SMRC: Aligning Large Language Models with Student Reasoning for Mathematical Error Correction

Anonymous EMNLP submission

Abstract

Large language models (LLMs) often make reasoning errors when solving mathematical problems, and how to automatically detect and correct these errors has become an important research direction. However, existing approaches *mainly focus on self-correction within the model*, which falls short of the "teacher-style" correction required in educational settings, *i.e.*, systematically guiding and revising a student's problem-solving process. To address this gap, we propose SMRC (*Student Mathematical Reasoning Correction*), a novel method that aligns LLMs with student reasoning. Specifically, SMRC formulates student reasoning as a multi-step sequential decision problem and introduces Monte Carlo Tree Search (MCTS) to explore optimal correction paths. To reduce the cost of the annotating process-level rewards, we leverage breadth-first search (BFS) guided by LLMs and final-answer evaluation to generate reward signals, which are then distributed across intermediate reasoning steps via a back-propagation mechanism, enabling fine-grained process supervision. Additionally, we construct a benchmark for high school mathematics, MSEB (Multi-Solution Error Benchmark), consisting of 158 instances that include problem statements, student solutions, and correct reasoning steps. We further propose a dual evaluation protocol centered on **solution accuracy** and **correct-step retention**, offering a comprehensive measure of educational applicability. Experiments demonstrate that SMRC significantly outperforms existing methods on two public datasets (ProcessBench and MR-GSM8K) and our MSEB in terms of effectiveness and overall performance.

1 Introduction

In recent years, large language models (LLMs) have achieved remarkable progress in complex mathematical reasoning tasks, thanks to their powerful logical and computational abilities (Li et al., 2025). These advances have not only improved the performance of LLMs in scientific and engineering domains but have also accelerated their integration into educational applications. However, existing studies reveal that even state-of-the-art reasoning models remain prone to logical inconsistencies and computational errors (Boye and Moëll, 2025). To address these challenges, researchers have proposed self-correction approaches for LLMs. By incorporating iterative refinement, multi-candidate verification, and consistency checks, these methods move beyond traditional single-pass generation and establish a unified generate–verify–correct framework, enabling models to autonomously detect and rectify internal reasoning errors, thereby improving the reliability of mathematical inference.

Meanwhile, the applications of LLMs in education have become increasingly diverse (Liu et al., 2024; from BNU and TAL, 2025). Beyond correcting their own reasoning errors, LLMs are now being leveraged as "teacher-like" agents to identify and correct students' reasoning mistakes in problem-solving. However, these two types of correction differ fundamentally in both goals and mechanisms, as illustrated in Figure 1. Model self-correction focuses on enhancing logical consistency within the model's reasoning, whereas student-oriented correction emphasizes pedagogical coherence and cognitive support (Li et al., 2024; Narciss and Huth, 2004). In educational contexts, effective correction involves not only providing the correct final answer but, more importantly, maintaining continuity with the student's original reasoning path to foster deeper cognitive development. Recent benchmark datasets, such as ProcessBench (Zheng et al., 2024) and MR-GSM8K (Zeng et al., 2023b), represent initial steps in this direction. Nonetheless, these datasets rely on synthetically generated student responses, which fail to capture the nuanced cognitive patterns and systematic misconceptions of real learners. This limitation

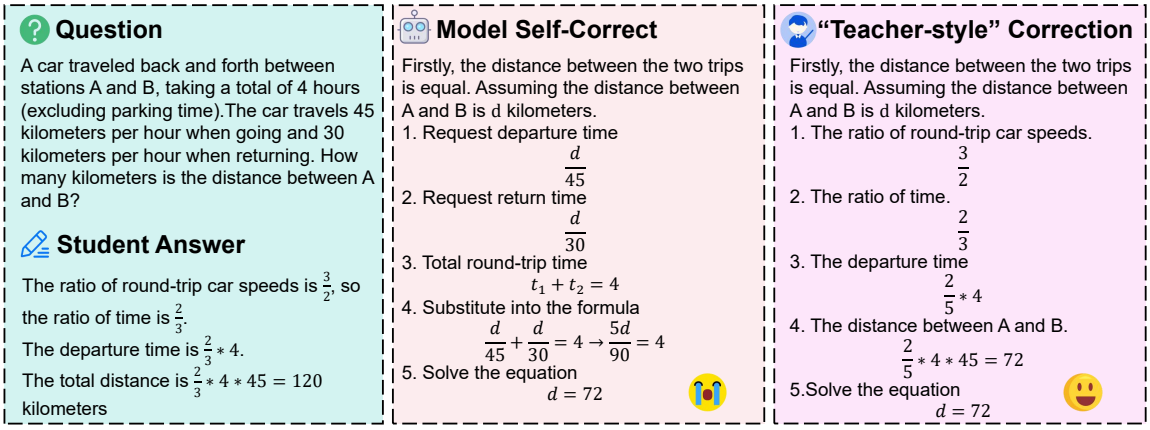

Figure 1: Example of two correction modes: given an input question and student answer (Left), model self-correction detects reasoning errors in LLMs (Middle), while "teacher-style" correction identifies errors in the student's reasoning process (Right).

