Tailoring Diagnostic Modeling to Individual Learners: Personalized Distractor Generation via MCTS-Guided Reasoning Reconstruction

Tao Wu[1], **Jingyuan Chen**[1†], **Wang Lin**[1], **Jian Zhan**[1], **Mengze Li**[2], **Fangzhou Jin**[1], **Min Zhang**[3], **Kun Kuang**[1], **Fei Wu**[1†]
[1] Zhejiang University, [2] Hong Kong University of Science and Technology
[3] East China Normal University
twu22@zju.edu.cn, jingyuanchen@zju.edu.cn

Abstract

Distractors—incorrect yet plausible answer choices in multiple-choice questions (MCQs)—are vital in educational assessments, as they help identify student misconceptions by presenting potential reasoning errors. Current distractor generation methods typically produce shared distractors for all students, ignoring the individual variations in reasoning, which limits their diagnostic effectiveness. To tackle this challenge, we introduce the task of **Personalized Distractor Generation**, which tailors distractors to each student's specific cognitive flaws, inferred from their past question-answering (QA) history. While promising, this task is particularly demanding due to the limited number of QA records available for each student, which are insufficient for training, as well as the absence of their underlying reasoning process. To overcome this, we propose a novel, training-free two-stage framework. In the first stage, Monte Carlo Tree Search (MCTS) is used to reconstruct the student's reasoning process from past errors, creating a student-specific misconception prototype. In the second stage, this prototype guides the simulation of the student's reasoning on new questions, generating personalized distractors that resonate with their individual misconceptions. Our experiments, conducted on 1,361 students across 6 subjects, demonstrate that this approach outperforms existing methods in generating plausible, personalized distractors, and also effectively adapts to group-level settings, highlighting its robustness and versatility. Project page: https://mccartney01.github.io/pdg/.

1 Introduction

Multiple-choice questions (MCQs) are widely used in educational assessment, especially in online learning where large-scale evaluation is essential (Biancini et al., 2024; Mucciaccia et al., 2025).

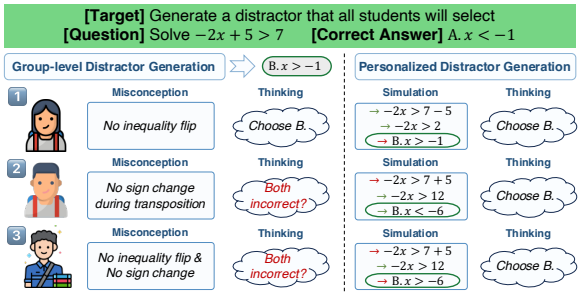


Figure 1: Group-level distractors often fail to reveal diverse student misconceptions, while personalized distractors better align with individual reasoning errors.

A key component of MCQs is the design of *distractors*—incorrect but plausible answer choices intended to uncover student misconceptions by targeting specific errors (Alhazmi et al., 2024).

Recent studies have explored automating distractor generation using large language models (LLMs), typically by learning common error patterns across large student populations (Çavusoglu et al., 2024; Byun and Choi, 2025; Feng et al., 2024). These methods generate a **shared** set of distractors per question that reflect broadly observed misconceptions (Scarlatos et al., 2024; Fernandez et al., 2024). However, such **group-level distractors** often fail to capture the diverse, student-specific nature of reasoning errors (Alhazmi et al., 2024; Lee et al., 2025). When a student's misconception does not align with the predefined distractors, the question loses its diagnostic effectiveness (Liang et al., 2018). For example, as shown in Figure 1, a distractor targeting one misconception (*e.g.*, failing to flip the inequality sign) fails for students 2 and 3, whose errors stem from different issues (*e.g.*, incorrect transposition). As a result, they may guess, leaving their misconceptions unexposed and the question diagnostically ineffective.

To address this, we propose a new task, **Personalized Distractor Generation**, which aims to generate tailored distractors based on each student's

† Corresponding authors.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 21500–21524

specific misconceptions identified from their historical errors[1]. Unlike group-level distractors, personalized distractors precisely align with individual misconceptions, enabling more accurate diagnoses across diverse learners, as shown in Figure 1.

Can existing distractor generation methods tackle the personalized distractor generation task? Unfortunately, no. Existing methods typically rely on large-scale student QA records to learn group-level error patterns, making them ineffective at capturing individual student's unique misconceptions with limited QA records (Wu et al., 2024a). Additionally, MCQA records only include selected options without intermediate reasoning steps, making it hard to infer why students chose specific distractors or identify underlying misconceptions (Reinhart et al., 2022). A potential solution is to leverage LLMs to reconstruct these missing reasoning processes, given their strong reasoning capabilities (Ahn et al., 2024). However, this is challenging, as LLMs are typically optimized to produce correct explanations, not to simulate student-like errors (Wu et al., 2025; Srivatsa et al., 2025). Fine-tuning on error-rich data may help, but it risks injecting incorrect knowledge and degrading the model's overall reasoning performance (Sonkar et al., 2024).

To tackle these challenges, we propose a training-free two-stage framework for personalized distractor generation. In the first stage, we analyze each student's past QA records to extract key concepts and reconstruct their reasoning trajectories using a Monte Carlo Tree Search (MCTS) approach. MCTS breaks down problem-solving into steps, sampling errors at each step to explore potential reasoning paths that may lead to the chosen distractor. The search is guided by a self-evaluation mechanism that favors realistic student-like reasoning, resulting in the most plausible trajectory. This trajectory, along with the extracted knowledge concepts, forms a personalized misconception prototype that captures the student's errors. In the second stage, given a new question, we retrieve relevant misconceptions from the prototype and simulate the student's reasoning process to generate a personalized distractor reflecting the misconceptions.

To support our study, we curate a dataset, **`Student_1361`**, covering 1,361 students across 6 subjects, ranging from elementary to undergraduate levels. The dataset provides temporally stable MCQA sequences, making it ideal for personalized distractor generation. Experiment results show that our method consistently outperforms strong baselines in generating plausible, student-specific distractors. Additionally, real-world experiments confirm the diagnostic effectiveness of personalized distractors in revealing individual misconceptions. Overall, our contributions are as follows:

- We define the novel task of personalized distractor generation, aimed at producing student-specific distractors based on individual misconceptions to enhance diagnostic effectiveness.
- We introduce a training-free two-stage framework that uses MCTS to reconstruct student reasoning and generate personalized distractors.
- Extensive experiments show that our method excels at generating student-specific distractors, with real-world experiments validating the diagnostic effectiveness of personalized distractors.

2 Related Work

2.1 Distractor Generation

Recent work has leveraged large language models (LLMs) to automate plausible distractor generation for MCQs (Feng et al., 2024; Bitew et al., 2023), typically by learning common error patterns from large-scale student response data (Mucciaccia et al., 2025; Çavusoglu et al., 2024). Early approaches such as option tracing (An et al., 2022; Ghosh et al., 2021) have studied student answer patterns, but they typically formulate the problem as prediction over predefined options or latent knowledge states rather than open-ended distractor generation. More recent works have introduced large language models to move beyond this index-based formulation, enabling open-ended generation of plausible distractors in natural language. For example, D-GEN (Byun and Choi, 2025) models distractor distributions using MCQ datasets, Scarlatos et al. (2024) ranks candidates by plausibility, DiVERT (Fernandez et al., 2024) encodes group-level misconceptions, and LOOKALIKE (Parikh et al., 2025) uses direct preference optimization (Rafailov et al., 2023) to align with typical student errors. However, these methods produce shared, group-level distractors that fail to capture individual reasoning errors, limiting their diagnostic value (Alhazmi et al., 2024; Lee et al., 2025). To address this gap, we propose personalized distractor generation, which aims to generate student-specific distractors that reflect each learner's individual misconcep-

[1]We provide detailed comparison between personalized and group-level distractor generation in Appendix A.

tions and reasoning patterns, thereby enabling more fine-grained and effective diagnostic assessment.

2.2 Cognitive Diagnosis

Cognitive diagnosis aims to assess students' mastery of knowledge concepts and identify their misconceptions, providing interpretable insights for personalized learning (Wang et al., 2024a; Dong et al., 2025; Zhu et al., 2025). Recent approaches often rely on deep neural networks that encode students' cognitive states as latent parameters (Han et al., 2025; Sun et al., 2025), which, while effective for performance prediction, lack interpretability and make it difficult to pinpoint specific reasoning errors. Efforts like student simulation (Wu et al., 2025) or remediation (Wang et al., 2024b) attempt explicit modeling of cognitive states in natural language, but require detailed reasoning traces that are rarely available in multiple-choice question settings. To address this, we propose an MCTS-based approach that reconstructs reasoning trajectories directly from selected answers, enabling interpretable and student-specific misconception modeling for personalized distractor generation.

3 Dataset Curation

Personalized distractor generation assumes that a student's misconceptions remain stable over short periods, enabling reliable analysis and the generation of individualized distractors. This requires datasets with two essential properties: (1) temporally stable cognitive states for each student and (2) sequential MCQA records.

To meet these requirements, we construct a dataset called **`Student_1361`**, which includes 1,361 students across 6 subjects. These subjects span a wide range of difficulty levels, from elementary-level to undergraduate-level topics, including mathematics, computer science, and more. To ensure diversity, the dataset combines data sourced from publicly available datasets (*e.g.*, Eedi (Wang et al., 2020)) and data collected from an online question-answering platform. These data sources provide a large volume of student answer records and challenging questions, making it ideal for testing and generating personalized distractors.

To ensure quality and consistency, we apply several processing steps. First, we retain only the QA sequences completed within a week for each student to maintain cognitive stability. We also filter out questions unsuitable for distractor generation, such as open-ended or ambiguous question stems (*e.g.*, "*Which of the following is true?*"). Detailed filtering criteria are provided in Appendix B.2.

The final `Student_1361` dataset contains a total of around 80K QA records, with each student having an average of 49 historical QA records and 9 test records. The test QA records consist only of questions the student answered incorrectly and are used to evaluate personalized distractor generation methods. The ground-truth distractor is the actual incorrect option selected by the student, serving as a reliable benchmark for this task. Dataset statistics and illustrations are provided in Appendix B.

4 Method

4.1 Overview

Personalized distractor generation focuses on creating student-specific incorrect answer options that reflect individual misconceptions, rather than generic distractors shared across large student populations. To tackle this task, we propose a two-stage framework that operates independently for each student. In the first stage, given a student's M past QA records $P = \{P_i\}_{1 \le i \le M} = \{(s_i, a_i, d_i)\}_{1 \le i \le M}$, where s_i, a_i, d_i denote question stem, correct answer and selected distractor, respectively, we analyze each selected distractor to reconstruct a plausible reasoning process. This helps identify recurring error patterns, which are then summarized into a personalized misconception prototype. In the second stage, for each new question s_j ($M+1 \le j \le M+N$, where N is the number of the student's test QA records), we leverage this prototype to simulate the student's most likely reasoning process, predict their answer, and generate a tailored distractor that reflects their specific misconceptions.

4.2 Misconception Prototype Construction

To generate personalized distractors, it is essential to first understand the student's recurring misconceptions, which are often embedded in erroneous reasoning processes (Lau et al., 2011). To achieve this, we construct a personalized misconception prototype for each student. Specifically, for each past QA record, we extract key knowledge concepts and apply Monte Carlo Tree Search (MCTS) (Zhang et al., 2024a,b; Malik et al., 2021) to reconstruct a plausible reasoning trajectory. These trajectories are then summarized into a personalized prototype representing the student's typical misconceptions. This process involves 3 key steps.

