161, 2020a. doi: 10.1609/aaai.v34i04.6080.

Shanshan Wang, Zhen Zeng, Xun Yang, Ke Xu, and Xingyi Zhang. Boosting neural cognitive diagnosis with student's affective state modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 620–627, 2024. doi: 10.1609/aaai.v38i1.27818.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5776–5788, 2020b.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding. *arXiv preprint arXiv:2309.07597*, 2023.

Algorithm 1 ROUTECD training procedure. Stages 1–2 are also used at inference time, while Stage 3 and the LLM teacher are used only during training.

Require: Responses $\mathcal{D}$; exercise texts $\{x_e\}$; concept metadata $\{q_e\}$; LLM agreement scores $\{\omega_i^e, \omega_{ij}^e\}$; coefficients $\lambda_z, \lambda_A, \lambda_d, \lambda_s, \lambda_{KL}$.
Ensure: Trained parameters $\Theta = \{\theta_{\text{CRI}}, \theta_{\text{RouteCD}}\}$.

for each mini-batch $\mathcal{B} \subseteq \mathcal{D}$ **do**
  Stage 1: CRI route induction [Inf]
  Encode exercise: $h_e = \text{Enc}_{\text{txt}}(x_e), \quad u_e = [h_e; W_q q_e]$
  Initialize Bloom nodes: $r_i^{(0)} = \phi(W_r[p_i; u_e] + b_r), \ \forall i \in V_B$
  Score prerequisite edges: $a_{ij} = \sigma(w_a^\top [r_i^{(0)}; r_j^{(0)}; u_e] + b_a)$
  Normalize incoming edge weights: $\alpha_{ij} = \exp(a_{ij}) / \sum_{k \neq j} \exp(a_{kj})$
  for $t = 0, \ldots, T-1$ **do**
    $m_j^{(t)} = \sum_{i \neq j} \alpha_{ij} W_m r_i^{(t)}$
    $r_j^{(t+1)} = \text{GRU}(r_j^{(t)}, m_j^{(t)})$
  end for
  Predict route: $\hat{z}_i = \sigma(w_z^\top [r_i^{(T)}; u_e] + b_z)$
  $\hat{A}_{ij} = \sigma(w_A^\top [r_i^{(T)}; r_j^{(T)}; u_e] + b_A) \cdot \mathbf{1}[i \neq j]$
  Stage 2: Route-aware diagnosis [Inf]
  $r_e = \sum_i \hat{z}_i r_i^{(T)} / (\sum_k \hat{z}_k + \epsilon)$
  $D_e = \sigma(\text{reshape}(W_D[r_e; u_e] + b_D))$
  $h_s = \text{HistEnc}(\{[D_{e_t}; y_{s,e_t}]\}_{t<T})$
  $\mu_s = W_\mu h_s, \quad \log \sigma_s^2 = W_\sigma h_s$
  $z_s = \mu_s + \sigma_s \odot \epsilon_z, \quad \epsilon_z \sim \mathcal{N}(0, I)$
  $S_s = \sigma(\text{reshape}(W_S z_s + b_S))$
  $G_{s,e} = D_e - S_s$
  $\hat{y}_{s,e} = \sigma(\text{MLP}_{pred}([\text{Pool}([G_{s,e}; D_e; S_s]); h_e; h_s]))$
  Stage 3: Training losses
  Compute $\mathcal{L}_{node}$ and $\mathcal{L}_{edge}$ from LLM agreement scores.
  Compute $\mathcal{L}_{dag}$ and $\mathcal{L}_{sparse}$ from $\hat{A}_e$.
  Compute $\mathcal{L}_{resp}$ and $\mathcal{L}_{KL}$.
  $\mathcal{L} = \mathcal{L}_{resp} + \lambda_z \mathcal{L}_{node} + \lambda_A \mathcal{L}_{edge} + \lambda_d \mathcal{L}_{dag} + \lambda_s \mathcal{L}_{sparse} + \lambda_{KL} \mathcal{L}_{KL}$
  Update Θ by gradient descent on $\mathcal{L}$.
end for
return Θ

APPENDIX OVERVIEW

This appendix provides supplementary details and analyses for ROUTECD. Appendices A–E give implementation and methodological details, including the training algorithm, the non-chain route formulation, LLM-to-graph supervision, graph regularization, and loss definitions. Appendix F describes the LLM annotation pipeline, including prompt templates, multi-round querying, evidence-span aggregation, text encoder variants, and dataset-specific adaptations. Appendix G reports additional experimental analyses, including ablation visualization, route-quality analysis, Bloom-level demand patterns, cold-start performance, cross-domain CRI evaluation, and qualitative diagnostic case studies.

A ROUTECD TRAINING ALGORITHM

Algorithm 1 summarizes the training procedure of ROUTECD. Stages 1–2 are also used during inference. At inference time, the LLM teacher and all training losses are removed, and only the learned route inducer, student profile, and prediction head are executed.

B NON-CHAIN COGNITIVE ROUTES

In CRI, the predicted prerequisite matrix $\hat{A}_e$ is not constrained to follow the fixed Bloom order Remember $\rightarrow$ Understand $\rightarrow$ Apply $\rightarrow$ Analyze $\rightarrow$ Evaluate $\rightarrow$ Create. Instead, we treat Bloom-level dependencies as exercise-specific directed relations.

This design is motivated by two considerations. First, the revised Bloom's taxonomy represents learning objectives through the interaction between a knowledge dimension and a cognitive-process dimension, rather than enforcing a strict cumulative hierarchy among cognitive processes (Anderson et al., 2001; Krathwohl, 2002; Larsen et al., 2022). For example, an exercise may require applying a known procedure without explicitly involving evaluation or creation, or it may require analysis based on a small set of directly provided facts. Therefore, forcing every route to follow a global Bloom chain would introduce unnecessary intermediate operations.

Second, cognitive diagnosis research often models prerequisite relations among attributes or knowledge components as local structures rather than a single universal order (Leighton et al., 2004; Tu et al., 2019). The same principle applies to cognitive processes: whether one Bloom process serves as a prerequisite for another depends on the wording, context, and required solution procedure of a specific exercise. Accordingly, CRI allows routes to skip lower-level processes, combine multiple processes, and form local directed dependencies that better reflect exercise-level cognitive demand.

C LLM-TO-GRAPH SUPERVISION

Standard response datasets usually provide student answers and concept labels, but do not contain annotations for Bloom-process activations, prerequisite dependencies, or supporting evidence spans. To obtain scalable supervision for CRI, we use an LLM teacher only during training. Given an exercise, the teacher returns candidate Bloom nodes, directed prerequisite edges, confidence scores, and evidence spans.

Because a single LLM annotation may be noisy or prompt-sensitive, we query the teacher M times with shuffled Bloom label orders and paraphrased instructions. Let $Z_e^{(m)}$ and $E_e^{(m)}$ denote the Bloom nodes and directed edges selected in the m-th query. We aggregate the annotations by agreement:

$$\begin{aligned} \omega_i &= \frac{1}{M}\sum_{m=1}^{M} \mathbf{1}[i \in Z_e^{(m)}], \\ \omega_{ij} &= \frac{1}{M}\sum_{m=1}^{M} \mathbf{1}[(i,j) \in E_e^{(m)}], \end{aligned} \tag{21}$$

Here, ω_i and ω_{ij} measure how consistently node i and edge (i,j) are selected across LLM annotations.

The agreement scores are converted into binary weak labels using fixed thresholds τ_z and τ_A:

$$\begin{aligned} \tilde{z}_i &= \mathbf{1}[\omega_i > \tau_z], \\ \tilde{A}_{ij} &= \mathbf{1}[\omega_{ij} > \tau_A], \end{aligned} \tag{22}$$

Thus, a Bloom process or prerequisite edge is treated as positive only when it is selected consistently enough by the teacher.

To account for teacher reliability, each weak label is weighted by its agreement score:

$$\begin{aligned} w_i^z &= \tilde{z}_i\omega_i + (1-\tilde{z}_i)(1-\omega_i), \\ w_{ij}^A &= \tilde{A}_{ij}\omega_{ij} + (1-\tilde{A}_{ij})(1-\omega_{ij}), \end{aligned} \tag{23}$$

This weighting scheme gives larger weights to labels with stronger teacher agreement and down-weights ambiguous labels.