becomes particularly evident in practical teaching scenarios—models trained on simulated data often struggle to generalize to authentic student reasoning. Therefore, there is an urgent need for a correction framework grounded in real student data that simultaneously ensures mathematical rigor and pedagogical alignment, paving the way toward more effective human–AI collaborative teaching.

To overcome these limitations in existing research, we propose SMRC (Student Mathematical Reasoning Correction), a novel framework designed to align LLMs with students' reasoning processes. Specifically, SMRC formulates student problem-solving as a multi-step sequential decision-making task and employs Monte Carlo Tree Search (MCTS) to explore optimal correction paths. To reduce the cost of the annotating process-level rewards, we leverage LLM-guided breadth-first search (BFS) in conjunction with final-answer evaluation to generate reward signals, which are then back-propagated across intermediate reasoning steps to enable fine-grained process supervision. Additionally, we construct a high school mathematics benchmark dataset, termed **MSEB (Multi-Solution Error Benchmark), comprising 158 authentic student solution records.** Unlike existing benchmarks that rely on model-simulated errors, MSEB preserves the complete reasoning trajectories of students, capturing their genuine cognitive patterns and common error types. Each problem in MSEB includes multiple valid solution strategies, enabling comprehensive evaluation of correction methods in terms of both maintaining mathematical correctness and preserving the student's original reasoning logic. Our contributions are summarized as follows:

- We introduce the novel research problem of Student Mathematical Reasoning Correction (SMRC), emphasizing that in educational contexts, correction should not only ensure mathematical accuracy but also preserve students' original reasoning paths to support cognitive development.
- SMRC employs Monte Carlo Tree Search to explore optimal correction paths and combines LLM-guided breadth-first search with back-propagated process-level rewards to achieve fine-grained supervision, effectively balancing correction accuracy with fidelity to students' thoughts.
- We construct the MSEB dataset, comprising 158 authentic high-school student solution records. The dataset preserves complete reasoning paths and includes multiple valid solution methods, enabling comprehensive evaluation of correction approaches in terms of both mathematical correctness and preservation of student reasoning. Extensive experiments validate the effectiveness of our method.

2 Related Work

Self-correction for LLMs. Self-correction has emerged as a critical approach for improving reasoning accuracy in large language models. Recent datasets such as ProcessBench (Zheng et al., 2024) and MR-GSM8K (Zeng et al., 2023b) use open-source models to generate erroneous reasoning patterns that simulate student problem-solving.

Representative methods include Self-Check (Miao et al., 2024), which employs internal consistency verification, and Self-Refine (Madaan et al., 2023), which uses iterative refinement strategies. However, existing approaches optimize primarily for answer correctness while neglecting a key educational requirement: preserving students' original reasoning strategies. In real-world tutoring, effective correction must build upon students' initial approaches rather than replacing them entirely. This gap motivates our work—developing student-centered correction that maintains fidelity to original reasoning while achieving high accuracy.

Test-time scaling (TTS). As pretraining-phase scaling has plateaued, test-time scaling has emerged as a key research focus. By reallocating computational resources from training to inference, TTS improves performance without additional training. Representative approaches include: (1) Parallel Scaling: generating k candidates and selecting the best via reward models (Wang et al., 2023; Sun et al., 2024); (2) Tree Search: using reward-guided exploration to find high-quality reasoning paths (Zhang et al., 2023; Wang et al., 2024); (3) Long Chain-of-Thought: producing extended reasoning sequences with iterative verification (Xia et al., 2025; Yang et al., 2025); (4) Multi-round Refinement: iteratively improving outputs through self-correction. Our work adopts Monte Carlo Tree Search (MCTS) for mathematical error correction, using external reward models to compute Q-values that guide path selection and prevent inherent model biases. This design balances thought preservation with correction accuracy (Welleck et al., 2023; Xi et al., 2024).