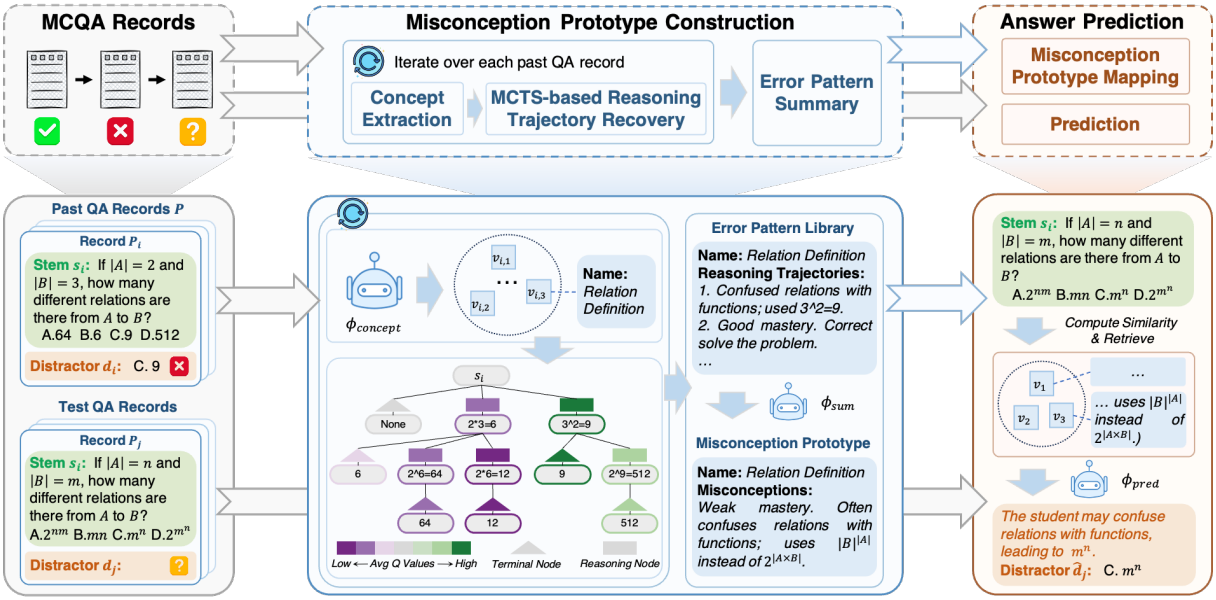


Figure 2: Overview of our two-stage personalized distractor generation framework. In the first stage, given a student's multiple-choice question-answering (MCQA) records, we construct a personalized misconception prototype. This involves: (1) extracting relevant knowledge concepts for each record; (2) applying Monte Carlo Tree Search (MCTS) to reconstruct plausible reasoning trajectories that lead to the student's selected distractors; and (3) summarizing these error trajectories and related concepts into a personalized misconception prototype. In the second stage, given a new question, we retrieve relevant misconceptions from the prototype, feed them into the model to simulate a plausible reasoning, and generate a personalized distractor.

4.2.1 Concept Extraction

To generate effective personalized distractors, it's crucial to understand where a student's misconception stems from. In multiple-choice questions, a wrong answer can stem from confusion between multiple concepts. To support precise and interpretable misconception analysis, we use the model $\phi_{concept}$ to automatically extract relevant concepts for each record P_i:

$$[v_{i,1}, v_{i,2}, \cdots] = \phi_{concept}(P_i) \quad (1)$$

where $[v_{i,1}, v_{i,2}, \cdots]$ denote the extracted concepts. This enables the framework to pinpoint errors related to specific concepts, facilitating accurate generation of student-specific distractors.

4.2.2 MCTS-based Reasoning Trajectory Recovery

Student misconceptions often manifest in their incorrect reasoning processes, which are crucial to reconstruct in order to identify underlying errors—especially since such reasoning steps are typically absent in MCQA settings. Current LLMs might struggle to simulate such error-prone reasoning within a single prompt, as they are typically optimized to generate correct answers (Wu et al., 2025). To address this, we propose an MCTS-based approach that decomposes the reasoning into multiple steps. At each step, the model explores potential errors, and through the search, it identifies the most plausible reasoning trajectory that leads to the student's chosen distractor. This trajectory serves as the foundation for analyzing the student's misconceptions. We then outline the MCTS node definition and its four phases—selection, expansion, simulation, and backpropagation—with pseudocode and a detailed illustration of the whole MCTS process provided in Appendix C.

Node Definition. In our framework, each node in the reasoning tree represents a single reasoning step, with the intermediate result being either correct or incorrect. However, errors don't always result from explicit mistakes. Some students may follow correct reasoning steps but stop prematurely, leading to incorrect conclusions. For example, as illustrated in Figure 2, a student solving a relation-counting problem might correctly compute $2 \times 3 = 6$, but mistakenly stop there, missing the crucial next step of calculating 2^6.

To account for such behaviors, we define two types of nodes: *reasoning nodes* and *terminal nodes*. Each reasoning node represents a step in the reasoning process, followed by a terminal node that

marks where the reasoning stops. Terminal nodes indicate the end of a reasoning and are not further expanded. This design allows variable-length reasoning paths, helping to capture a wide range of student error patterns.

Selection. The selection phase begins at the root node and recursively selects the best child until reaching one of the conditions: a terminal node, a node that is not fully expanded (defined in the Expansion phase), or the maximum tree depth D.

Following the standard MCTS strategy used in board games, each node n_k maintains two statistics: V, the number of visits, and Q, the accumulated reward from simulations. At each selection, the Upper Confidence Bound for Trees (UCT) formula is applied to choose the best child node:

$$UCT(n_k) = \frac{n_k.Q}{n_k.V} + c\sqrt{\frac{\log(n.V)}{n_k.V}} \quad (2)$$

where n is the parent node of n_k and c is a constant, typically set to $\sqrt{2}$.

Expansion. The expansion phase extends the selected reasoning node by generating its child nodes. Since LLMs can theoretically generate an unlimited number of next steps (Chen et al., 2024a), we limit each reasoning node to at most B children ($B \geq 2$), excluding its terminal child, to maintain search efficiency. One of these children represents the correct next step, assuming flawless reasoning, while the remaining $B-1$ are plausible but incorrect alternatives. To avoid redundant error steps, all $B-1$ incorrect children are generated together in a single expansion. This means all children of a node are generated when it is first expanded. Thus, a reasoning node is considered not fully expanded if any of its non-terminal children remains unvisited. In such cases, one unvisited child is randomly selected for further simulation.

Simulation. The simulation phase completes a full reasoning trajectory starting from the selected child node. Given the current partial reasoning path, the model continues to generate subsequent steps until a final answer is reached, simulating how a student might reason from that point. The resulting answer is then passed to the backpropagation phase for trajectory evaluation.

Backpropagation. The backpropagation phase evaluates and propagates rewards for two types of reasoning trajectories: (1) the full trajectory extended through simulation, and (2) the partial trajectory ending at the selected node without further simulation. Each trajectory is evaluated based on two criteria. The first is *match score* r_1, which indicates whether the simulated final answer matches the selected distractor:

$$r_1 = 1 \ \ if \ \ d = d_i \ \ else \ \ 0 \quad (3)$$

where d is the simulated final answer and d_i is the selected distractor. While essential, it is not sufficient on its own, as early in the search, the correct distractor may not yet be reached, yielding zero reward and limited exploration. Additionally, even when the answer matches, the reasoning might still be implausible.

To address this, we introduce the *plausibility score* $r_2 \in [0, 1]$, which is obtained by prompting the model ϕ_{eval} to assess the trajectory's coherence and realism. The final reward is a weighted sum: $r = r_1 + \alpha r_2$, where α controls the influence of the plausibility score. This reward is then propagated backward to update statistics of each node along the trajectory.

After running MCTS for L iterations, we select the most plausible trajectory t_i that leads to the selected distractor. Combined with the associated knowledge concepts, this trajectory captures both the reasoning path and the underlying misconception, which are stored in the student's error pattern library to form their misconception prototype.

4.2.3 Error Pattern Summary

By applying MCTS-based reasoning recovery to a student's past QA records, we extract a set of knowledge concepts along with their corresponding erroneous reasoning trajectories. For each concept v_k, we aggregate its associated trajectories $[t_{k,1}, t_{k,2}, ...]$ and summarize them into a generalized misconception m_k using a summarization model ϕ_{sum}:

$$m_k = \phi_{sum}(v_k, [t_{k,1}, t_{k,2}, ...]) \quad (4)$$

This misconception abstracts the specific context of individual questions, allowing the misconception to be generalized across new problems that involve the same concept. The resulting set of concept–misconception pairs forms a personalized misconception prototype, which guides the generation of targeted distractors for future questions.

4.3 Answer Prediction

This stage simulates how the student might incorrectly reason through a new question based on

Methods	LLaMA-3-70B			Deepseek-V3			Claude-4-Sonnet			GPT-3.5-turbo			GPT-4o		
	Acc	Plaus	Coh	Acc	Plaus	Coh	Acc	Plaus	Coh	Acc	Plaus	Coh	Acc	Plaus	Coh
Eedi_100															
Random	10.2	2.66	2.31	25.8	3.1	2.63	22.3	3.36	2.96	11.0	2.56	2.13	18.4	2.71	2.32
Similarity	12.7	2.8	2.43	24.2	3.15	2.71	22.8	3.44	3.09	12.2	2.62	2.22	22.9	2.95	2.55
Level	15.7	2.5	2.22	25.1	2.8	2.5	23.0	2.92	2.66	16.6	2.39	2.11	25.4	2.67	2.39
Level+Random	13.7	2.43	2.17	24.8	2.75	2.44	24.0	2.88	2.63	12.9	2.16	1.9	24.7	2.65	2.34
Level+Similarity	13.7	2.43	2.19	24.7	2.75	2.44	23.7	2.83	2.56	13.1	2.16	1.88	23.9	2.63	2.33
Mastery	14.0	2.68	2.21	25.5	3.19	2.7	23.5	3.34	2.82	15.5	2.74	2.28	25.4	3.06	2.61
IO	14.2	2.74	2.3	26.4	3.29	2.83	24.1	3.51	3.1	17.4	2.73	2.25	24.4	3.0	2.53
CoT	16.2	2.74	2.27	27.1	3.35	2.83	25.3	3.52	3.12	14.9	2.73	2.25	23.7	3.0	2.54
Ours	**17.7**	**2.93**	**2.45**	**32.1**	**3.56**	**2.98**	**27.0**	**3.72**	**3.31**	**22.1**	**3.12**	**2.53**	**28.2**	**3.17**	**2.64**
Discrete_40															
Random	9.0	2.17	1.78	13.5	2.54	2.07	13.5	2.91	2.48	15.5	2.45	2.04	14.3	2.49	2.08
Similarity	11.4	2.3	1.92	16.7	2.62	2.12	15.1	2.91	2.39	15.1	2.54	2.07	12.7	2.63	2.24
Level	14.7	2.34	1.93	11.0	2.49	2.08	14.7	2.75	2.38	13.9	2.36	1.9	15.5	2.48	2.09
Level+Random	15.1	2.35	1.96	15.5	2.58	2.18	14.7	2.67	2.25	15.5	2.36	1.98	15.1	2.51	2.16
Level+Similarity	13.1	2.3	1.88	10.6	2.44	1.99	18.4	2.79	2.34	13.9	2.29	1.89	13.9	2.44	2.11
Mastery	17.1	2.55	2.05	22.9	2.82	2.27	26.5	2.9	2.31	18.8	2.6	2.07	17.1	2.76	2.31
IO	18.4	2.54	2.11	22.0	2.95	2.33	26.5	3.04	2.47	18.0	2.6	2.1	16.3	2.69	2.26
CoT	16.7	2.59	2.15	21.2	2.96	2.38	27.8	2.99	2.42	18.0	2.64	2.16	18.0	2.72	2.26
Ours	**28.6**	**2.82**	**2.33**	**34.7**	**3.24**	**2.69**	**40.4**	**3.32**	**2.76**	**28.6**	**2.97**	**2.38**	**31.0**	**3.05**	**2.52**

Table 1: Performance comparison between our method and baselines. The best results are **bolded**.

prior misconceptions, generating an erroneous answer that serves as a personalized distractor. Traditional approaches typically identify similar questions based on question stems and use the errors made on these past questions to predict potential mistakes on new ones. However, this text-level similarity can be misleading, as questions with similar stems may test entirely different knowledge concepts, resulting in distractors that fail to reflect the student's true weaknesses.