The node- and edge-level graph supervision losses are weighted binary cross-entropies:

$$\begin{aligned} \mathcal{L}_{node} &= -\sum_i w_i^z \left[\tilde{z}_i \log \hat{z}_i + (1-\tilde{z}_i)\log(1-\hat{z}_i)\right], \\ \mathcal{L}_{edge} &= -\sum_{i\neq j} w_{ij}^A \left[\tilde{A}_{ij} \log \hat{A}_{ij} + (1-\tilde{A}_{ij})\log(1-\hat{A}_{ij})\right], \end{aligned} \tag{24}$$

Here, $\mathcal{L}_{node}$ supervises Bloom-process activation, and $\mathcal{L}_{edge}$ supervises directed prerequisite-edge selection. Self-loops are excluded from edge supervision because a Bloom process is not treated as a prerequisite of itself.

The evidence spans returned by the LLM are used as training-time grounding evidence for positive nodes and edges. They make the weak supervision auditable, but they are not predicted at inference time. During inference, the LLM teacher is removed entirely, and CRI predicts $\hat{z}_e$ and $\hat{A}_e$ using only the learned neural route inducer.

D Graph Regularization Details

To encourage concise prerequisite routes, CRI regularizes the predicted adjacency matrix to be sparse and approximately acyclic. We first remove self-loops from the predicted adjacency matrix:

$$\bar{A}_e = \hat{A}_e \odot (\mathbf{1}\mathbf{1}^\top - I), \tag{25}$$

where I is the identity matrix and $\odot$ denotes element-wise multiplication.

We then apply a NOTEARS-style acyclicity penalty to discourage circular prerequisite relations:

$$\mathcal{L}_{dag} = \mathrm{tr}\big(\exp(\bar{A}_e \odot \bar{A}_e)\big) - |V_B|, \tag{26}$$

This penalty increases when cyclic dependencies are present and is minimized when the induced route is acyclic.

Finally, sparse edge regularization penalizes unnecessary edge mass:

$$\mathcal{L}_{sparse} = \sum_{i \neq j} \hat{A}_{ij}, \tag{27}$$

Together, $\mathcal{L}_{dag}$ and $\mathcal{L}_{sparse}$ encourage the induced cognitive route to remain interpretable, sparse, and diagnostically concise.

E Training Objective Details

The response prediction loss is binary cross-entropy:

$$\mathcal{L}_{resp} = -\sum_{(s,e)} \left[y_{s,e} \log \hat{y}_{s,e} + (1 - y_{s,e}) \log(1 - \hat{y}_{s,e}) \right], \tag{28}$$

Here, $y_{s,e} \in \{0, 1\}$ is the observed response label, and $\hat{y}_{s,e}$ is the predicted probability of a correct response.

The variational student state is regularized by:

$$\mathcal{L}_{KL} = D_{\mathrm{KL}}(q(\mathbf{z}_s \mid \mathcal{H}_s) \,\|\, \mathcal{N}(0, I)). \tag{29}$$

Here, $q(\mathbf{z}_s \mid \mathcal{H}_s)$ denotes the latent student-state distribution inferred from response history, and $\mathcal{N}(0, I)$ is the standard Gaussian prior.

The route supervision losses $\mathcal{L}_{node}$ and $\mathcal{L}_{edge}$ are defined in Appendix C, and the graph regularization terms $\mathcal{L}_{dag}$ and $\mathcal{L}_{sparse}$ are defined in Appendix D.

F LLM Annotation Pipeline Details

This appendix provides implementation details of the LLM annotation pipeline, including prompt templates, multi-round querying, evidence-span handling, exercise text encoder choices, and dataset-specific adaptations. The formal agreement-weighted supervision losses are defined in Appendix C. The LLM annotations are used only during training; inference relies solely on the learned route inducer.

F.1 Prompt Templates

For each exercise, the LLM teacher is queried M times with shuffled Bloom label orders and paraphrased instructions. Figure 4 shows the node activation prompt used for DBE-KT22. The

Node Activation Prompt (DBE-KT22 variant)

System: You are an expert in database education and Bloom's Revised Taxonomy. You assess database course exercises and determine which cognitive levels a student must engage to solve them correctly.

User:
Exercise `\{id\}`: `\{question\_title\}` + `\{question\_text\}`
Knowledge concepts required:

- `\{kc\_name\}`: `\{kc\_description\}` *(repeated per KC)*

Bloom's Taxonomy levels *(order shuffled across M annotation rounds to reduce order bias)*:

- **R** – Remember (recall key database concepts, definitions, and rules needed for this question)
- **U** – Understand (explain why a concept or operation applies in this context)
- **Ap** – Apply (correctly execute the required database operation or procedure)
- **An** – Analyze (identify which concepts are needed and how they interact)
- **E** – Evaluate (check and verify the correctness of an answer or approach)
- **C** – Create (construct a novel query, schema, or solution strategy)

Task: Assign a relevance score in [0.0, 1.0] to **each** of the 6 Bloom levels, reflecting how strongly that cognitive level is engaged when solving this exercise. Scores do NOT need to sum to 1 — each is independent.
Return the scores in the canonical order `R, U, Ap, An, E, C`, regardless of the order in which the Bloom levels are shown above. For each Bloom level with score > 0.5, also provide one or more short evidence spans copied from the exercise text that justify why this level is activated. If no clear evidence exists for a level, use an empty list.
Respond with ONLY a valid JSON object in the following format:

```
        \{"scores":\{"R":0.8,"U":0.5,"Ap":0.9,"An":0.3,"E":0.1,"C":0.0\},
\ "evidence\_spans":\{"R":["..."],"U":[],"Ap":["..."],"An":[],"E":[],"C":[]\}\}
```

No explanation, no markdown, just the JSON object.

`\{\{answer\}\}`

Figure 4: Node activation prompt template for DBE-KT22. The LLM outputs Bloom activation scores and supporting evidence spans. Placeholders in `\{braces\}` are filled per exercise, and Bloom level order is shuffled across M annotation rounds.

FrcSub variant replaces the knowledge-component block with fraction-subtraction skill definitions, such as α_1: "convert a whole number to a fraction". In all prompt variants, the returned scores are mapped back to the canonical Bloom order $\{\mathrm{R}, \mathrm{U}, \mathrm{Ap}, \mathrm{An}, \mathrm{E}, \mathrm{C}\}$.

For directed prerequisite edges, we use a separate prompt that asks the LLM to output a 6×6 adjacency matrix directly, as shown in Figure 5. This avoids deriving edge labels indirectly from normalized node scores, which may suppress useful dependency signals. The matrix rows and columns always follow the canonical Bloom order $\{\mathrm{R}, \mathrm{U}, \mathrm{Ap}, \mathrm{An}, \mathrm{E}, \mathrm{C}\}$.

F.2 Multi-round Querying and Evidence-Span Aggregation

We query the LLM $M{=}5$ times per exercise by default. Across rounds, we shuffle the order of Bloom labels and paraphrase the task instruction to reduce order bias and prompt-specific artifacts. For node annotation, each round returns Bloom-level relevance scores and supporting evidence spans. For edge annotation, each round returns a 6×6 directed adjacency matrix and supporting evidence spans for high-confidence prerequisite edges.

The outputs from the M rounds are aggregated by agreement. Specifically, node and edge scores are first converted into round-level selections using a score threshold of 0.5; the fraction of rounds in which each node or edge is selected is then used as its agreement score. Nodes and edges whose agreement scores exceed the corresponding thresholds τ_z and τ_A are retained as weak labels, and the agreement scores are used as reliability weights. The detailed definitions of agreement scores, weak labels, reliability weights, and graph supervision losses are provided in Appendix C.