Reward models (RMs). Reward models guide model outputs toward human preferences by converting them into optimizable scalar signals (Gao et al., 2022). They are categorized into two types (Wang et al., 2025): Outcome-supervised Reward Models (ORMs) evaluate final answers, while Process-supervised Reward Models (PRMs) assess intermediate reasoning steps (Zhong et al., 2025). These models enable quality optimization during training or inference (Zeng et al., 2023a). Unlike traditional reward models, our approach introduces a specialized reward model for student mathematical reasoning correction that identifies optimal nodes where students achieve maximum reasoning progress. To address the scarcity of process-supervised labels, we develop an algorithm that infers process-level supervision from outcome labels through reasoning tree construction and label propagation. This enables precise guidance within the SMRC framework, optimizing for both mathematical correctness and consistency with students' original reasoning approaches.

3 Preliminaries

In this section, we will briefly and separately introduce the problem definition of self-correction for LLMs and the Monte Carlo Tree Search method.

3.1 Self-Correction for LLMs

Self-correction for LLMs aims to enable models to autonomously identify and correct errors in their mathematical reasoning processes. In recent years, researchers have proposed various self-correction methods, such as iterative self-refinement strategies and internal consistency verification mechanisms (Madaan et al., 2023; Miao et al., 2024), driving significant research interest in this area. We formalize this task as:

Let Q denote a mathematical problem and $A = \{a_1, a_2, \ldots, a_n\}$ represent a reasoning sequence that may contain logical errors or miscalculations. The objective of LLM self-correction specifically aims to generate a corrected sequence A^* that satisfies:

$$A^* = \arg\max_{A'} \underbrace{P(A' \vdash Q_{\text{answer}} | Q, A)}_{\text{Correctness Probability}} \quad (1)$$

In this formulation, $P(\cdot)$ denotes the probability function, $\vdash$ denotes logical entailment, and Q_{answer} represents the correct answer to problem Q. Given reasoning sequence A, the objective is to find A^* most likely to correctly derive the solution.

3.2 Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS) (Świechowski et al., 2023) provides a principled framework for decision-making in complex search spaces through balanced exploration and exploitation. The algorithm iteratively constructs a search tree via four key phases: *selection*, *expansion*, *simulation*, and *backpropagation*.

During selection, the algorithm iteratively selects the child maximizing the UCT criterion:

$$s^* = argmax_{s \in node} \text{UCT}(s) \quad (2)$$

where the UCT value is defined as:

$$\text{UCT}(s) = \frac{Q(s)}{N(s)} + c\sqrt{\frac{\ln N(s_{\text{parent}})}{N(s)}} \quad (3)$$

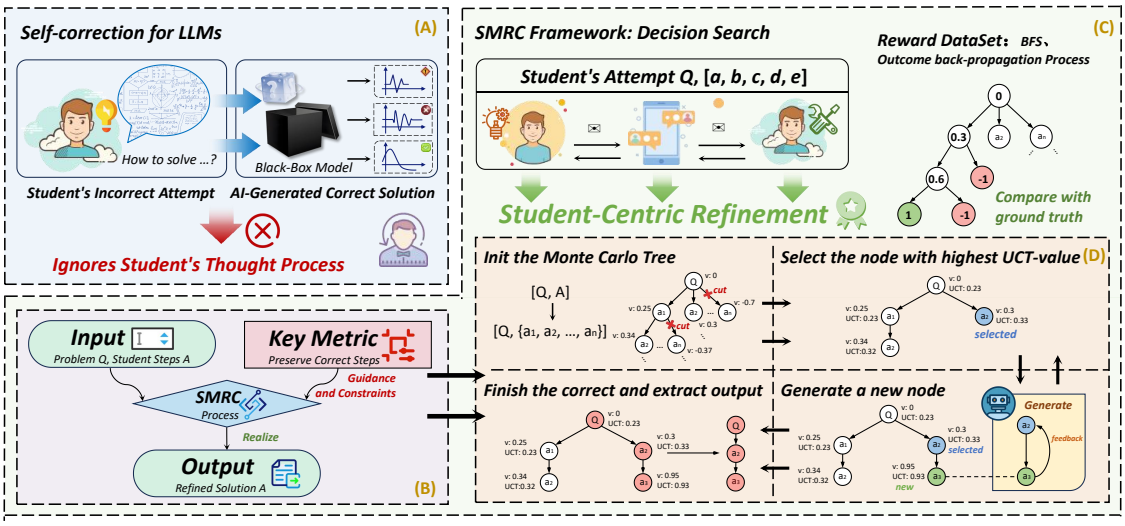

Figure 2: (A) Self-correction methods for LLMs often fail to address students' reasoning processes, relying instead on direct model-generated answers (B) SMRC inputs student questions and initial attempts, using Monte Carlo Tree Search to generate educational corrections; (C) The framework combines outcome and process rewards to train the reward model (D) This method starts from student attempts and provides final corrections through four reasoning steps.