To overcome this, we shift to a concept-level approach. Instead of relying on question stems, we focus on the student's misconception prototype, which summarizes reasoning errors based on the knowledge concepts involved. Given a new question s_j, we first identify its relevant concepts, then retrieve the corresponding misconceptions $[m_{j,1}, m_{j,2}, ...]$ from the prototype. These generalized misconceptions are then applied to the new question s_j to simulate a more accurate reasoning:

$$\hat{d}_j = \phi_{pred}(s_j, a_j, [m_{j,1}, m_{j,2}, ...]) \quad (5)$$

where a_j denotes the correct answer, ϕ_{pred} denotes the model. The output $\hat{d}_j$ serves as a personalized distractor, reflecting the student's specific reasoning errors. By tailoring distractors to individual misconceptions, this method ensures both their plausibility and diagnostic value.

Parameter	M (avg)	N (avg)	D	B	c	L (Eedi_100)	L (Discrete_40)	α
Value	49	9	5	3	$\sqrt{2}$	10	20	0.2

Table 2: Key parameter settings of our method.

5 Experiments

5.1 Experiment Setup

5.1.1 Datasets

Given the substantial computational and financial costs for running extensive experiments on the `Student_1361` dataset—with around 100 different experimental settings, as discussed in Section 5.2 and 5.3—we conduct our analytical experiments on two subsets: `Eedi_100`, which contains 100 students focusing on elementary math, and `Discrete_40`, which covers 40 undergraduate students working on discrete mathematics. These subsets allow for comprehensive testing while ensuring feasibility. For the main comparison, we extend our analysis to `Student_1361`, where we compare our method with the second-best baseline, ensuring that our results are both robust and representative as discussed in Section 5.3.

5.1.2 Parameter Settings

We list key parameter settings of our framework in Table 2. Here, M and N denote the average number of past and test QA records per student, respectively. The number of MCTS iterations L is

Methods	LLaMA-3-70B			Deepseek-V3			Claude-4-Sonnet			GPT-3.5-turbo			GPT-4o		
	Acc	Plaus	Coh	Acc	Plaus	Coh	Acc	Plaus	Coh	Acc	Plaus	Coh	Acc	Plaus	Coh
Eedi_100															
Full Method	17.7	2.93	2.45	32.1	3.56	2.98	27.0	3.72	3.31	22.1	3.12	2.53	28.2	3.17	2.64
w/o $\phi_{concept}$	14.3	2.86	2.34	30.9	3.53	2.96	22.9	3.58	3.06	14.4	2.89	2.36	22.8	3.04	2.51
w/o Terminal Node	17.4	2.9	2.44	28.1	3.51	2.89	26.1	3.68	3.22	17.4	3.06	2.45	25.8	3.1	2.57
w/o ϕ_{eval}	16.6	2.89	2.4	27.9	3.49	2.86	26.3	3.67	3.27	16.8	3.02	2.49	25.7	3.11	2.59
w/o ϕ_{sum}	17.3	2.91	2.39	28.9	3.49	2.87	26.5	3.66	3.17	20.6	3.1	2.48	27.1	3.14	2.6
Discrete_40															
Full Method	28.6	2.82	2.33	34.7	3.24	2.69	40.4	3.32	2.76	28.6	2.97	2.38	31.0	3.05	2.52
w/o $\phi_{concept}$	19.6	2.69	2.23	22.4	3.06	2.43	26.5	3.15	2.51	19.6	2.79	2.23	20.4	2.96	2.35
w/o Terminal Node	25.7	2.71	2.31	30.2	3.14	2.54	38.8	3.18	2.6	26.5	2.81	2.35	28.2	2.97	2.47
w/o ϕ_{eval}	22.4	2.71	2.23	31.0	3.16	2.56	31.8	3.09	2.58	24.5	2.88	2.28	24.1	2.9	2.38
w/o ϕ_{sum}	26.1	2.72	2.3	30.2	3.18	2.6	36.7	3.25	2.64	27.8	2.92	2.38	29.8	3.01	2.49

Table 3: Ablation results on `Eedi_100` and `Discrete_40` dataset across five LLMs.

set based on subject complexity, with reasoning-intensive datasets using larger L values. We provide a detailed analysis of this choice in Section 5.3, and report ablation studies on other parameters in Appendix D.

To ensure model-agnostic evaluation, we use the same underlying LLM for all components of our framework (*e.g.*, $\phi_{concept}$, ϕ_{sum}, etc.). We experiment with five representative models covering different architectures and scales: two open-source models (*LLaMA-3-70B* (Team, 2024) and *DeepSeek-V3* (DeepSeek-AI, 2024)) and three close-source ones (*Claude-4-Sonnet* (Anthropic, 2024), *GPT-3.5-turbo* (Ouyang et al., 2022), and *GPT-4o* (OpenAI, 2023)).

5.1.3 Baselines

We compare our method with six heuristic baselines that do not model student reasoning trajectories and two reasoning-aware baselines that attempt to reconstruct student thinking. Heuristic baselines include: 1) *Random*: randomly selects a past QA record as reference; 2) *Similarity*: retrieves the most similar record based on stem embeddings; 3) *Level* (Benedetto et al., 2024): estimates the student's cognitive level from past QA accuracy; 4) *Level+Random*: combines Level-based estimation with random reference selection; 5) *Level+Similarity*: combines Level-based estimation with similarity-based retrieval; 6) *Mastery* (Wu et al., 2025): models the student's conceptual mastery but ignores misconceptions. Reasoning-aware baselines include: 7) *IO* (Yao et al., 2023): directly simulates possible student error trajectories; 8) *CoT* (Wei et al., 2022): uses a chain-of-thought prompting strategy to emulate reasoning.

5.1.4 Metrics

We evaluate performance using three metrics. The first is Accuracy (*Acc*), which measures whether the generated distractor matches the student's selected one. [2] The other two are LLM-based metrics that assess the quality of the generated reasoning trajectories: Plausibility (*Plaus*), which reflects how realistic the reasoning is as a student response, and Coherence (*Coh*), which measures its logical consistency and fluency. Both Plaus and Coh are scored on a 1–5 scale (higher is better), with the advanced o3-mini model serving as the evaluator.

Additional experimental settings and results are provided in Appendix D, including further evidence that our method can also generate high-quality group-level distractors.

5.2 Performance Comparison

As shown in Table 1, our method consistently outperforms all baselines across both datasets and model backbones, achieving significant improvements in all three metrics. These results highlight the effectiveness of our approach in generating plausible personalized distractors that align with realistic student misconceptions. Additionally, more powerful LLMs, such as Claude-4-Sonnet, lead to better performance, indicating that stronger models provide richer search spaces and more coherent reasoning candidates during the MCTS process.

Beyond performance improvements, the results also reveal two innovative findings:

1) **Explicit reasoning trajectory recovery is essential.** Baselines that simulate reasoning trajectories (*e.g.*, IO, CoT) outperform heuristic methods

[2] We use `math-verify` (Hugging Face, 2025) to compare generated distractors with the student selected ones.

Subsets	Eedi_700			Discrete_106			Python_192		
	Acc	Plaus	Coh	Acc	Plaus	Coh	Acc	Plaus	Coh
CoT	15.6	3.16	2.61	20.4	2.6	2.18	17.5	2.45	2.29
Ours	21.0	3.47	2.9	26.5	3.23	2.64	20.3	2.69	2.39
Subsets	Java_200			CP_39			OS_124		
	Acc	Plaus	Coh	Acc	Plaus	Coh	Acc	Plaus	Coh
CoT	14.9	2.72	2.18	3.6	2.64	2.52	11.0	2.64	2.06
Ours	17.5	2.77	2.21	6.7	2.78	2.67	12.4	2.68	2.16

Table 4: Performance comparison on `Student_1361`.

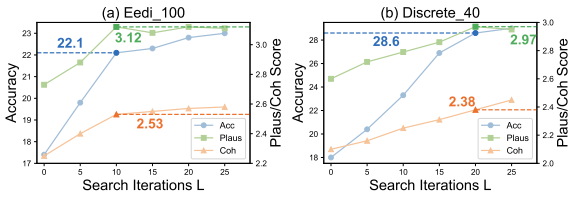


Figure 3: Results of different MCTS search iterations.

in most cases, emphasizing the importance of modeling how students arrive at incorrect answers. This is especially critical in MCQ settings, where the selected choice itself offers limited diagnostic value.

2) **MCTS excels in reasoning-intensive scenarios.** The advantage of our method is especially prominent on `Discrete_40`, where questions require more complex reasoning. While single-pass methods struggle to recover realistic trajectories, MCTS's iterative exploration enables more accurate reconstruction of plausible misconceptions.

5.3 In-Depth Analysis

We further validate four vital issues as follows.

5.3.1 Extended Evaluation Across `Student_1361`

We further evaluate our approach on the entire `Student_1361` dataset, comparing it with the second-best baseline, CoT. For this comparison, we use the GPT-3.5-Turbo model, with $L = 10$ for the `Eedi_700` subset and $L = 20$ for all other subsets. As shown in Table 4, our method outperforms CoT across all subjects, demonstrating the robustness and generalizability of our approach.

5.3.2 Effectiveness of Each Component

We perform ablation studies to evaluate the contribution of each core component in our two-stage framework. 1) Replacing the concept extractor $\phi_{concept}$ with simple stem-level similarity causes a substantial drop in performance. This is likely due to the limitation of surface-level retrieval in math problems, where similar wording often masks different underlying concepts. This confirms *the necessity of concept-level reasoning for accurate misconception analysis.* 2) Removing terminal nodes disables the modeling of truncated reasoning paths—common in real student errors where reasoning halts prematurely. The drop in performance demonstrates their value in capturing diverse student reasoning errors. This validates *the importance of explicitly modeling early termination.* 3) Eliminating the plausibility signal ϕ_{eval} from the reward function leaves only sparse answer-matching supervision, especially ineffective during early search stages. This underscores *the role of plausibility in shaping effective search guidance.* 4) Feeding raw reasoning trajectories into the prediction model leads to degraded performance, likely due to information overload and redundancy from key misconceptions. This shows *the benefit of distilling error patterns via trajectory summarization.*

5.3.3 Impact of MCTS Search Iterations

We examine the impact of varying MCTS search iterations L using GPT-3.5-turbo, as shown in Figure 3. On the `Eedi_100` dataset, all evaluation metrics improve up to L=10, after which performance plateaus or slightly declines, indicating that shallow searches are sufficient for simple problems and deeper exploration may introduce noise. In contrast, on the more challenging `Discrete_40` dataset, performance continues to improve steadily up to L=20, demonstrating the need for deeper exploration in reasoning-intensive tasks. These trends support our default choice of L values for different datasets.

5.3.4 Human Evaluation

We further conduct a human evaluation study on `Discrete_40` to assess the plausibility of the distractors generated by our method. Considering the large number of experimental settings and the cost of manual evaluation, we compare our method with the strongest competing baseline, CoT, which is the second-best-performing method according to the LLM-based evaluation results in Table 1. This choice enables a focused comparison against a strong reference model.

We recruit 20 undergraduate students with solid backgrounds in Discrete Mathematics as evaluators. Following the same evaluation criteria as the LLM-based setting, they rate each simulated solution in terms of *Plaus* and *Coh*. Each distractor is independently evaluated by two raters, and the final

Methods	LLaMA-3-70B		Deepseek-V3		Claude-4-Sonnet		GPT-3.5-turbo		GPT-4o	
	Plaus	Coh	Plaus	Coh	Plaus	Coh	Plaus	Coh	Plaus	Coh
CoT	2.54	2.13	2.89	2.27	2.85	2.39	2.64	2.12	2.72	2.25
Ours	2.77	2.32	3.16	2.59	3.25	2.64	2.81	2.23	2.88	2.43

Table 5: Human evaluation of the generated distractors on `Discrete_40`.