Edge Annotation Prompt

System: [identical to node prompt]

User:

[same exercise text and KC / skill block as node prompt]

Task: For this specific exercise, determine the directed cognitive dependency between Bloom levels. Output a 6×6 JSON matrix M where $M[i][j]$ is a float in [0.0, 1.0] indicating how strongly the row-level Bloom process must be mastered *before* the column-level Bloom process when solving this exercise. Rows and columns must follow the canonical order `R, U, Ap, An, E, C`. Diagonal entries must be 0.0 (no self-loops).

For each directed edge with score > 0.5, also provide one or more short evidence spans copied from the exercise text that justify the prerequisite dependency. Use edge keys in the format `"R->Ap"`, `"Ap->An"`, etc. If no clear evidence exists for an edge, omit that edge from `edge\_evidence\_spans`.

Respond with ONLY a valid JSON object in the following format:

```
\{"matrix":[[0,0.8,0,0,0,0],[0,0,0.7,0,0,0],
 \ [0,0,0,0.6,0,0],[0,0,0,0,0.4,0],
 \ [0,0,0,0,0,0.2],[0,0,0,0,0,0]],
\ "edge\_evidence\_spans":\{"R->U":["..."],"U->Ap":["..."]\}\}
```

No explanation, no markdown, just the JSON object.

`\{\{answer\}\}`

Figure 5: Edge annotation prompt template. The LLM outputs a 6×6 directed adjacency matrix and supporting evidence spans for high-confidence prerequisite edges.

Dataset	Provider	Node	Edge	Node cons.%	Edge cons.%
FrcSub	GPT-4o	6	30	92.5	78.3
FrcSub	Claude-3.5-Sonnet	6	30	91.7	76.8
DBE-KT22	GPT-4o	6	30	88.1	71.4
DBE-KT22	Claude-3.5-Sonnet	6	30	87.6	70.9
ASSIST	GPT-4o	6	30	85.3	68.2
MOOC	GPT-4o	6	30	84.7	67.5

Table 5: LLM label consistency rates across datasets and providers. Node and edge consistency are computed over the six Bloom nodes and 30 off-diagonal directed edges, respectively.

Evidence spans are aggregated only for retained positive nodes and edges. For each retained node or edge, we collect spans from the rounds in which it is selected, remove duplicate or near-duplicate spans, and keep the resulting span set as training-time grounding evidence. These spans make the LLM-derived supervision auditable, but they are not predicted by CRI and are not required at inference time.

Table 5 reports the empirical consistency rates observed across datasets and providers.

F.3 Exercise Text Encoder Variants

We evaluate several text encoders for constructing the exercise representation $\mathbf{h}_e$ used by CRI. Table 6 summarizes their properties, and Figure 9 in the main paper reports their downstream response-prediction performance.

For TF-IDF + SVD, we compute character-level n-grams (n=1, 2, 3) over available question titles and bodies, and reduce the representation to 128 dimensions using truncated SVD with L_2 normalization. For mixed Chinese–English content in MOOC, we additionally evaluate BGE-small-zh-v1.5 using mean pooling over the last hidden states, followed by a projection layer to match the cognitive model dimension.

Encoder	Type	Params	d_{out}	Frozen
MiniLM-L6-v2	Transformer	22M	384	Yes
DeBERTa-v3-base(He et al., 2023)	Transformer	184M	768	Yes
BGE-small-zh-v1.5(Xiao et al., 2023)	Transformer	24M	512	Yes
LLM emb.	LLM decoder	—	varies	Yes
TF-IDF + SVD(Salton & Buckley, 1988)	Bag-of-words	—	128	Yes

Table 6: Text encoders evaluated in this work. "Frozen" means that encoder parameters are fixed during CRI training. d_{out} is the embedding dimension before the projection layer.

F.4 Per-Dataset Adaptation Notes

DBE-KT22. Exercise texts are raw HTML strings from the CodeBench platform. We strip HTML tags before encoding. Each question includes one or more knowledge component names and descriptions, which are appended to the prompt as structured context. Bloom definitions are adapted to the database domain; for example, *Create* refers to constructing a query, schema, or solution strategy. Evidence spans are copied from the cleaned exercise text.

FrcSub. Exercises are fraction-subtraction expressions annotated with eight procedural skills (α_1–α_8) from the DeCarlo Q-matrix. We include the skill names and definitions in the prompt body. Bloom definitions are adapted to arithmetic; for example, *Remember* refers to recalling fraction-subtraction rules, while *Create* refers to constructing or generalizing a solution strategy. Evidence spans are extracted from the fraction expression and the corresponding skill descriptions.

MOOC. Exercise texts are the original Chinese question stems and answer options from a large-scale Chinese MOOC platform, extracted directly from each exercise's raw record rather than synthesized. Each question is also tagged with one or more concept labels, which we append to the prompt as structured context, mirroring the treatment of knowledge-component names for DBE-KT22. Evidence spans are copied from the Chinese question text and options.

ASSIST. This dataset provides concept names but limited or no raw exercise text. We therefore construct pseudo-exercise descriptions by concatenating the concept name with a generic context sentence, e.g., "This exercise tests the student's ability to apply the concept of `{KC name}`". This keeps the annotation interface consistent with the other datasets, although the resulting labels and evidence spans are expected to be noisier than those for text-rich datasets.

Annotation cost. LLM annotation is performed once before training and is not repeated during model training or inference. At inference time, RouteCD uses only the learned route inducer and does not call the LLM teacher.

G Additional Experimental Results

G.1 Ablation Visualization

Figure 6 provides a visual comparison of the ablation results on DBE-KT22. The figure complements Table 3 by showing how the text encoder, CRI route module, and LLM-derived supervision contribute to response prediction across AUC, ACC, and RMSE. The base CD backbone performs worst across the three metrics, while adding individual components consistently improves performance. Among the single-component variants, the CRI-only model brings the largest improvement, indicating that explicit route-based cognitive structure provides useful diagnostic signal beyond the backbone. Combining text encoding with CRI further improves performance, and the full RouteCD model achieves the best overall results, confirming the complementary effects of text semantics, cognitive route induction, and LLM-derived graph supervision.

G.2 Dataset split details.

Table 7 reports training, validation, and test interaction counts per dataset.

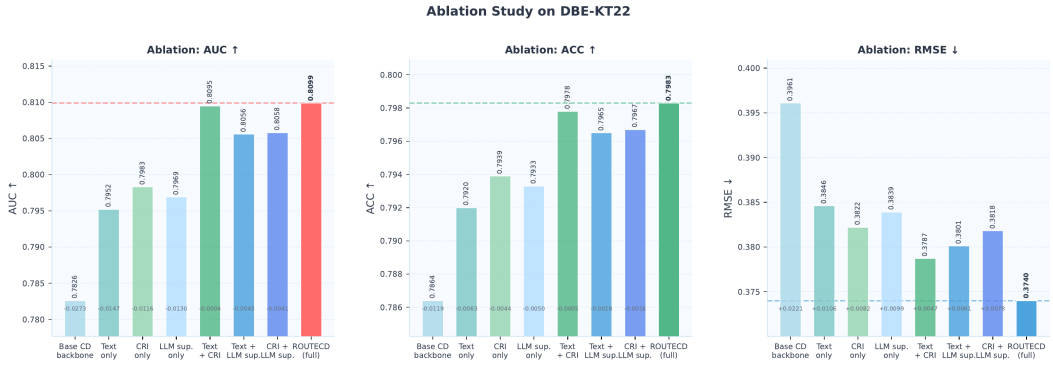


Figure 6: Ablation results on DBE-KT22. The three subplots report AUC, ACC, and RMSE, respectively. Dashed lines mark the full RouteCD model's performance, and bar annotations show the difference from the full model, highlighting the complementary effects of text encoding, CRI route induction, and LLM-derived supervision.

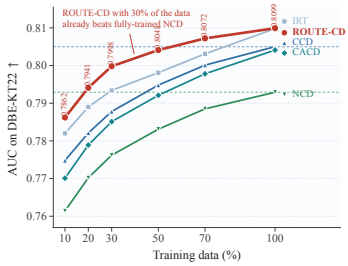


Figure 7: Data efficiency on DBE-KT22 under different training-data ratios.