Here, $Q(s)$ denotes the cumulative reward of node s, $N(s)$ its visit count, and c the exploration coefficient. The term $\frac{Q(s)}{N(s)}$ represents exploitation, favoring nodes with higher average rewards, while $c\sqrt{\frac{\ln N(s_{\text{parent}})}{N(s)}}$ represents exploration, encouraging the search of less frequently visited nodes.

In the expansion phase, new child nodes are added, followed by rollout-based simulation to obtain a reward and backpropagation to update node statistics. This process gradually concentrates search on the most promising regions.

4 Methodology

While existing self-correction frameworks in Section 3 emphasize answer correctness, they overlook the preservation of students' reasoning, which is crucial in educational settings. We propose Student Mathematical Reasoning Correction (SMRC), which formulates error correction as a sequential decision-making problem and applies Monte Carlo Tree Search (MCTS) with a specialized reward function to improve answer accuracy while maintaining alignment with students' reasoning trajectories.

4.1 Problem Formulation: Student-Aware Mathematical Reasoning Correction

Self-correction tasks for large language models have primarily focused on mathematical reasoning correctness. However, in educational contexts, correctness alone is insufficient: abrupt changes in reasoning may confuse students and hinder learning. We therefore define a new task, ***student-aware mathematical reasoning correction***, which requires corrected solutions to be both accurate and consistent with students' reasoning processes.

Although this task may appear similar to conventional self-correction, it differs fundamentally in its objective. The corrected reasoning sequence A^* must not only yield the correct answer to a problem Q, but also preserve valid steps from the student's original reasoning sequence A. This requires explicitly balancing solution correctness with fidelity to the student's cognitive trajectory.

Formally, the objective is defined as:

$$A^* = \arg\max_{A'} P\big(A' \vdash Q_{\text{answer}} \mid Q, A\big) + \lambda \sum_{i=1}^{n} \mathbb{I}\big(\text{preserve}(s_i, A')\big)\,, \tag{4}$$

where $\mathbb{I}(\cdot)$ is an indicator function that equals 1 if a correct reasoning step $s_i \in A_{\text{correct}} \subseteq A$ is preserved in A', and 0 otherwise.

4.2 SMRC Framework Overview

To correct mathematical errors while preserving students' original reasoning patterns, we propose the SMRC framework (Fig. 2). The framework consists of two core components: a reward model and

Monte Carlo Tree Search (MCTS). In the following, we describe each component in detail.

4.2.1 Reward Model in SMRC

In SMRC, the reward model outputs are treated as Q-values to guide node selection in MCTS. Since **fine-grained process supervision is extremely scarce**, we propose a **difference-based backtracking algorithm** that transforms sparse outcome-level labels into dense process-level supervision signals. The algorithm consists of two steps:

Reasoning Tree Construction: We construct reasoning trees using LLM-based BFS. The root node is initialized with value 0, while leaf nodes are assigned ± 1 according to answer correctness.

Differential Reward Allocation: For each leaf node, we backtrack along its path to the nearest ancestor with an assigned reward, compute the reward difference, and uniformly distribute it across the nodes in the corresponding path segment. The final reward of each node is obtained by cumulative aggregation, reflecting the overall quality of the reasoning path from the root to that node.

The reward model trained with these process-level supervision signals accurately evaluates the contribution of each reasoning step. **Its outputs are directly used as Q-value estimates in MCTS**, providing core guidance for node selection. A construction example is provided in the appendix H.

4.2.2 Monte Carlo Tree Search in SMRC

To systematically explore and optimize reasoning path in student problem-solving processes, SMRC employs the Monte Carlo Tree Search (MCTS) to locate errors and generate correction solutions. The specific process is as follows:

Tree Initialization: The student's solution is decomposed into n atomic reasoning steps. Starting from the root node (empty path), the tree is built layer by layer: the first layer contains single steps, and later layers extend paths by adding unused steps in their original order. Enumerating all ordered subsets yields a complete tree with 2^n nodes. An example is shown in Appendix G.

Node Selection: The reward model computes Q-values for each node to evaluate the quality of its reasoning path. Based on Q-values, the Upper Confidence Bound for Trees (UCT) criterion is used to balance exploration and exploitation, selecting the most promising student reasoning node as the basis for subsequent reasoning.

Reasoning Path Generation and Validation: Given the reasoning path of the selected node, the large language model generates subsequent steps to extend the student's solution. A reward model evaluates the newly generated nodes, and those that fail to yield positive improvement over the selected node are pruned, retaining only high-value branches to improve search efficiency.