Group	Error Rate (Discrete_40)	Error Rate (CEval)	Abstain Rate	PD Rate
Control	0.22	0.12	0.14	-
Treatment	0.23	0.21	0.06	0.4

Table 6: Results of the user study, comparing original CEval distractors for the control group with personalized distractors for the treatment group.

score is computed as the average of their ratings.

As shown in Table 5, our method consistently receives higher human evaluation scores than CoT across all model settings. This alignment between human judgments and LLM-based evaluation provides evidence that the distractors generated by our method are more plausible and coherent.

5.4 User Study and Case Analysis

To empirically evaluate the effectiveness of personalized distractors, we conduct a user study with 30 students sampled from `Discrete_40`. These students are required to answer new questions selected from a held-out dataset, CEval Discrete Math (Huang et al., 2023). As shown in Table 6, the students are divided into two groups with comparable error rates on `Discrete_40`: a control group, which receives original CEval questions with default distractors, and a treatment group, which receives modified distractors, including one personalized distractor (generated specifically for each student by our method using Claude-4-Sonnet) and two frequent distractors (selected from the most common predicted distractors across all students).

To better assess whether the distractors truly reflect students' misconceptions, we add an additional choice to each question: "*None of the above*". Selecting this option means none of the distractors reflect the student's actual misconceptions, indicating insufficient diagnostic effectiveness.

We evaluate the performance using three metrics: 1) *Error Rate*: the proportion of responses selecting a distractor (excluding the correct answer and "*None of the above*"); 2) *Abstain Rate*: the proportion of responses selecting "*None of the above*"; 3) *PD Rate*: among all incorrect responses, the proportion that selects the personalized distractor.

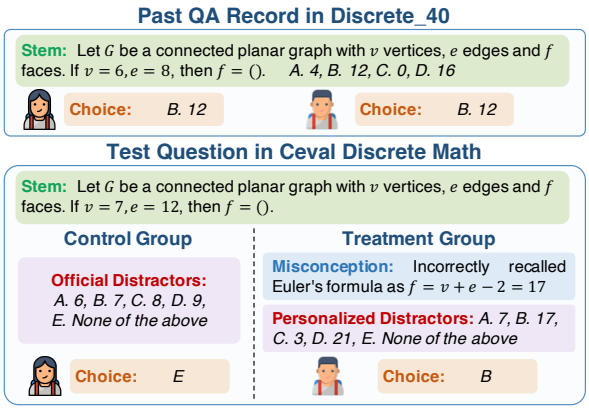


Figure 4: The control-group student chooses "*None of the above*", because none of the default group-level distractors reflects their actual misconception. In contrast, the treatment-group student selects the personalized distractor that accurately reflects the misconception.

Results in Table 6 show that the treatment group exhibits a higher error rate, lower abstain rate, and substantial PD rate, suggesting that personalized distractors effectively align with students' misconceptions and improve diagnostic effectiveness.

Figure 4 presents a representative case for this effect. Two students had previously misunderstood Euler's formula and selected the same incorrect answer. When presented with a similar question, the control group student selects "*None of the above*", as none of the default distractors matches the misconception. In contrast, the treatment group student chooses the personalized distractor, which accurately reflects his prior misunderstanding, demonstrating the advantage of personalization in targeting specific cognitive errors.

6 Conclusion

We propose the novel task of personalized distractor generation to precisely diagnose student misconceptions. To address sparse QA data and missing reasoning traces, we introduce a training-free MCTS-based framework that reconstructs student reasoning and generates tailored distractors. Experiments show that our method produces more accurate distractors, and the user study demonstrates the diagnostic value of personalized distractors.

Limitations

In this section, we discuss the limitations of our work as follow:

- Our framework relies on past QA records to support reliable personalization, and performance may be less strongly individualized for students with extremely sparse data. However, the system does not break down in such cases; it naturally falls back to concept-level or group-level distractors and improves as more records accumulate.
- The reasoning trajectories reconstructed by LLM-guided MCTS might not exactly replicate students' true cognitive processes; rather, they represent plausible approximations inferred from their past responses. Although such approximations may occasionally diverge from the student's actual reasoning at a fine-grained level, our experiments show that the generated distractors remain pedagogically meaningful and effective for diagnosing misconceptions. Future work may further improve faithfulness by incorporating richer behavioral signals or confidence-aware filtering when constructing misconception prototypes.
- Although our framework involves multiple model components (*e.g.*, different ϕ-modules) and may appear structurally complex, it remains computationally and financially feasible in practice. Many modules operate in lightweight or shared-inference settings, several stages can be parallelized, and the system scales smoothly under realistic workloads. We provide detailed analyses in Appendix C.3 and C.4.

Ethics Statement

`Student_1361` uses only pre-existing or anonymized data sources. The `Eedi_700` subset is sourced from the publicly available Eedi dataset and contains no personally identifiable information. The other 5 subsets are constructed from data on an online platform. To ensure privacy, we remove all sensitive information and retain only anonymized student IDs. Additionally, we confirm that all user-generated data is strictly authorized under the platform's *terms of service* during user registration process.

The user study was reviewed and approved by an independent institutional ethics committee prior to data collection. All participants were adults and provided informed consent before taking part in the study, and participation was voluntary with the option to withdraw at any time. No personally identifiable information was collected, and all data were anonymized and used solely for research purposes.

Acknowledgements

This research was supported by grants from the "Pioneer" and "Leading Goose" R&D Program of Zhejiang under Grant No. 2025C02022 and the National Natural Science Foundation of China (No.62307032).

References

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2024: Student Research Workshop, St. Julian's, Malta, March 21-22, 2024*, pages 225–237. Association for Computational Linguistics.

Elaf Alhazmi, Quan Sheng, Wei Emma Zhang, Munazza Zaib, and Ahoud Alhazmi. 2024. Distractor generation in multiple-choice tasks: A survey of methods, datasets, and evaluation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 14437–14458. Association for Computational Linguistics.

Suyeong An, Junghoon Kim, Minsam Kim, and Juneyoung Park. 2022. No task left behind: Multi-task learning of knowledge tracing and option tracing for better student assessment. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 4424–4431. AAAI Press.

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku.

Luca Benedetto, Giovanni Aradelli, Antonia Donvito, Alberto Lucchetti, Andrea Cappelli, and Paula Buttery. 2024. Using llms to simulate students' responses to exam questions. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 11351–11368. Association for Computational Linguistics.

Giorgio Biancini, Alessio Ferrato, and Carla Limongelli. 2024. Multiple-choice question generation using large language models: Methodology and educator insights. In *Adjunct Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization, UMAP Adjunct 2024, Cagliari, Italy, July 1-4, 2024*. ACM.

Semere Kiros Bitew, Johannes Deleu, Chris Develder, and Thomas Demeester. 2023. Distractor generation for multiple-choice questions with predictive prompting and large language models. In *Machine Learning and Principles and Practice of Knowledge Discovery in Databases - International Workshops of ECML PKDD 2023, Turin, Italy, September 18-22, 2023, Revised Selected Papers, Part II*, volume 2134 of *Communications in Computer and Information Science*, pages 48–63. Springer.

Grace Byun and Jinho D. Choi. 2025. D-GEN: automatic distractor generation and evaluation for reliable assessment of generative models. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 3316–3349. Association for Computational Linguistics.

Devrim Çavusoglu, Seçil Sen, and Ulas Sert. 2024. Disgem: Distractor generation for multiple choice questions with span masking. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 9714–9732. Association for Computational Linguistics.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024a. Alphamath almost zero: Process supervision without process. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Jingyuan Chen, Tao Wu, Wei Ji, and Fei Wu. 2024b. Wisdombot: Tuning large language models with artificial intelligence knowledge. *Frontiers of Digital Education*, 1(2):159–170.

DeepSeek-AI. 2024. Deepseek-v3 technical report. *CoRR*, abs/2412.19437.

Zhiang Dong, Jingyuan Chen, and Fei Wu. 2025. Knowledge is power: Harnessing large language models for enhanced cognitive diagnosis. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 164–172. AAAI Press.

Wanyong Feng, Jaewook Lee, Hunter McNichols, Alexander Scarlatos, Digory Smith, Simon Woodhead, Nancy Otero Ornelas, and Andrew S. Lan. 2024. Exploring automated distractor generation for math multiple-choice questions via large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 3067–3082. Association for Computational Linguistics.

Nigel Fernandez, Alexander Scarlatos, Wanyong Feng, Simon Woodhead, and Andrew S. Lan. 2024. Divert: Distractor generation with variational errors represented as text for math multiple-choice questions. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 9063–9081. Association for Computational Linguistics.

Aritra Ghosh, Jay Raspat, and Andrew S. Lan. 2021. Option tracing: Beyond correctness analysis in knowledge tracing. In *Artificial Intelligence in Education - 22nd International Conference, AIED 2021, Utrecht, The Netherlands, June 14-18, 2021, Proceedings, Part I*, Lecture Notes in Computer Science, pages 137–149. Springer.

Ze Han, Yu Su, Qi Mo, and Xinyu Nie. 2025. Neural network student cognitive diagnosis model based on multi-relational graph. In *Proceedings of the 2025 2nd International Conference on Generative Artificial Intelligence and Information Security, GAIIS 2025, Hangzhou, China, February 21-23, 2025*, pages 515–521. ACM.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Zihan Huang, Tao Wu, Wang Lin, Shengyu Zhang, Jingyuan Chen, and Fei Wu. 2025. Autogeo: Automating geometric image dataset creation for enhanced geometry understanding. *IEEE Trans. Multim.*, 27:3105–3116.

Hugging Face. 2025. Math-Verify: A robust mathematical expression evaluation system. `https://github.com/huggingface/Math-Verify`. Accessed: 2025-05-15.

Paul Ngee Kiong Lau, Sie-Hoe Lau, Kian Sam Hong, and Hasbee Usop. 2011. Guessing, partial knowledge, and misconceptions in multiple-choice tests. *J. Educ. Technol. Soc.*, 14(4):99–110.

Yooseop Lee, Suin Kim, and Yohan Jo. 2025. Generating plausible distractors for multiple-choice questions via student choice prediction. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 23669–23692. Association for Computational Linguistics.

Ang Li, Yiquan Wu, Yinghao Hu, Lizhi Qing, Shihang Wang, Chengyuan Liu, Tao Wu, Adam Jatowt, Ming Cai, Fei Wu, and Kun Kuang. 2025. Coevo: Co-evolution of LLM and retrieval model for domain-specific information retrieval. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 14980–14999. Association for Computational Linguistics.

Mengze Li, Kairong Han, Jiahe Xu, Yueying Li, Tao Wu, Zhou Zhao, Jiaxu Miao, Shengyu Zhang, and Jingyuan Chen. 2024. Cross-modal observation hypothesis inference. In *Proceedings of the 32nd ACM International Conference on Multimedia, MM 2024, Melbourne, VIC, Australia, 28 October 2024 - 1 November 2024*, pages 466–475. ACM.

Chen Liang, Xiao Yang, Neisarg Dave, Drew Wham, Bart Pursel, and C. Lee Giles. 2018. Distractor generation for multiple choice questions using learning to rank. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications@NAACL-HLT 2018, New Orleans, LA, USA, June 5, 2018*, pages 284–290. Association for Computational Linguistics.

Zhenghao Lin, Zihao Tang, Xiao Liu, Yeyun Gong, Yi Cheng, Qi Chen, Hang Li, Ying Xin, Ziyue Yang, Kailai Yang, Yu Yan, Xiao Liang, Shuai Lu, Yiming Huang, Zheheng Luo, Lei Qu, Xuan Feng, Yaoxiang Wang, Yuqing Xia, Feiyang Chen, Yuting Jiang, Yasen Hu, Hao Ni, Binyang Li, Guoshuai Zhao, Jui-Hao Chiang, Zhongxin Guo, Chen Lin, Kun Kuang, Wenjie Li, Yelong Shen, Jian Jiao, Peng Cheng, and Mao Yang. 2025. Sigma: Differential rescaling of query, key and value for efficient language models. *CoRR*, abs/2501.13629.