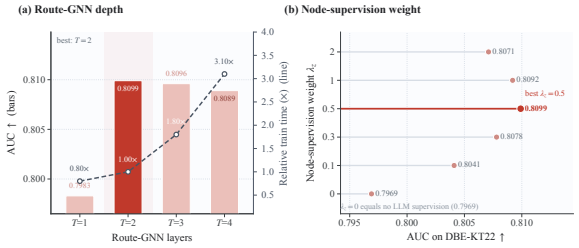


Figure 8: Hyperparameter sensitivity on DBE-KT22. Left: effect of route GNN depth T. Right: effect of node-supervision weight λ_z.

G.3 Computational Efficiency

Because RouteCD introduces additional parameters, auxiliary losses, and a route-induction GNN on top of a text-only backbone, we quantify the resulting overhead and its trade-off against predictive performance. Table 8 compares RouteCD with three ablated variants on DBE-KT22: without LLM-derived graph supervision, without the CRI route GNN, and the text-only backbone.

RouteCD increases the parameter count by only 1.33% over the text-only backbone (271,976 vs. 268,392), with 0.30MB additional inference memory and 0.78ms additional latency per batch. Training time increases from 21.73s to 30.10s per epoch, but this overhead is confined to offline training: LLM annotation is performed once per exercise and cached, and inference requires no LLM calls. This modest overhead accompanies an AUC improvement from 0.7952 to 0.8099 and an RMSE reduction from 0.3846 to 0.3740, indicating a favorable performance–cost trade-off.

G.4 Statistical Significance of Predictive Gains

To complement the averaged results in Table 2 (main text), we assess statistical significance across all 12 dataset–metric combinations using two-sided paired tests with Holm correction for multiple comparisons, based on the same five matched random seeds, data splits, and evaluation settings used throughout. Table 9 reports the results.

RouteCD achieves statistically significant improvements over the strongest metric-wise baseline on all three FrcSub metrics (AUC, ACC, and RMSE) and on MOOC AUC and ACC—five of the twelve comparisons, each surviving conservative Holm correction applied jointly across all twelve. On the remaining comparisons, on ASSIST, DBE-KT22, and MOOC RMSE, RouteCD matches the strongest specialized baseline on each metric without a statistically significant regression in

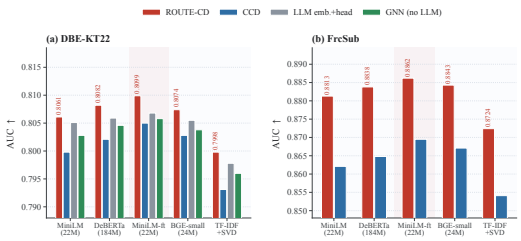


Figure 9: Text encoder sensitivity on DBE-KT22 and FrcSub.

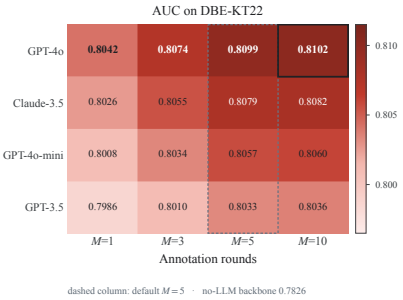


Figure 10: LLM teacher sensitivity on DBE-KT22 across teacher models and annotation rounds.

Dataset	Train	Val	Test
FrcSub	7,504	1,072	2,144
ASSIST	187,191	26,742	53,482
DBE-KT22	110,290	15,756	31,511
MOOC	618,892	88,413	176,827

Table 7: Train/validation/test interaction counts under a 7:1:2 split.

either direction (mean differences within 0.0001–0.0016 of zero, including an exact tie on ASSIST AUC), despite being a single unified architecture evaluated identically across four datasets that differ substantially in scale and textual richness, whereas each baseline is compared only on the metrics where it already performs best. Taken together with the auditable, fine-grained route-level diagnosis established through expert validation, cross-teacher stability, and representative case studies (Appendices G.5.4–G.5.5), these results indicate that ROUTECD's gains in interpretability do not come at the cost of predictive performance.

G.5 COGNITIVE ROUTE QUALITY ANALYSIS

A central concern about CRI is whether the induced cognitive routes provide meaningful exercise-level cognitive structure rather than merely capturing surface-level text patterns. We examine this issue through six complementary analyses: the relationship between route quality and downstream performance, route sparsity and structure, student representation visualization, human expert validation of the induced routes, cross-teacher stability of the induced routes, and a controlled comparison of Bloom-process representation strategies.

G.5.1 ROUTE QUALITY AND DIAGNOSTIC PERFORMANCE

To assess whether better induced routes lead to better diagnosis, we examine the relationship between CRI route quality and downstream response-prediction performance. Specifically, we compute the Pearson correlation between CRI node-level F1 and downstream AUC across different LLM teachers and annotation rounds on DBE-KT22. Figure 11 reports the result.

The strong positive correlation indicates that route quality is closely associated with diagnostic performance. Teacher settings that produce higher-quality Bloom-node predictions also tend to yield higher downstream AUC. This provides indirect evidence that the induced routes are not merely auxiliary annotations, but contribute useful cognitive structure for response prediction. Importantly, this relationship emerges across teacher and annotation-round settings, rather than being explicitly optimized as a training objective.

G.5.2 ROUTE SPARSITY AND STRUCTURE

Figure 12 analyzes the sparsity of induced route graphs on DBE-KT22. We compare the average number of activated Bloom nodes and selected prerequisite edges across exercise types, with and without the sparsity regularization term $\mathcal{L}_{sparse}$.

Model	AUC↑	ACC↑	RMSE↓	Parameters	Train time / epoch	Inference time / batch	Inference memory
Text-only backbone	0.7952	0.7920	0.3846	268,392	21.73s	1.45ms	78.58MB
w/o CRI route GNN	0.8056	0.7965	0.3801	269,924	26.87s	1.95ms	78.87MB
w/o LLM graph supervision	0.8095	0.7978	0.3787	271,976	29.01s	2.19ms	78.88MB
RouteCD	**0.8099**	**0.7983**	**0.3740**	271,976	30.10s	2.23ms	78.88MB

Table 8: Cost–performance analysis on DBE-KT22. All variants require no LLM calls at inference time.

Dataset	Metric	Strongest Baseline	Mean Diff.	Holm-adjusted p	Sig.
FrcSub	AUC	EduEmbed	+0.0143	0.000108	✓
FrcSub	ACC	EduEmbed	+0.0182	0.000154	✓
FrcSub	RMSE	LLM emb.+head	−0.0120	0.000460	✓
ASSIST	AUC	RCD	+0.0000	1.000000	–
ASSIST	ACC	NCD	−0.0016	0.077392	–
ASSIST	RMSE	RCD	+0.0004	0.592592	–
DBE-KT22	AUC	IRT	+0.0001	1.000000	–
DBE-KT22	ACC	EduEmbed	+0.0005	0.445045	–
DBE-KT22	RMSE	IRT	+0.0007	0.104088	–
MOOC	AUC	CCD	+0.0009	0.036288	✓
MOOC	ACC	CACD	+0.0011	0.031932	✓
MOOC	RMSE	CCD	−0.0001	1.000000	–

Table 9: Paired significance tests between RouteCD and the strongest metric-wise baseline across all 12 dataset–metric comparisons, using five matched random seeds and Holm correction for multiple comparisons. Positive differences favor RouteCD for AUC/ACC; negative differences favor RouteCD for RMSE. ✓: significant at α=0.05 after Holm correction.

The results show that $\mathcal{L}_{sparse}$ consistently reduces both node and edge density compared with the variant without sparsity regularization. At the same time, the induced routes remain sensitive to exercise complexity: simple recall exercises activate fewer Bloom processes and edges, while analysis and design exercises induce richer cognitive routes. This pattern suggests that the model avoids overly dense graphs without collapsing all exercises into trivial sparse structures, supporting the interpretability of the induced route graphs.

To assess whether this sparsity pattern generalizes beyond DBE-KT22, Table 10 reports route sparsity and acyclicity statistics across all four datasets, computed over unique exercises after thresholding predicted edge probabilities at 0.5 and removing self-loops. FrcSub, DBE-KT22, and MOOC all yield sparse graphs (1.9–2.9 edges of 30 possible per exercise) and are fully acyclic after regularization. ASSIST is denser (9.85 edges) and retains a 25.85% cycle rate even after regularization, though joint sparsity and acyclicity regularization still reduces its pre-regularization cycle rate from 68.39% to this level—confirming that the regularizers are necessary rather than redundant on datasets with sparser textual grounding, where CRI has less direct evidence for resolving prerequisite structure.

G.5.3 Student Representation Visualization

Figure 13 visualizes the learned student cognitive state representations on the DBE-KT22 test set. Students are grouped by their overall response accuracy. The visualization shows a clear separation between low-performing and high-performing students, with intermediate groups located between them. This qualitative pattern suggests that the concept-by-process student state captures meaningful variation in student mastery and supports route-aware diagnosis.

G.5.4 Human Expert Validation of Induced Routes

Because CRI is trained entirely from LLM-generated weak supervision, we assess whether the induced cognitive routes reflect genuine exercise-level cognitive demand rather than teacher-specific bias. We recruited three doctoral researchers specializing in intelligent education to independently judge all 212 DBE-KT22 exercises and 20 FrcSub exercises, evaluating whether the activated Bloom-process nodes and prerequisite edges are appropriate given each exercise's text and concept annotations.

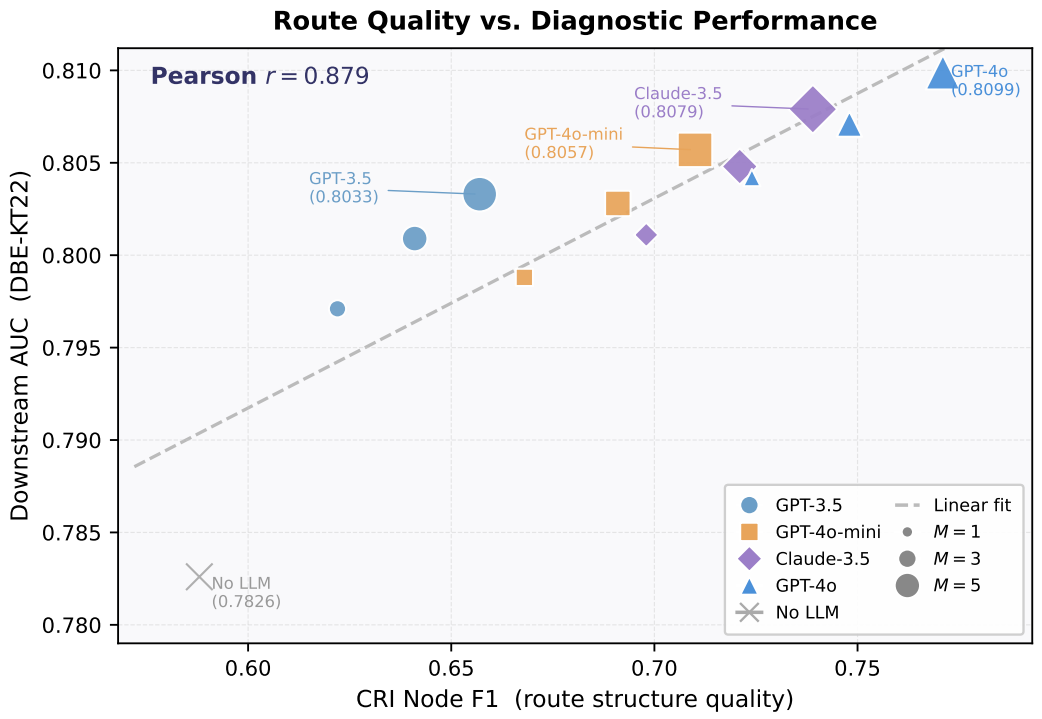


Figure 11: Route quality versus diagnostic performance on DBE-KT22. Higher CRI node F1 is positively correlated with downstream AUC, suggesting that more accurate route induction contributes to better response prediction.

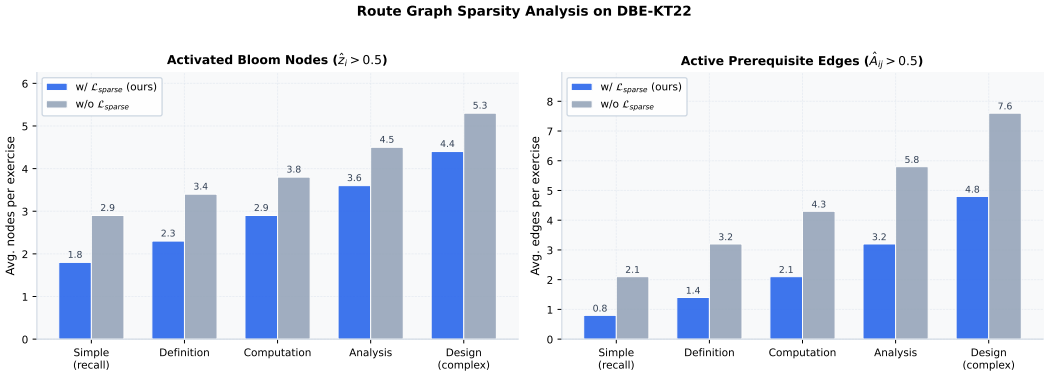


Figure 12: Route graph sparsity analysis on DBE-KT22. Sparsity regularization reduces the number of activated Bloom nodes and prerequisite edges while preserving the trend that more complex exercises induce richer route structures.

As shown in Table 11, majority-vote acceptance reached 89.2% (node sets) and 94.3% (edge sets) on DBE-KT22, and 95.0% on both node and edge sets on FrcSub; complete-route acceptance—requiring the entire induced graph to be judged correct—reached 84.0% on DBE-KT22 and 95.0% on FrcSub. Because binary judgments were strongly skewed toward acceptance, we report raw agreement and majority acceptance alongside Fleiss' κ rather than interpreting κ in isolation, since chance-agreement correction mechanically compresses κ even under strong rater concordance (raw agreement was at least 90.0% on both datasets). Under conservative adjudication, only 35 of the 232 exercises required

Dataset	Unique Exercises	Avg. Edges	Edge Density	Cycle Rate
FrcSub	20	1.900	6.33%	0.00%
ASSIST	17,671	9.851	32.84%	25.85%
DBE-KT22	212	2.887	9.62%	0.00%
MOOC	2,488	2.593	8.64%	0.00%

Table 10: Route sparsity and acyclicity across all four datasets, after sparsity/acyclicity regularization. Edge density and cycle rate are computed over the 30 possible directed edges among the six Bloom processes.