Ali Malik, Mike Wu, Vrinda Vasavada, Jinpeng Song, Madison Coots, John Mitchell, Noah D. Goodman, and Chris Piech. 2021. Generative grading: Near human-level accuracy for automated feedback on richly structured problems. In *Proceedings of the 14th International Conference on Educational Data Mining, EDM 2021, virtual, June 29 - July 2, 2021*. International Educational Data Mining Society.

Sérgio Silva Mucciaccia, Thiago Meireles Paixão, Filipe Wall Mutz, Claudine Santos Badue, Alberto Ferreira de Souza, and Thiago Oliveira-Santos. 2025. Automatic multiple-choice question generation and evaluation systems based on LLM: A study case with university resolutions. In *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pages 2246–2260. Association for Computational Linguistics.

Bang Nguyen, Tingting Du, Mengxia Yu, Lawrence Angrave, and Meng Jiang. 2025. QG-SMS: enhancing test item analysis via student modeling and simulation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 26152–26168. Association for Computational Linguistics.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Nisarg Parikh, Nigel Fernandez, Alexander Scarlatos, Simon Woodhead, and Andrew S. Lan. 2025. Lookalike: Consistent distractor generation in math mcqs. *CoRR*, abs/2505.01903.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Alex Reinhart, Ciaran Evans, Amanda Luby, Josue Orellana, Mikaela Meyer, Jerzy Wieczorek, Peter Elliott, Philipp Burckhardt, and Rebecca Nugent. 2022. Think-aloud interviews: A tool for exploring student statistical reasoning. *Journal of Statistics and Data Science Education*, 30(2):100–113.

Alexander Scarlatos, Wanyong Feng, Andrew S. Lan, Simon Woodhead, and Digory Smith. 2024. Improving automated distractor generation for math multiple-choice questions with overgenerate-and-rank. In *Proceedings of the 19th Workshop on Innovative Use of NLP for Building Educational Applications, BEA 2024, Mexico City, Mexico, June 20, 2024*, pages 222–231. Association for Computational Linguistics.

Shashank Sonkar, Xinghe Chen, Naiming Liu, Richard G. Baraniuk, and Mrinmaya Sachan. 2024. Llm-based cognitive models of students with misconceptions. *CoRR*, abs/2410.12294.

KV Aditya Srivatsa, Kaushal Kumar Maurya, and Ekaterina Kochmar. 2025. Can llms reliably simulate real students' abilities in mathematics and reading comprehension? *CoRR*, abs/2507.08232.

Xinjie Sun, Qi Liu, Kai Zhang, Shuanghong Shen, Fei Wang, Yan Zhuang, Zheng Zhang, Weiyin Gong, Shijin Wang, Lina Yang, and Xingying Huo. 2025. HCD: A hierarchy constraint-aware neural cognitive diagnosis framework. *Neural Networks*, 190:107668.

Zihao Tang, Zheqi Lv, Shengyu Zhang, Fei Wu, and Kun Kuang. 2024. Modelgpt: Unleashing llm's capabilities for tailored model generation. *CoRR*, abs/2402.12408.

Zihao Tang, Xin Yu, Ziyu Xiao, Zengxuan Wen, Zelin Li, Jiaxi Zhou, Hualei Wang, Haohua Wang, Haizhen Huang, Weiwei Deng, Feng Sun, and Qi Zhang. 2026. Mnemis: Dual-route retrieval on hierarchical graphs for long-term LLM memory. *CoRR*, abs/2602.15313.

Llama Team. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Fei Wang, Weibo Gao, Qi Liu, Jiatong Li, Guanhao Zhao, Zheng Zhang, Zhenya Huang, Mengxiao Zhu, Shijin Wang, Wei Tong, and Enhong Chen. 2024a. A survey of models for cognitive diagnosis: New developments and future directions. *CoRR*, abs/2407.05458.

Qingsong Wang, Tao Wu, Wang Lin, Yueying Feng, Gongsheng Yuan, Chang Yao, and Jingyuan Chen. 2025a. Cognitive-level adaptive generation via capability-aware retrieval and style adaptation. In *Findings of the Association for Computational Linguistics: EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 11054–11069. Association for Computational Linguistics.

Rose E. Wang, Qingyang Zhang, Carly Robinson, Susanna Loeb, and Dorottya Demszky. 2024b. Bridging the novice-expert gap via models of decision-making: A case study on remediating math mistakes. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 2174–2199. Association for Computational Linguistics.

Shulei Wang, Wang Lin, Hai Huang, Hanting Wang, Sihang Cai, WenKang Han, Tao Jin, Jingyuan Chen, Jiacheng Sun, Jieming Zhu, and Zhou Zhao. 2025b. Towards transformer-based aligned generation with self-coherence guidance. *CoRR*, abs/2503.17675.

Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2021. Denoising implicit feedback for recommendation. In *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021*, pages 373–381. ACM.

Zichao Wang, Angus Lamb, Evgeny Saveliev, Pashmina Cameron, Yordan Zaykov, José Miguel Hernández-Lobato, Richard E. Turner, Richard G. Baraniuk, Craig Barton, Simon Peyton Jones, Simon Woodhead, and Cheng Zhang. 2020. Results and insights from diagnostic questions: The neurips 2020 education challenge. In *NeurIPS 2020 Competition and Demonstration Track, 6-12 December 2020, Virtual Event / Vancouver, BC, Canada*, volume 133 of *Proceedings of Machine Learning Research*, pages 191–205. PMLR.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Siyu Wu, Yang Cao, Jiajun Cui, Runze Li, Hong Qian, Bo Jiang, and Wei Zhang. 2024a. A comprehensive exploration of personalized learning in smart education: From student modeling to personalized recommendations. *CoRR*, abs/2402.01666.

Tao Wu, Jingyuan Chen, Wang Lin, Mengze Li, Yumeng Zhu, Ang Li, Kun Kuang, and Fei Wu. 2025. Embracing imperfection: Simulating students with diverse cognitive levels using llm-based agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 9887–9908. Association for Computational Linguistics.

Tao Wu, Mengze Li, Jingyuan Chen, Wei Ji, Wang Lin, Jinyang Gao, Kun Kuang, Zhou Zhao, and Fei Wu. 2024b. Semantic alignment for multimodal large language models. In *Proceedings of the 32nd ACM International Conference on Multimedia, MM 2024, Melbourne, VIC, Australia, 28 October 2024 - 1 November 2024*, pages 3489–3498. ACM.

Weicai Yan, Wang Lin, Zirun Guo, Ye Wang, Fangming Feng, Xiaoda Yang, Zehan Wang, and Tao Jin. 2025. Diff-prompt: Diffusion-driven prompt generator with mask supervision. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Rest-mcts*: LLM self-training via process reward guided tree search. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. 2024b. Accessing GPT-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *CoRR*, abs/2406.07394.

Yumeng Zhu, Caifeng Zhu, Tao Wu, Shulei Wang, Yiyun Zhou, Jingyuan Chen, Fei Wu, and Yan Li.

2025. Impact of assignment completion assisted by large language model-based chatbot on middle school students' learning. *Educ. Inf. Technol.*, 30(2):2429–2461.

In this appendix, we present the following content:

A Detailed Comparison Between Group-level and Personalized Distractor Generation

We summarize the main difference between group-level and personalized distractor generation as follows:

- **Objective and application scenario.** Group-level distractor generation aims to produce generic distractors applicable to a wide range of students. It is typically used in standardized testing or large-scale problem database construction. In contrast, personalized distractor generation focuses on generating student-specific incorrect options that reflect each learner's unique misconceptions. This makes it especially suitable for online learning (Chen et al., 2024b) and adaptive assessment (Wang et al., 2025a) settings.

- **Data requirements and input format.** Group-level distractor generation typically requires datasets with only the question stem and a fixed set of distractors. Some methods also leverage statistical signals from answer distributions, such as the proportion of students selecting each option (Scarlatos et al., 2024). Personalized distractor generation, on the other hand, requires a richer input: a sequence of past QA records for each student, including both the question stem and the specific distractor selected. This enables fine-grained modeling of individual cognitive patterns.

- **Generation strategy and cognitive modeling.** Group-level distractor generation often relies on extracting frequent error patterns from a large population, but it lacks the ability to capture individual differences in reasoning. Personalized distractor generation emphasizes reconstructing how each student reasons incorrectly, often by modeling their error trajectories and constructing a personalized misconception prototype to guide generation.

- **Evaluation methodology and objectivity.** In group-level distractor generation, datasets usually provide several pre-defined distractors per question, and evaluation is performed by measuring how many of the generated distractors match the ground truth (Feng et al., 2024). However, due to the diversity of student error types, fixed ground-truth distractors cannot cover all plausible mistakes. This makes the evaluation less objective. Some recent work proposes LLM-based evaluation system (Nguyen et al., 2025), but it incurs high cost. In contrast, personalized distractor generation has a clear and objective evaluation criterion: if the generated distractor matches the student's actual selected incorrect answer, it is considered correct. This offers a more direct

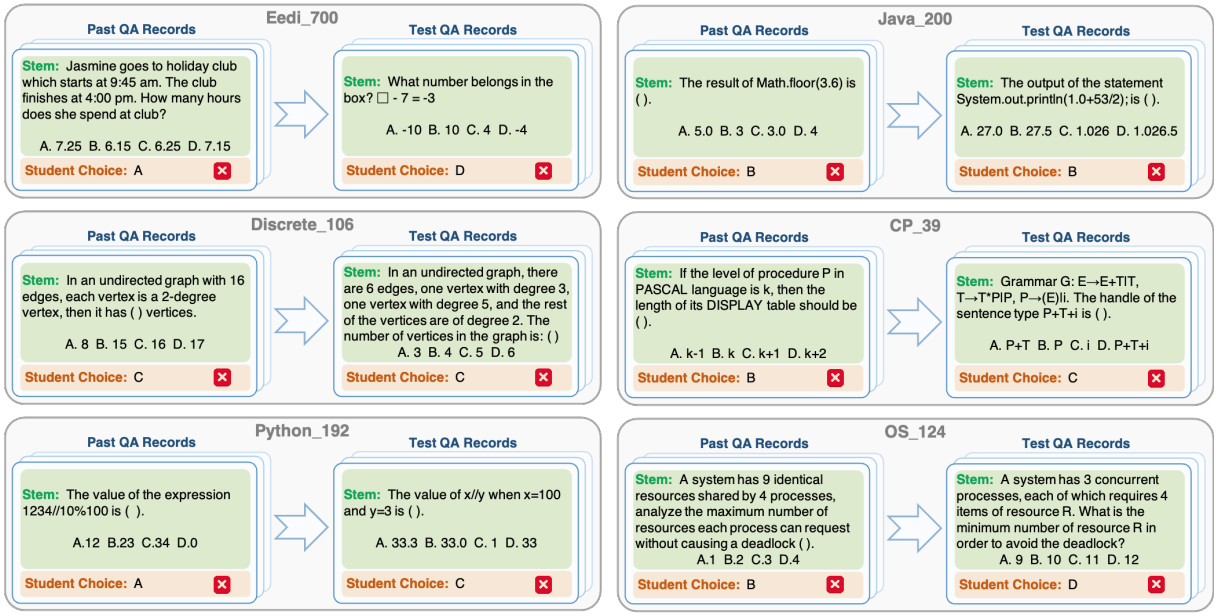


Figure 5: Sample illustrations of Student_1361.