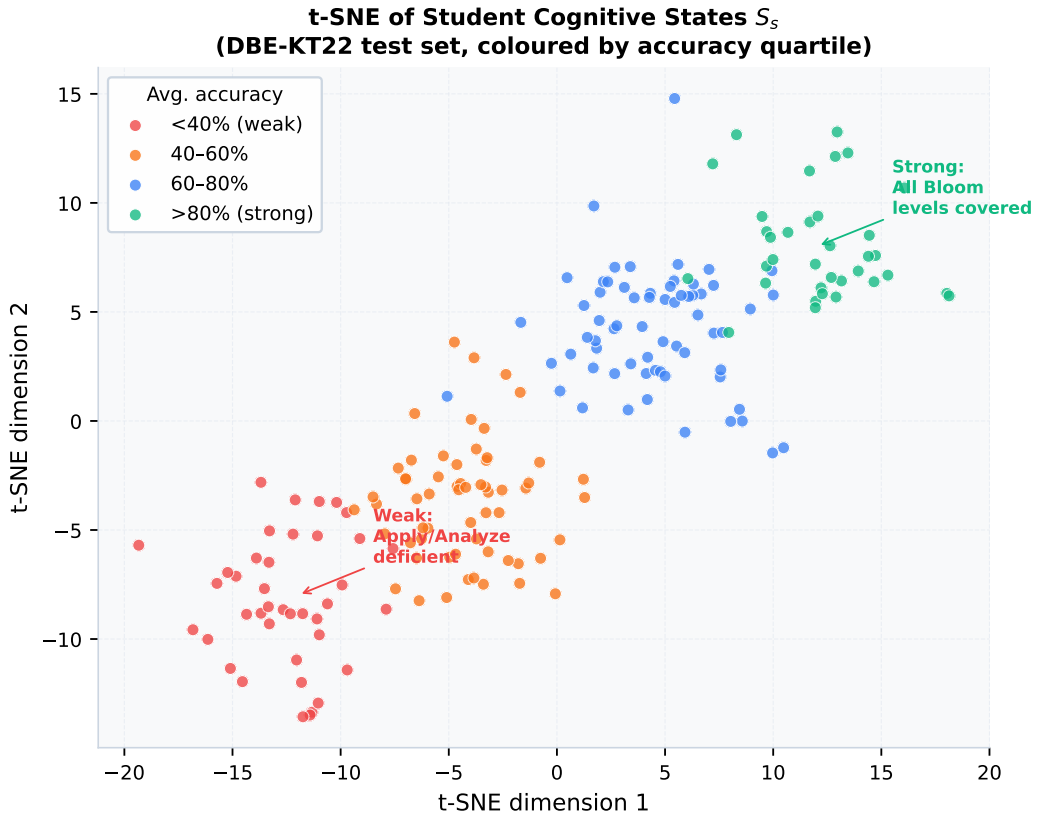


Figure 13: t-SNE visualization of learned student cognitive states on the DBE-KT22 test set. Students are colored by overall response accuracy groups. The learned representations show a clear progression from low-accuracy to high-accuracy students.

any correction, providing direct human evidence that the LLM-generated routes are largely consistent with expert judgments of exercise-level cognitive demand.

G.5.5 Cross-Teacher Stability of Induced Routes

A separate concern is whether induced routes are idiosyncratic to a single LLM teacher rather than reflecting stable exercise-level structure. We induce cognitive routes for all 212 DBE-KT22 exercises using five independent teachers—GPT-4o, Claude-3.5-Sonnet, GPT-4o-mini, GPT-3.5-Turbo, and Qwen3.5-35B-A3B—following the same annotation pipeline with $M=5$ query rounds per exercise.

Table 12 reports pairwise structural agreement between all ten teacher pairs. The average Node, Edge, and Full-Graph Jaccard scores are 0.830, 0.629, and 0.735, respectively, with a mean normalized graph edit distance of 0.051, indicating substantial agreement in Bloom-node selection and moderate

Dataset	Exercises	Node Fleiss' κ	Edge Fleiss' κ	Node Majority Acc.	Edge Majority Acc.
DBE-KT22	212	0.466	0.328	89.2%	94.3%
FrcSub	20	0.196	0.196	95.0%	95.0%

Table 11: Expert validation results for LLM-generated cognitive routes.

Teacher Pair	Node Jaccard↑	Edge Jaccard↑	Full-Graph Jaccard↑	GED↓	Normalized GED↓
GPT-4o vs Claude-3.5-Sonnet	0.8323	0.6400	0.7358	2.0519	0.0570
GPT-4o vs GPT-4o-mini	0.8049	0.5549	0.6792	2.5708	0.0714
GPT-4o vs GPT-3.5-Turbo	0.8550	0.6644	0.7577	1.9009	0.0528
GPT-4o vs Qwen3.5-35B-A3B	0.7493	0.5250	0.6402	2.6179	0.0727
Claude-3.5-Sonnet vs GPT-4o-mini	0.8299	0.6329	0.7458	1.5943	0.0443
Claude-3.5-Sonnet vs GPT-3.5-Turbo	0.9167	0.8002	0.8603	1.0000	0.0278
Claude-3.5-Sonnet vs Qwen3.5-35B-A3B	0.8571	0.7046	0.7863	1.3396	0.0372
GPT-4o-mini vs GPT-3.5-Turbo	0.8117	0.5832	0.7084	1.9623	0.0545
GPT-4o-mini vs Qwen3.5-35B-A3B	0.8173	0.5476	0.7000	1.6981	0.0472
GPT-3.5-Turbo vs Qwen3.5-35B-A3B	0.8281	0.6393	0.7381	1.7642	0.0490
Average (10 pairs)	**0.8302**	**0.6292**	**0.7352**	**1.8500**	**0.0514**

Table 12: Pairwise structural agreement of cognitive routes induced by five different LLM teachers on DBE-KT22.

stability in prerequisite-edge induction, with most residual disagreement concentrated at the edge level.

Separately, Table 13 reports each teacher's agreement with the five-teacher majority route. Every teacher achieves a node F1 above 0.91, with full-graph F1 ranging from 0.8521 to 0.9596, confirming that no single teacher dominates the consensus route. Notably, GPT-4o—our default teacher, chosen for downstream utility rather than inter-teacher typicality—shows the *lowest* agreement with the majority route among the five teachers, indicating that RouteCD's gains are not simply a proxy for teacher-consensus conformity.

G.5.6 Bloom-Process Representation Variants

RouteCD represents each Bloom process with a randomly initialized, trainable ID embedding $\mathbf{p}_i$ (Eq. 3), rather than encoding Bloom stages through their label words or textual definitions. To assess whether richer semantic initialization improves route induction or response prediction, Table 14 compares the original learnable-ID representation against one-hot, label-word, definition-enhanced, and taxonomy-aware alternatives under identical settings on DBE-KT22.

The original learnable-ID representation achieves the best response-prediction performance (AUC 0.8098 ± 0.0021, ACC 0.7985 ± 0.0018, RMSE 0.3736 ± 0.0012). Although the one-hot and taxonomy-aware variants obtain comparable or slightly higher Node/Edge F1, none of the richer initializations translate into better response prediction. This suggests that the initial Bloom-stage embedding is not the primary bottleneck for CRI; relationships among cognitive stages are instead learned mainly through exercise-conditioned route induction (Eq. 3–5) and graph propagation, rather than through the initialization of node identity itself.

G.6 Bloom-Level Demand Analysis

Figure 14 shows the average Bloom-process activation patterns across exercise types in CRI-DBE22. Several patterns are noteworthy. Recall and Definition exercises mainly activate Remember and Understand, while Computation exercises show the strongest activation at Apply. Analysis exercises activate Analyze more strongly, together with substantial Apply and Understand demand. Design exercises show the highest activation at Create, while also requiring Apply, Analyze, and Evaluate. These patterns align with educational expectations and provide qualitative evidence that the induced routes capture meaningful differences in cognitive demand across exercise types, rather than assigning similar Bloom profiles to all exercises.

LLM Teacher	Node F1↑	Edge F1↑	Full-Graph F1↑
GPT-4o	0.9137	0.7677	0.8521
Claude-3.5-Sonnet	0.9763	0.9354	0.9596
GPT-4o-mini	0.9125	0.7429	0.8563
GPT-3.5-Turbo	0.9678	0.8974	0.9387
Qwen3.5-35B-A3B	0.9223	0.7948	0.8768

Table 13: Agreement between each LLM teacher and the five-teacher majority route on DBE-KT22.

Bloom Representation	AUC↑	ACC↑	RMSE↓	Node F1↑	Edge F1↑
One-hot ID + projection	0.7979	0.7902	0.3791	**0.9131**	0.8150
Original learnable ID embedding	**0.8098**	**0.7985**	**0.3736**	0.9106	**0.8313**
Label-word initialization	0.7999	0.7899	0.3788	0.8965	0.8140
Definition-enhanced initialization	0.8001	0.7906	0.3783	0.8967	0.8181
Taxonomy-aware initialization	0.7998	0.7889	0.3786	0.9113	0.8307

Table 14: Comparison of Bloom-process representation strategies on DBE-KT22 (mean over five seeds; standard deviations omitted for brevity, all within ± 0.003).

G.7 Concept-Support Constraints on the Demand Matrix

The concept-by-process demand matrix D_e is conditioned on the exercise concept vector q_e but is not explicitly masked or constrained to assign nonzero demand only to annotated concepts. To assess whether this lack of an explicit constraint introduces spurious, hard-to-justify demand on unrelated concepts, we measure the *off-annotation demand mass*: the proportion of D_e's total demand mass assigned to concepts outside the exercise's annotated Q-matrix support.

We compare the original (unmasked) D_e against two concept-constrained variants, using $M_e = q_e \mathbf{1}_6^\top$: a hard mask $D_e^{\text{hard}} = M_e \odot D_e$, and a soft mask $D_e^{\text{soft}} = [M_e + \epsilon(1 - M_e)] \odot D_e$ with ϵ=0.05. Table 15 reports the results on DBE-KT22 (single-seed analysis; the three variants share text backbone, exercise features, data splits, and optimization settings).

The unmasked D_e already concentrates about half of its total demand mass (48.44%) on the annotated concepts, even though these concepts make up only $\sim$2% of all concepts, indicating that D_e is meaningfully concept-conditioned despite not being concept-constrained. Hard masking eliminates off-annotation demand mass by construction (0.00%) and soft masking reduces it to 21.89%, but both constraints slightly reduce predictive performance (AUC from 0.7964 to 0.7679/0.7678), revealing a trade-off between unrestricted latent representations and explanations whose support is guaranteed to align with concept annotations. We clarify that off-annotation demand mass is not necessarily invalid—the Q-matrix may omit latent cross-concept dependencies—but offer the hard-masked variant as an optional strict constraint for settings that require Q-matrix-consistent explanations.

G.8 ASSISTments Route Variation Under Sparse Text

Because ASSISTments lacks authentic exercise descriptions, we construct pseudo-exercise text by concatenating concept names with a fixed template (Section 4.1), treating this as a text-sparse stress test rather than the primary evidence for text-grounded route validity. To assess whether CRI still induces routes that vary meaningfully with exercise difficulty under this sparse setting, we stratify ASSISTments exercises into Easy, Medium, and Hard groups using empirical difficulty computed exclusively from training-set interactions (1 minus the training-only correct-response rate), and report two soft structural statistics: *Soft Node Mass* (the sum of predicted node-activation probabilities) and *Soft Edge Mass* (the sum of predicted edge probabilities).

As shown in Table 16, across the training-derived difficulty strata, test ACC decreases from 0.7820 to 0.6926 and RMSE increases from 0.3879 to 0.4624, confirming that the strata reflect genuinely harder exercises. Soft Node Mass increases from 1.2774 to 2.4229 and Soft Edge Mass increases from 9.1365 to 14.5677, indicating that induced routes grow systematically more complex as exercise difficulty increases, rather than collapsing to a single template-fixed structure. We note that this analysis demonstrates systematic route variation associated with difficulty rather than independently

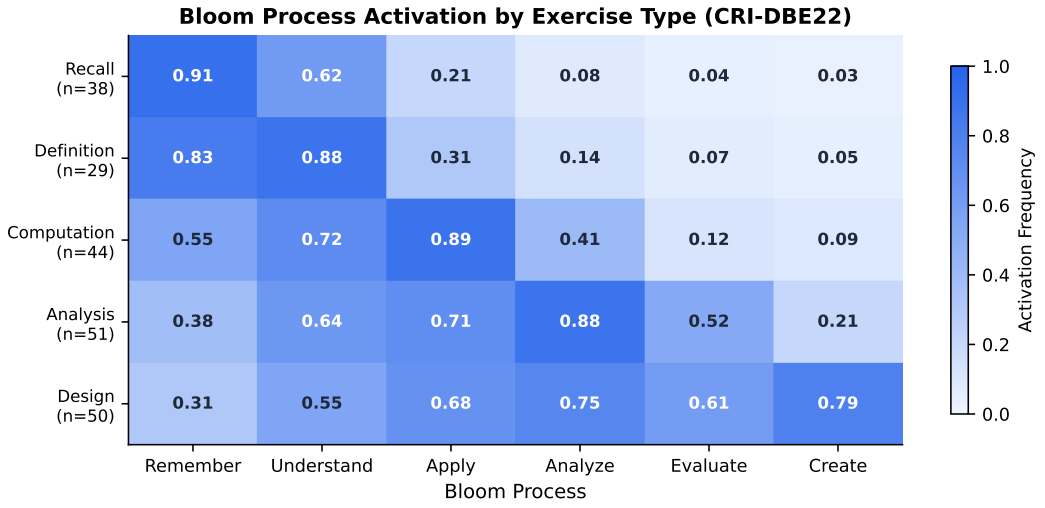


Figure 14: Bloom-process activation by exercise type on CRI-DBE22. Rows correspond to exercise types, and columns correspond to Bloom processes. Brighter cells indicate higher activation frequency.

Demand Variant	Test AUC↑	Test ACC↑	Test RMSE↓	Off-Annotation Demand Mass↓
Unmasked D_e (original)	**0.7964**	**0.7865**	**0.3801**	51.56%
Soft-masked D_e^{soft}	0.7678	0.7763	0.3897	21.89%
Hard-masked D_e^{hard}	0.7679	0.7825	0.3883	**0.00%**

Table 15: Effect of concept-support constraints on the demand matrix D_e on DBE-KT22 (single-seed analysis). Off-Annotation Demand Mass is the proportion of total demand mass assigned to concepts not in the exercise's Q-matrix annotations; it should not be interpreted as proving those entries are educationally invalid, since the Q-matrix may omit latent cross-concept dependencies.

establishing cognitive validity; on ASSISTments, cognitive-route validity is supported more directly by the pattern above than by textual richness, while expert validation and cross-teacher stability (Appendices G.5.4–G.5.5) provide validity evidence on the text-rich datasets.

G.9 Cold-Start Performance

Figure 15 evaluates performance under different student response-history lengths. RouteCD achieves stronger AUC than the baselines in the low-history groups, especially for students with only 1–5 or 6–15 observed responses. This advantage is consistent with the intuition that cognitive routes provide exercise-side diagnostic signal that partially compensates for sparse student-side data. As history length increases, the gap between models becomes smaller, but RouteCD remains competitive. This property is practically important in real educational systems, where new students often have limited interaction history.

G.10 Cross-Domain Generalisation

Figure 16 evaluates whether CRI can generalize its structured route induction ability across domains. We compare CRI with rule-chain, DeBERTa-based, and LLM-majority baselines under a zero-shot setting, using node F1 and edge F1 as route-quality metrics. Across Math, Reading, and Science, CRI achieves the strongest node and edge F1, indicating that the learned route inducer transfers better than fixed rule chains or purely text-based baselines.

Difficulty	Exercises	ACC↑	RMSE↓	Soft Node Mass	Soft Edge Mass
Easy	3,921	0.7820	0.3879	1.2774	9.1365
Medium	5,164	0.7614	0.4232	1.7544	11.4459
Hard	4,142	0.6926	0.4624	2.4229	14.5677

Table 16: ASSISTments results stratified by training-only empirical exercise difficulty. Soft Node/Edge Mass are the summed predicted node/edge activation probabilities, averaged per exercise within each stratum.