Subset	Subject	Source	Difficulty	Student Number	M (avg)	N (avg)
Eedi_700	Math	Eedi (Wang et al., 2020)	Elementary	700	63	10
Discrete_106	Discrete Math	Online Platform	Undergraduate	106	37	10
Python_192	Python Programming	Online Platform	Undergraduate	192	47	10
Java_200	Java Programming	Online Platform	Undergraduate	200	32	10
CP_39	Compiler Principles	Online Platform	Undergraduate	39	17	5
OS_124	Operating System	Online Platform	Undergraduate	124	24	5

Table 7: Statistics of Student_1361.

and interpretable evaluation of distractor quality.

B Dataset Details

B.1 Dataset Statistics and Sample Illustrations

Our Student_1361 dataset covers a wide range of subjects and cognitive levels, spanning from elementary mathematics to multiple undergraduate domains, including discrete mathematics, Python programming, Java programming, compiler principles, and operating systems. In total, it contains 1,361 students, with each subset providing substantial QA records (on average 50 past records and 10 test records per student). This broad subject coverage, variation in difficulty, and large-scale student participation make the dataset highly diverse and representative, providing a rich foundation for studying personalized misconceptions and distractor generation. Detailed dataset statistics are reported in Table 7, and Figure 5 provides sample examples from each subset.

B.2 Invalid Data Criteria

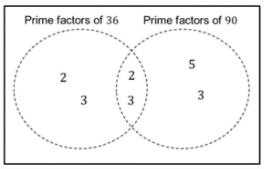
To ensure the suitability of each QA record for the distractor generation task, we apply strict filtering rules to exclude questions from the original data source that cannot support reasoning-based analysis or distractor construction. Specifically, a QA record is retained only if it meets the following conditions:

1. The question stem must be purely textual, without any diagrams or images, ensuring that the problem can be solved based solely on the text. We will further explore extending the framework to multimodal learning scenarios (Wu et al., 2024b; Huang et al., 2025) in future work.

2. The question must be deterministic, meaning the correct answer should be clearly inferable from the stem alone, without requiring the answer choices for disambiguation.

3. All answer choices must be numerical values, formulas, mathematical expressions or code.

Invalid Type 1. Contain Images



Here is the completed Venn diagram. What calculation works out the highest common factor of 36 and 90?

Invalid Type 2. Non-deterministic stem

Which of the following equivalence statements about biconditional connectives is incorrect?

A. P<->Q is equivalent to (P→Q) ∧ (Q→P)
B. P<->Q is equivalent to (P∧Q)∨(¯P∧¯Q)
C. P<->Q is equivalent to ¬P<->¬Q
D. P<->Q is equivalent to P<->¬Q

Invalid Type 3. Compound distractors

An undirected graph G has an Eulerian path if and only if ( ) ?
A. All nodes in G have degrees that are even numbers.
B. In G, there are at most two odd degree nodes.
C. G is connected and all nodes have even degrees.
D. G is connected and has at most two vertices with odd degree.

Figure 6: Illustrations of invalid data.

This restriction ensures objective and accurate evaluation, as existing tools like math-verify perform reliably only on purely mathematical content, while there is no robust method for comparing distractors that contain both natural language and math expressions.

We provide several invalid QA record samples in Figure 6.

C Method Details

C.1 Pseudocode of the MCTS algorithm

we provide pseudocode for the overall MCTS process (Algorithm 1), selection (Algorithm 2), expansion (Algorithm 3), simulation (Algorithm 4), backpropagation (Algorithm 5), function of judging whether a node is fully expanded (Algorithm 6), and function of collecting all the terminal node that match the student's actual chosen distractor after the MCTSeach (Algorithm 7).

Algorithm 1 MCTSearch for reasoning trajectory recovery

Input: Question stem s, student chosen distractor d, max search iteration L, max tree depth D, max children number B, model for generating correct next step $\phi_{correct}$, model for generating several incorrect next steps $\phi_{incorrect}$, model for simulation $\phi_{simulation}$, model for producing plausibility reward ϕ_{eval}, plausibility reward weight α, UCT constant c

Output: A plausible reasoning trajectory t that leads to d

$n_0 \leftarrow create\_node(depth = 0, t = None, r = None, is\_terminal = False, Q = 0, V = 0, child = None, parent = None)$
$l \leftarrow 1$
while $l \leq L$ **do**
  $n_l \leftarrow Selection(n_0, D, c)$
  $n_l \leftarrow Expansion(n_l, D, B, \phi_{correct}, \phi_{incorrect})$
  $t_l, d_l \leftarrow Simulation(n_l, \phi_{simulation})$
  $Backpropagation(n_l, \phi_{eval}, t_l, d_l, d, \alpha)$
end while
Initial empty list of terminal nodes that match the student chosen distractor $MNs = []$
$MNs \leftarrow search\_match\_nodes(n_0, d, MNs)$
if MNs has a node **then**
  $n^* \leftarrow \max(\frac{\hat{n}.Q}{\hat{n}.V})$ for $\hat{n}$ in MNs
else
  $n^* \leftarrow n_0$
  while n^* has a child **do**
    $n^* \leftarrow \max(\frac{\hat{n}.Q}{\hat{n}.V})$ for $\hat{n}$ in $n^*.child$
  end while
end if
return $t \leftarrow n^*.t$

Algorithm 2 Selection

Input: Root node n_0, max tree depth D, UCT constant c

Output: Selected not-fully-expanded node n_l

$n_l \leftarrow n_0$
while $is\_fully\_expanded(n_l)$ and $n_l.depth < D$ **do**
  $n_l \leftarrow \max(\frac{\hat{n}.Q}{\hat{n}.V} + c\sqrt{\frac{\log(n_l.V)}{\hat{n}.V}})$ for $\hat{n}$ in $n_l.child$
end while
return n_l

Algorithm 3 Expansion

Input: Selected node n_l, max tree depth D, max children number B, model for generating correct next step $\phi_{correct}$, model for generating several incorrect next steps $\phi_{incorrect}$, model for producing plausibility reward ϕ_{eval}
Output: Expanded node n_l

if $n_l.depth \geq D$ **then**
  return n_l
end if
if number of $n_l.child = 0$ **then**
  terminal node $\hat{n} \leftarrow create\_node($
  $depth = n_l.depth + 1, t = n_l.t,$
  $r = n_l.r, is\_terminal = True,$
  $Q = 0, V = 0, child = None,$
  $parent = n_l)$
  $n_l.child.append(\hat{n})$
  $Backpropagation(n_l.child[0], \phi_{eval}, n_l.t, n_l.r, d, \alpha)$
  $t_{correct}, r_{correct} \sim \phi_{correct}(n_l)$
  correct node $\hat{n} \leftarrow create\_node(depth = n_l.depth + 1, t = [n_l.t, t_{correct}], r = r_{correct}, is\_terminal = False, Q = 0, V = 0, child = None, parent = n_l)$
  $n_l.child.append(\hat{n})$
  $t^1_{incorrect}, r^1_{incorrect}, \ldots, t^{B-1}_{incorrect}, r^{B-1}_{incorrect} \sim \phi_{incorrect}(n_l)$
  for each $k \in [1, 2, \ldots, B-1]$ **do**
    incorrect node $\hat{n} \leftarrow create\_node($
    $depth = n_l.depth + 1,$
    $t = [n_l.t, t^k_{incorrect}], r = r^k_{incorrect},$
    $is\_terminal = False, Q = 0,$
    $V = 0, child = None,$
    $parent = n_l)$
    $n_l.child.append(\hat{n})$
  end for
end if
for each $k \in [1, 2, \ldots, B]$ **do**
  if $n_l.child[k].V = 0$ **then**
    $n_l \leftarrow n_l.child[k]$ **break**
  end if
end for
return n_l

Algorithm 4 Simulation

Input: Expanded node n_l, model for simulation $\phi_{simulation}$
Output: Simulated trajectory t_l and result d_l

$t_l, d_l \sim \phi_{simulation}(n_l)$
return t_l, d_l

Algorithm 5 Backpropagation

Input: Expanded node n_l, model for producing plausibility reward ϕ_{eval}, simulated trajectory t_l, simulated result d_l, student chosen distractor d, plausibility reward weight α

match reward $r_1 \leftarrow (d_l = d)$
plausibility reward $r_2 \sim \phi_{eval}(t_l)$
reward $r \leftarrow r_1 + \alpha r_2$
while n_l has a parent node **do**
  $n_l.Q \leftarrow n_l.Q + r$
  $n_l.V \leftarrow n_l.V + 1$
  $n_l \leftarrow n_l.parent$
end while

Algorithm 6 is_fully_expanded

Input: Node n
Output: Whether this node is fully expanded

if n is a terminal node **then**
  return True
end if
for $\hat{n} \in n.child$ **do**
  if $\hat{n}.V = 0$ **then**
    return False
  end if
end for
return True

Algorithm 7 search_match_nodes

Input: Current node n, student chosen distractor d, current list of matched terminal nodes MNs
Output: Updated MNs

if n is a terminal node and $n.r = d$ **then**
  $MNs.append(n)$
  return MNs
end if
for $\hat{n} \in n.child$ **do**
  $MNs \leftarrow search\_match\_nodes(\hat{n}, d, MNs)$
end for
return MNs

C.2 MCTS Process Details

We provide a detailed illustration of the MCTS-based reasoning trajectory reconstruction process in Figure 7. Starting from the root node, MCTS performs repeated cycles of selection, expansion, simulation, and backpropagation to explore plausible reasoning steps and recover the trajectory that best explains the student's chosen distractor. Be-

low, we highlight three key design principles in this process.

First, **although a terminal node inherits the reasoning process and intermediate result of its parent node, it does not inherit the parent's reward and must be evaluated separately.** A terminal node is evaluated at the moment when its parent node is expanded for the first time, *i.e.*, when the terminal node is first generated, as illustrated in step (1) and (7) of Figure 7. At this point, the terminal node represents the decision to stop at the current step and treat the intermediate result as the final answer.

Although it shares the same reasoning history as its parent, it cannot reuse the parent's reward, because the parent's reward is obtained through simulation that continues beyond the current step and reflects its *future potential*. In contrast, the terminal node involves no further simulation, as it corresponds to a completed "submit-answer" state. Therefore, its reward must be computed directly from the terminal output (*e.g.*, match and plausibility). After this one-time evaluation and backpropagation, the terminal node is never selected or expanded again, and its visit count remains fixed at $V = 1$, functioning as an absorbing end state rather than a search frontier.

Second, **the model $\phi_{correct}$ continues reasoning from the student's existing (possibly incorrect) trajectory, rather than regenerating a globally correct solution.** Our goal is to reconstruct how the student might have reasoned, instead of overwriting earlier mistakes. Accordingly, when $\phi_{correct}$ is invoked, it does not restart the reasoning process. Rather, it produces a logically coherent continuation that is conditioned on the existing partial reasoning—even if prior steps contain errors—thereby "continuing correctly from a wrong premise" while preserving the cognitive context of the misconception.

Third, **reaching the target distractor during simulation does not terminate the search; we continue exploring and later select the most plausible trajectory.** During simulation, the search may reach the target distractor at an early stage and form a complete reasoning chain, as shown in step (4) and (6) of Figure 7. However, such trajectories may still be implausible or inconsistent with realistic student behavior. Therefore, we do not stop once a trajectory matches the distractor. Instead, MCTS continues exploring alternative paths; after the search completes, multiple trajectories may reach the same distractor, and we select the one with the highest plausibility score as the final recovered reasoning path.

C.3 Scalability, Cost, and Computational Feasibility

Although our framework introduces a structured multi-stage pipeline rather than a single-pass generator, its computational and deployment cost remains controllable in practice. First, the approach does not depend on commercial LLMs: as shown in Table 1, open-source backbones such as LLaMA-3-70B achieve over 70% of the performance of the strongest commercial models (*i.e.*, Claude-4-Sonnet). This enables cost-efficient deployment by substituting commercial APIs with locally hosted models when desired, without modifying the framework design.

Second, most computation occurs during offline prototype construction, which is inherently parallelizable across students, questions, and tasks. In addition to model-side parallelism, the framework also supports batched and asynchronous API execution, allowing multiple reasoning calls to be processed concurrently. At inference time, the online cost is lightweight: misconception prototypes are reused, and only a small number of search steps are executed per new question.

Third, the computational scale of each component is explicitly bounded. Each student contributes only a limited number of past QA records within a short temporal window, resulting in tens of records rather than hundreds in our datasets. Likewise, the number of extracted knowledge concepts per question is capped to 15 to avoid redundancy, which constrains both retrieval and reasoning complexity. These design choices ensure that the overall search space remains compact and computationally manageable.

Taken together, these properties demonstrate that the proposed framework is scalable, cost-aware, and feasible for practical deployment, while retaining the interpretability and diagnostic benefits enabled by explicit reasoning-based personalization.

C.4 Error Attribution and Pipeline Traceability

To ensure that errors in a multi-stage pipeline can be analyzed and localized rather than being treated as opaque failures, our system is designed with explicit stage-level traceability. For every run, we

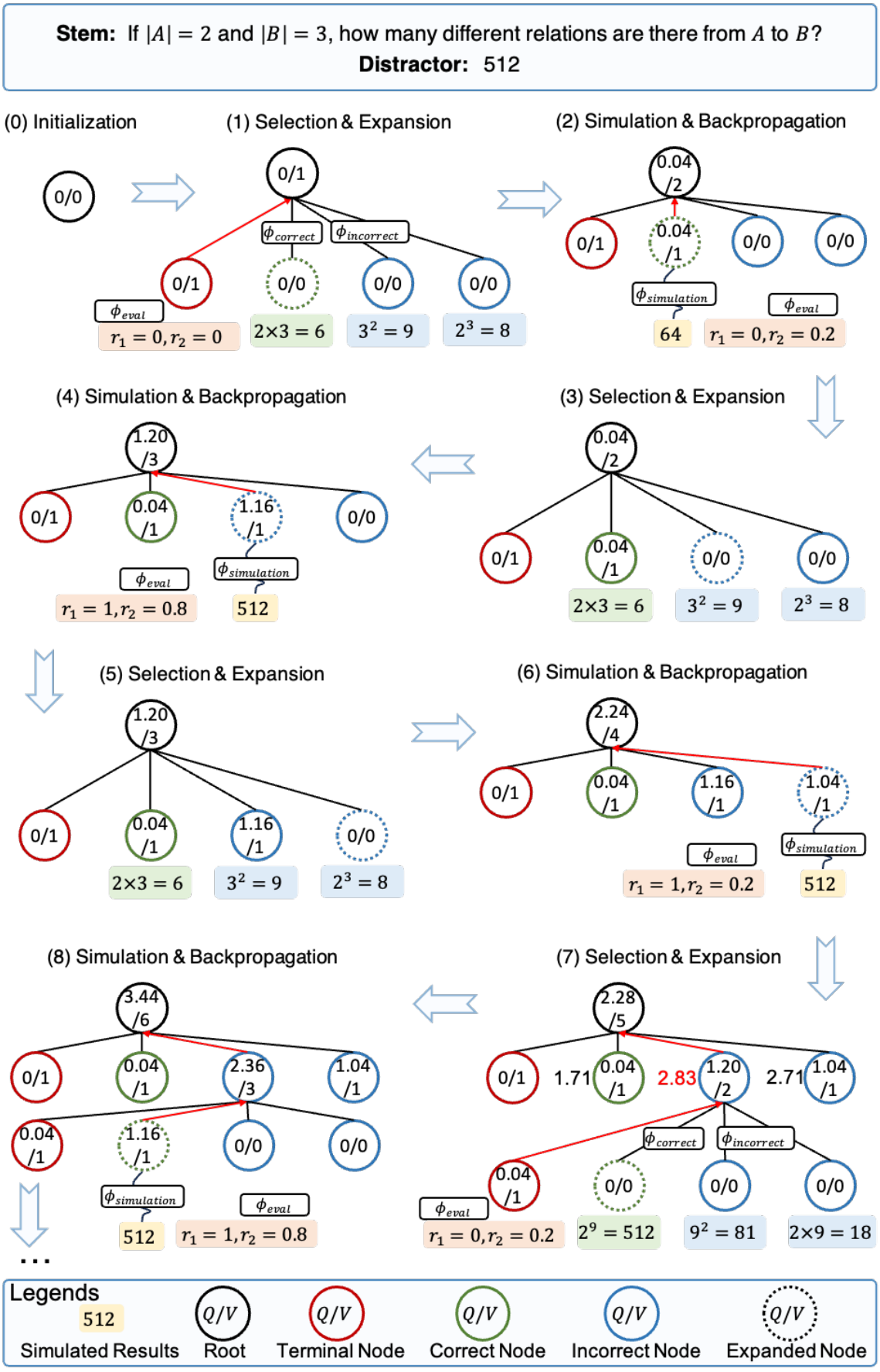


Figure 7: Illustration of the MCTS-based reasoning trajectory reconstruction process.

Level Prompt

You will be shown a math problem, and predict the answer of a student of level {level} to the problem. Please remind that a student's level ranges from 1 to 5, while 1 indicates the lowest ability and 5 indicates the highest ability.
Generate one wrong answer that this student is likely to get, with the corresponding reasoning process that lead to this wrong answer.

Figure 8: Details of the level prompt.

log each model invocation and store the outputs of all intermediate components, including concept extraction, trajectory reconstruction, misconception summarization, and distractor prediction. This allows the full reasoning chain to be replayed and inspected when an unexpected output occurs.

When an error is observed in the final distractor generation result, we can back-trace the pipeline to identify whether the failure originates from concept retrieval, reasoning reconstruction, prototype construction, or the final prediction stage. Each module can then be evaluated independently using its logged inputs and outputs, enabling targeted debugging rather than end-to-end trial-and-error adjustment. This design makes the system interpretable, diagnosable, and incrementally improvable, despite its multi-component structure.

D More Experimental Settings and Results

D.1 Model Details

LLaMA-3-70B (Team, 2024) is a 70B multilingual language model by Meta AI, released in Dec 2024. The Instruct version is tuned for dialogue and achieves strong performance across open- and closed-source benchmarks. The tested API version is *LLaMA-3-70B-Instruct*.

DeepSeek-V3 (DeepSeek-AI, 2024) is an open-source language model developed by DeepSeek, supporting both English and Chinese. It is optimized for instruction-following tasks and achieves competitive performance on a range of benchmarks.

Claude (Anthropic, 2024) is Anthropic's safety-focused language model designed for general-purpose tasks. The evaluated version is *claude-sonnet-4-20250514*, known for strong reasoning and alignment.

GPT-3.5 (Ouyang et al., 2022) is a widely used OpenAI model optimized for general natural language understanding and generation. The tested version is *gpt-3.5-turbo-0125*.

GPT-4o (OpenAI, 2023) is OpenAI's flagship model with advanced reasoning and multimodal support. The tested version is *gpt-4o-2024-11-20*.

Mastery Prompt

You are given a new question, and it is known that the student answered it incorrectly.

You are also given:

1. The knowledge concept this question tests.

2. The student's known mastery level of this concept.

Your task is to simulate the student's mistaken reasoning process based on their mastery level, and predict what incorrect answer the student most likely gave.
The reasoning and answer must strictly match the student's mastery level.

Output format (Python dict):
{
"simulated_reasoning": "<step-by-step reasoning that reflects the mastery level>",
"predicted_wrong_answer": "<final wrong answer, only the final numerical value or mathematical expression>"
}

Figure 9: Details of the mastery prompt.

D.2 Baseline Details

We compare our method with six heuristic baselines that do not model student reasoning trajectories and two reasoning-aware baselines that attempt to reconstruct student thinking:

1. *Random*, which randomly selects a record from past QA records as a reference.

2. *Similarity*, which selects a record based on task statement similarities.

3. *Level*, which estimates a student's ability level based on the accuracy of their past QA records. Specifically, we first compute the accuracy of a student's past QA records and then normalize it to a range of 1 to 5, where 5 represents the highest cognitive level. As mentioned in Benedetto et al. (2024), this relative cognitive level simulation approach yields better results. We incorporate this level into the prompt to indicate the student's proficiency, enabling the model to infer student misconception accordingly. The prompt is shown in Figure 8.

Input-Output (IO) Prompt

You are given a multiple-choice or open-ended question, along with the incorrect answer a student provided.

Your task is to reconstruct the likely reasoning process that led the student to this incorrect answer. Focus on identifying the specific cognitive misunderstanding or calculation mistake the student might have made. The explanation should follow the student's likely thinking path step-by-step, even if it's flawed.

Avoid general comments like "The student misunderstood." Be specific and concrete.

Output format:
{
"likely_reasoning": "<step-by-step explanation of how the student may have derived the wrong answer>",
}

Figure 10: Details of the IO prompt.

Chain-of-Thought (CoT) Prompt

You are given a multiple-choice or open-ended question, along with the incorrect answer a student provided.

Your task is to reconstruct the likely reasoning process that led the student to this incorrect answer. Focus on identifying the specific cognitive misunderstanding or calculation mistake the student might have made. The explanation should follow the student's likely thinking path step-by-step, even if it's flawed.

Avoid general comments like "The student misunderstood." Be specific and concrete.

Make a plan for the reasoning process.

Output format:
{
"plan": "<plan for the reasoning process>",
"likely_reasoning": "<step-by-step explanation of how the student may have derived the wrong answer>",
}

Figure 11: Details of the CoT prompt.

4. *Level+Random*, which incorporates *Random* and *Level*. We add the randomly selected past QA record into the *Level* prompt.

5. *Level+Similarity*, which incorporates *Random* and *Similarity*. We add the similarity-based retrieved past QA record into the *Level* prompt.

6. *Mastery* (Wu et al., 2025), which models the student's mastery over each extracted knowledge concept without inferring the underlying reasoning trajectory. As a result, it can only capture shallow information—such as whether the student understands a concept well or poorly—while failing to identify how the student is likely to make mistakes. The prompt is shown in Figure 9.

7. *IO* (Yao et al., 2023), which requires models to directly output the reasoning trajectories. This prompt is shown in Figure 10.

8. *CoT* (Wei et al., 2022), which introduces a chain of thoughts that connects the input to the output, with each thought forming a coherent language sequence that acts as a meaningful intermediate step toward solving the problem. This prompt is shown in Figure 11.

D.3 Metric Details

We propose two LLM-based metrics to evaluate the quality of generated reasoning trajectories. Specifically, we use the o3-mini model (API version: *o3-mini-2025-01-31*), which is an advanced open-source model, to ensure fair and consistent evaluation. Detailed prompt designs for these metrics are illustrated in Figure 12.

D.4 Ablation Study on Hyperparameter Choices

We conduct a sensitivity analysis to investigate how key parameters affect model performance. Specifically, we examine the impact of the MCTS tree depth D, the maximum number of child nodes per reasoning step B, and the weight α used to balance plausibility and match scores. The results are reported in Table 8, 9 and 10.

We find that varying the tree depth D has limited impact on performance. In practice, due to the fixed number of search iterations, the tree seldom grows to its maximum depth. However, setting D too small may harm performance in complex domains like Discrete Math, where reasoning often involves multiple intermediate steps.

For the child budget B, we observe that $B = 3$ yields the best results. A smaller value restricts the model's ability to explore diverse plausible mistakes, while a larger value spreads the search budget too thin across many shallow branches, limiting deeper exploration of each reasoning trajectory.

The coefficient α controls the weight of plausibility scores in the search. We find that $\alpha = 0.2$ achieves the best trade-off. When α is too small, plausibility receives little weight during node evaluation. As a result, in the early stages of search—when matched distractors are rarely en-

Evaluation Prompt

You are an expert math teacher evaluating whether a model-generated reasoning process could plausibly reflect a real student's thinking. You will be given:

1. A math question
2. A student's final answer
3. A reasoning process that explains how the student may have arrived at the answer

Note: The reasoning may contain errors — this is expected. Your task is not to judge whether the final answer is correct, but rather to evaluate the realism and coherence of the reasoning process.