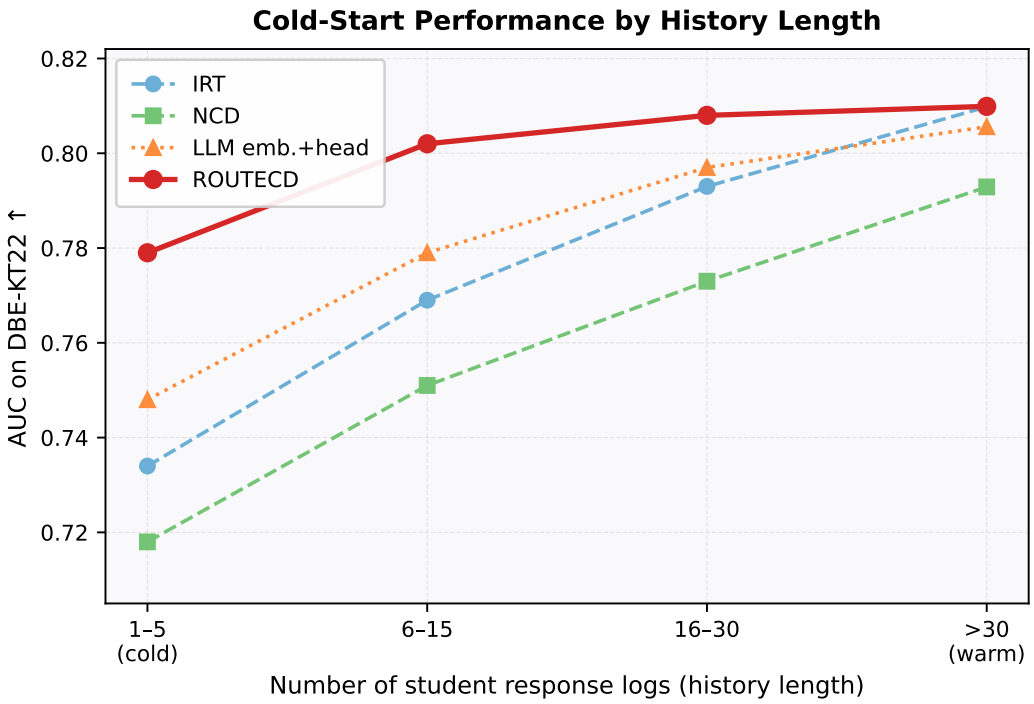


Figure 15: Cold-start performance on DBE-KT22 by student response-history length. Students are grouped by the number of observed response logs. RouteCD shows stronger AUC in low-history regimes and remains competitive as history length increases.

These results suggest that Bloom-based cognitive route structures capture transferable aspects of exercise-level cognitive demand. Although lexical content and subject-specific concepts vary across domains, the underlying cognitive processes and prerequisite patterns remain partially shared, allowing CRI to generalize beyond the source domain.

G.11 Qualitative Case Study: Cognitive Route Diagnosis

To illustrate the interpretability of RouteCD, we present two anonymized and lightly paraphrased DBE-KT22 examples. Each pair shares the same concept label but induces different cognitive routes, showing how route-aware diagnosis distinguishes different learning needs that would be collapsed by concept-level diagnosis.

G.11.1 Case 1: JOIN Concept

Exercise A (Simple recall). *"What does INNER JOIN do in SQL?"* Induced route: **Remember** → **Understand**. Both nodes are activated, with one prerequisite edge selected. The exercise mainly tests whether the student can recall and explain the semantics of the JOIN operation.

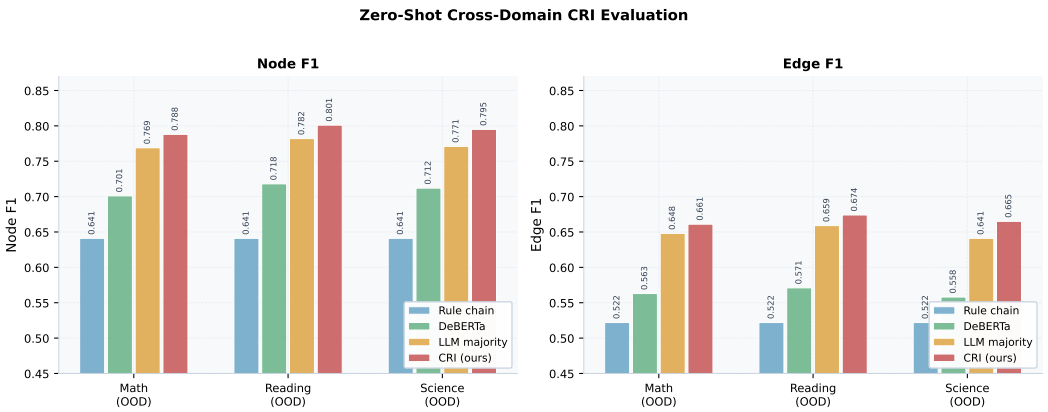


Figure 16: Zero-shot cross-domain CRI evaluation. Models are trained on the source domain and evaluated on out-of-domain Math, Reading, and Science exercises without fine-tuning. We report node-level and edge-level F1 for cognitive route induction.

Exercise B (Complex application). *"Given Customer(id, name) and Order(id, customer_id, amount), write an SQL query to list customers whose total orders exceed 100."* Induced route: **Understand** → **Analyze** → **Apply** → **Create**. Four nodes are activated, with three prerequisite edges selected. The exercise requires understanding the schema, analyzing key relations between tables, applying aggregation and filtering, and constructing the final SQL query.

Diagnostic contrast. The cognitive gap profiles distinguish students with different error patterns. One student shows larger gaps at the **Remember** and **Understand** levels for the JOIN concept, suggesting weakness in declarative understanding of JOIN semantics. Another student shows small gaps at **Remember/Understand** but larger gaps at **Analyze** and **Create**, suggesting difficulty with multi-table schema reasoning and query construction. A concept-level model would assign both cases to the same JOIN weakness, whereas RouteCD localizes the breakdown to different cognitive operations.

G.11.2 Case 2: WHERE / Filtering Concept

Exercise C (Single-condition filter). *"Select all employees with salary > 50000."* Induced route: **Remember** → **Apply**. Two nodes are activated, with one prerequisite edge selected. The exercise tests direct application of a WHERE clause with a single numeric condition.

Exercise D (Multi-condition filter with aggregation). *"Find departments where the average salary exceeds 60000 and at least 5 employees work there, ordered by department name."* Induced route: **Understand** → **Analyze** → **Apply** → **Evaluate** → **Create**. Five nodes are activated, with four prerequisite edges selected. The exercise requires understanding aggregation semantics, analyzing compound conditions, applying GROUP BY and HAVING clauses, checking the ordering constraint, and constructing the full query.

Diagnostic contrast. The cognitive gap profiles again suggest different interventions. A student with a large **Apply** gap on Exercise C likely needs practice with basic WHERE syntax, while a student with gaps concentrated at **Analyze** and **Evaluate** on Exercise D likely needs practice with compound conditions and aggregation constraints. RouteCD surfaces this distinction, whereas concept-level diagnosis would mainly identify both cases as weaknesses in filtering-related concepts.

Summary. These cases illustrate the diagnostic value of cognitive route induction: by representing exercise demand as an interpretable Bloom-based route and comparing it with student cognitive state in a shared concept-by-process space, RouteCD can pinpoint *which cognitive operations* contribute

Case	Exercise and Concepts	Activated Bloom Processes	Main Induced Dependencies	Largest Concept–Process Gap
Short route	Ex. 62: identify the operation that combines two relations by merging matching tuples; **Join, DML**	Remember, Understand	R→U (0.916)	DML × Understand (0.507)
Complex route	Ex. 164: determine whether two relational-algebra expressions are equivalent under the provided database schema; **Equivalent, Relational Algebra**	Remember, Understand, Apply, Analyze, Evaluate	R→U (0.840), U→Ap (0.778), Ap→An (0.791), An→E (0.743)	Equivalent × Apply (0.743)
Same concept, different route	Ex. 161: determine the number of rows produced by the given renamed self-join expression; **Join, Relational Algebra, Renaming**	Remember, Understand, Apply, Evaluate	R→U (0.625), R→Ap (0.629), U→Ap (0.754)	Renaming × Remember (0.446)

Table 17: Representative CRI case studies on DBE-KT22, quantifying the induced routes—including edge weights (confidence scores) on the dominant prerequisite dependencies—and the largest observed concept–process gap for a representative student on each exercise.

to a student's breakdown, not merely *which concept* is involved. Manual inspection indicates that the induced routes align with plausible pedagogical difficulty progressions in SQL instruction.

Quantitative summary. Table 17 summarizes three representative DBE-KT22 exercises that induce different cognitive routes despite partial concept overlap, complementing the qualitative cases above with explicit induced-route statistics—including edge weights on the dominant prerequisite dependencies—and the largest concept–process gap observed for a representative student on each exercise.

H LIMITATIONS

This paper has two main limitations. First, CRI relies on LLM-generated weak supervision, which may introduce teacher-specific biases and limits direct human validation of the induced cognitive routes. Second, route induction depends on the availability and quality of exercise text, potentially reducing its effectiveness on datasets with sparse or noisy item descriptions. Future work will incorporate expert annotations, richer educational resources, and interactive tutoring studies to address these issues.