You should score the reasoning on two dimensions, each from 1 to 5 (higher is better):

1. Plausibility (Realism of the Mistake)
Does the reasoning reflect a common, cognitively plausible mistake that a real student might make?

5 = Very realistic; reflects a well-known student misconception
4 = Realistic and understandable
3 = Somewhat plausible, though slightly uncommon
2 = Weakly plausible; seems unnatural
1 = Implausible or clearly artificial mistake

2. Causal Coherence (Logical Consistency)
Does the reasoning process follow a clear, step-by-step logic that leads to the final answer?

5 = Fully coherent, each step follows naturally
4 = Mostly coherent, small gaps
3 = Reasonably coherent, but with noticeable jumps
2 = Weakly coherent, hard to follow
1 = Incoherent or unrelated steps

Output Format (JSON):
{
"plausibility": <1–5>,
"causal_coherence": <1–5>,
"explanation": "<brief explanation of both scores>"
}

Figure 12: Details of the prompt for producing Plaus and Coh scores.

countered—the overall reward remains low across nodes, making it difficult to differentiate promising search paths. In this case, the search lacks meaningful guidance and becomes ineffective. Conversely, when α is too large, plausibility dominates over match scores, leading the model to prioritize fluent or logically sound paths that may not align with the student's actual misconceptions.

D	Eedi_100	Discrete_40
3	22.1	27.1
4	22.2	28.2
5	22.1	28.6
6	22.0	28.6
7	21.8	28.8

Table 8: Ablation results (Acc) on MCTS tree depth D.

D.5 Generalization to Group-level Distractor Generation

We further evaluate the generalizability of our framework in the group-level distractor generation setting. We compare it with three established baselines—D-GEN (Byun and Choi, 2025), O&A (Scarlatos et al., 2024), and DiVERT (Fernandez et al., 2024)—where O&A and DiVERT are trained on the Eedi dataset. All baseline methods generate a fixed set of three distractors per question.

In contrast, our method first operates at the personalized level. For each student, we use our framework to generate a personalized distractor for every question that matches the student's ability level (for example, students in `Eedi_100` answer only elementary-level questions, while students in `Discrete_40` are evaluated on discrete-math questions). This produces a pool of individualized distractors reflecting diverse student-specific misconceptions. We then aggregate all generated distractors across students on a per-question basis, compute their empirical frequency, and select the top three most frequent distractors as the final group-

B	Eedi_100	Discrete_40
2	20.7	26.7
3	22.1	28.6
4	21.0	29.0
5	20.5	26.1

Table 9: Ablation results (Acc) on maximum number of child nodes B.

α	Eedi_100	Discrete_40
0	16.8	24.5
0.1	21.1	28.2
0.2	22.1	28.6
0.3	22.1	27.8
0.5	21.7	25.9
1.0	20.3	23.7

Table 10: Ablation results (Acc) on plausibility score weight α.

level set. This aggregation reflects the intuition that misconceptions frequently observed across personalized cases correspond to shared misconception patterns at the group level.

We evaluate this process on questions from `Eedi_100` and `Discrete_40`, and additionally include two held-out datasets—Elementary Math from MMLU (Hendrycks et al., 2021) and Discrete Math from CEval (Huang et al., 2023)—to assess out-of-domain generalization. Performance is measured using Recall, which quantifies how well the generated distractors cover the actual distractors selected by students in real QA records (higher Recall indicates stronger coverage of real misconception behaviors).

Results in Table 11 show that baseline models perform well on in-domain questions but degrade substantially when evaluated on the held-out datasets. By contrast, our method maintains consistently strong performance across both in-domain and out-of-domain settings, demonstrating robustness and superior generalization. These findings suggest that personalization not only benefits individual diagnosis, but also provides an effective pathway for deriving high-quality group-level distractors from aggregated personalized misconceptions.

D.6 Analysis of Lower Acc Scores in `CP_39` and `OS_124`

We observe that the Acc values for Compiler Principles (`CP_39`) and Operating Systems (`OS_124`) are noticeably lower than those in the other subjects in Table 4. This does not primarily result from

Methods	Elementary Math		Discrete Math	
	Eedi_100	MMLU	Discrete_40	CEval
Group-level Distractor Generation Baselines				
D-GEN	20.48	18.28	28.57	28.89
O&A	26.63	22.22	20	18.89
DiVERT	31.66	25.09	20.71	20
Our Method				
LLaMA-3-70B	22.16	30.29	28.57	36.67
Deepseek-V3	31.1	34.95	36.43	40
Claude-4-Sonnet	28.31	32.44	34.29	45.56
GPT-3.5-turbo	24.39	27.96	30.71	44.44
GPT-4o	27.37	30.29	29.29	45.56

Table 11: Performance comparison on group-level distractor generation across four datasets.

poor generation quality, but rather from characteristics of these domains. First, although we have removed text-heavy options and retained only distractors consisting of numbers, expressions, or code, the remaining distractors in `CP_39` and `OS_124` are still substantially more complex than those in other subjects, often involving long or highly structured expressions. As a result, different students may express similar misconceptions through different but semantically related forms, making matching to the specific distractor previously selected by a student much more difficult. Second, misconceptions in these theoretical, concept-intensive subjects tend to be more diverse and less concentrated than in mathematics or programming tasks, where students' errors are often clustered around a few recurring patterns. Consequently, our model may generate a plausible distractor that accurately reflects a student's reasoning error, but it does not exactly coincide with the particular option chosen in the dataset, leading to lower Acc despite reasonable Plaus and Coh scores. These observations suggest that the drop in Acc mainly reflects the strictness of the exact-match metric under domains with heterogeneous misconception expressions, rather than a degradation in the model's ability to model student reasoning.

D.7 Evaluation of Intermediate Pipeline Outputs

Since our pipeline is relatively large and involves multiple interdependent components, we further evaluate the reliability of its intermediate outputs.

D.7.1 Evaluation of Knowledge Concept Extraction

In particular, we assess the accuracy of the knowledge concept extraction process on the

LLaMA-3-70B		Deepseek-V3		Claude-4-Sonnet		GPT-3.5-turbo		GPT-4o	
Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
0.7	0.76	0.74	0.75	0.65	0.83	0.71	0.77	0.72	0.75

Table 12: Evaluation of concept extraction accuracy of our method on the `Discrete_40` dataset.

Methods	LLaMA-3-70B	Deepseek-V3	Claude-4-Sonnet	GPT-3.5-turbo	GPT-4o
CoT	0.33	0.51	0.62	0.39	0.48
MCTS	0.46	0.6	0.74	0.5	0.63

Table 13: Mechanism match accuracy of MCTS vs. CoT across LLM backbones.

`Discrete_40` dataset, where each question is annotated with its corresponding concepts by the online learning platform. We treat these platform-provided annotations as ground truth and measure the precision and recall of the concepts extracted by our method. As shown in Table 12, our approach achieves consistently high precision and recall, indicating that the extracted concepts closely match the true conceptual structure of the questions and provide a reliable basis for subsequent reasoning reconstruction and distractor generation.

D.7.2 Evaluation of MCTS-reconstructed Reasoning

Empirically, we find that the MCTS-based reconstruction captures student error patterns to a meaningful extent. Specifically, we conduct additional validation for the MCTS reconstruction from both mechanistic alignment and counterfactual behavioral consistency perspectives.

Mechanistic alignment consistency. Importantly, the platform-sourced portion of the `Student_1361` dataset consists of high-quality MCQs authored by domain experts, and a subset of these questions includes expert-written error rationales for each distractor (*i.e.*, explicit descriptions of the underlying misconception or erroneous rule). Leveraging this resource, we randomly identify 30 questions in the `Discrete_40` subset that contain usable expert rationales, covering 90 distractors in total.

For each distractor, we run MCTS and CoT (as baseline) to obtain the highest-reward erroneous reasoning trajectory. Independent domain experts, who are blind to the source of the trajectories, then judge whether the key error mechanism implied by the reconstructed trajectory matches the expert-provided rationale (allowing differences in wording or intermediate steps as long as the core misconception is consistent). We report Mechanism Match Accuracy as the evaluation metric in Table 13. Across different LLM backbones, MCTS-based reconstruction achieves better agreement with expert rationales than CoT-based baselines, indicating that our method reliably recovers the typical error mechanisms underlying distractors.

Counterfactual behavioral consistency. To further provide behavioral evidence at the individual level, we design a counterfactual variant experiment. The core hypothesis is: if MCTS correctly reconstructs the erroneous reasoning process that led a student to select a particular distractor, then under an isomorphic problem variant (same reasoning structure with perturbed numerical values), the distractor recomputed from the same erroneous reasoning process should remain attractive to that student and thus be selected again with high probability.

Concretely, we sample 5 students from `Discrete_40` and select 5 historical incorrect questions per student (25 total), focusing on formula-driven problems with stable reasoning structures (*e.g.*, Euler's formula, counting rules). For each original question, we first use MCTS with the best underlying model, Claude-4-Sonnet, to reconstruct the erroneous reasoning trajectory corresponding to each distractor. We then create isomorphic variants by perturbing only the numerical values while preserving the underlying reasoning structure. For each variant, we recompute distractors by executing the same reconstructed erroneous reasoning paths, and combine them with the correct answer to form the new option set. Students then answer these variant questions in randomized order to minimize memory effects.

We use Behavioral Match@1 as the primary metric: a match is counted if the student again selects the distractor derived from their past QA records. The results show strong behavioral consistency, with Behavioral Match@1 reaching 0.68.

This provides additional evidence that the MCTS-reconstructed reasoning captures student-specific error tendencies in a behaviorally meaningful way.

D.8 Bad Case Analysis

Overall, our framework exhibits stable performance across most students and question types, and is generally able to recover meaningful misconception patterns from historical QA records. Nevertheless, we also observe occasional failure cases that reflect the inherent difficulty of distinguishing stable reasoning tendencies from incidental errors in student behavior.

In particular, some errors arise when a student's past records contain isolated, non-systematic mistakes—such as accidental mis-clicks, arithmetic slips, or atypical one-off reasoning attempts—that do not reflect the student's true misconception but are still reconstructed as plausible trajectories during MCTS. We note that this phenomenon is not specific to our approach: because such incidental mistakes are embedded in the raw QA data, they can influence any method that learns or infers patterns from historical records, including the baseline systems. In all such cases, the risk stems from the data itself rather than from the modeling mechanism (Wang et al., 2025b; Yan et al., 2025).

These cases tend to occur when the available history is sparse and heterogeneous, so that a small number of atypical mistakes exert disproportionate influence on the recovered prototype. We explored whether expanding the retrieval scope to include more past records could mitigate this issue, but found that retrieving additional tasks may also introduce irrelevant or weakly related reasoning traces, enlarging the prompt and increasing semantic noise, which can in turn degrade prediction quality. This suggests that the challenge lies not in data quantity alone, but in the fundamental difficulty of separating incidental noise from cognitively meaningful evidence—a challenge faced broadly across personalization-based modeling approaches.

It is worth noting that such edge cases are relatively rare and do not affect the overall stability of our framework, whose primary focus is to capture consistent, recurring misconception tendencies rather than isolated accidental behaviors. Fully resolving this phenomenon would require more advanced mechanisms for noise-aware misconception modeling, such as temporal weighting, confidence filtering, or selective prototype updating.

Finally, we note that similar challenges have also been reported in other domains—such as recommender systems—where spurious user behaviors or accidental interactions can bias profile construction (Wang et al., 2021; Lin et al., 2025; Li et al., 2025). In those settings, handling such noise often motivates separate lines of research (*e.g.*, debiasing, counterfactual modeling, or causal filtering (Tang et al., 2024, 2026; Li et al., 2024)) rather than being solved within a single pipeline. We view our observation in educational data as part of this broader challenge, and consider it a promising direction for future independent investigation.