Backpropagation and Output: The reward of a newly generated reasoning node is backpropagated to update the Q-values of its ancestor nodes. The algorithm terminates when a complete reasoning path exceeds a predefined reward threshold or the maximum number of iterations is reached, and returns the highest-reward path as the final solution.

SMRC turns error correction into a constrained reasoning task, selecting the best node via a reward model to preserve correct steps and improve correction accuracy. The complete algorithmic workflow is described in Algorithm 1.

Algorithm 1 Student Mathematical Reasoning Correction (SMRC)

Require: Math problem Q, Student's attempt A, Reward model R, Maximum attempts T, Reward threshold θ
Ensure: Corrected solution A^*

```
T ← InitializeTree(Q, A, R)
for t = 1 to T do
    s_select ← SelectNode(T)
    s_new ← GenerateValidatedStep(s_select)
    if GenerationFailed(s_new) then
        continue
    end if
    q_new ← R(s_new)
    if q_new ≤ s_select.Q then
        continue
    end if
    AddNode(s_select, s_new, q_new)
    UpdateAncestor(s_new, q_new)
    if q_new ≥ θ then
        return GetPathToRoot(s_new)
    end if
end for
s_best ← FindHighestScoringNode(T)
return GetPathToRoot(s_best)
```

4.3 MSEB Dataset Construction

To evaluate SMRC, we construct a student problem-solving dataset, **MSEB (Multi-Solution Error Benchmark)**. Unlike prior work relying on model-simulated student errors, MSEB is derived from

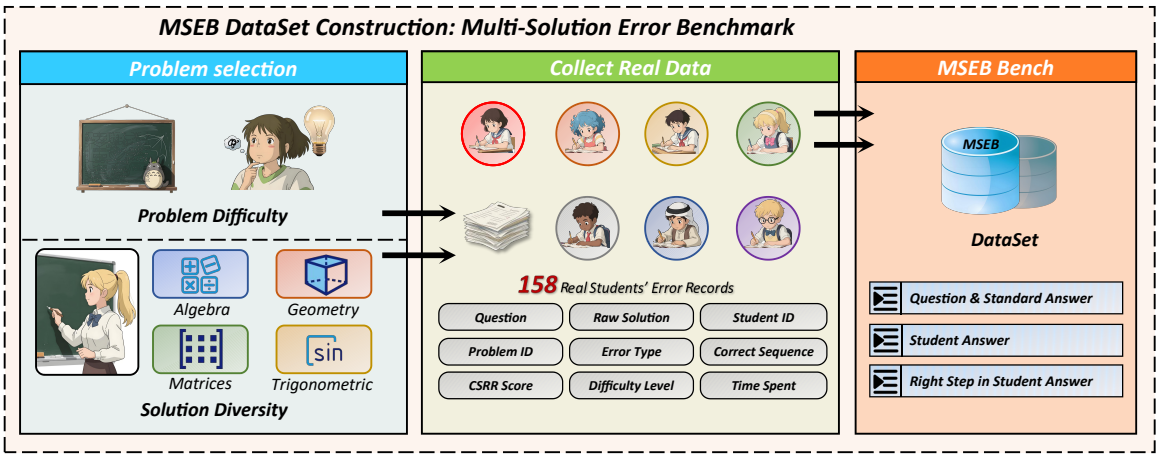

Figure 3: MSEB Dataset Construction Pipeline.

high school students' problem-solving processes, providing a reliable benchmark for fine-grained student error identification and correction.

Problem Selection: We select problems from a high school mathematics question bank covering algebra, geometry, matrices, and trigonometry, guided by two criteria: **(i) Problem Difficulty**, ensuring sufficient complexity to elicit diverse reasoning and common student errors; and **(ii) Solution Diversity**, selecting problems with multiple valid solution paths to assess whether correction methods preserve students' original correct reasoning.

Data Collection and Annotation: We collect complete written solutions from high school students solving the selected problems. After systematic organization and annotation, we construct a dataset of **158 real student error records**. Each record contains the original problem, standard answer(s), the student's full solution, and the correctly reasoned steps.

Example instances are provided in Appendix C.2. Additional dataset statistics and coverage analyses are reported in Appendix D. All data are anonymized to remove personally identifiable or sensitive information.

5 Experiments

In this section and appendix, we conduct experiments to evaluate the effectiveness of our proposed SMRC method, aiming to answer the following questions: (1) Can SMRC significantly improve the model's ability in student reasoning correction? (Section 5.2) (2) Can instruct models directly serve as reward models? (Section 5.3.1) (3) How effective is SMRC in different settings (or ablation study)? (Section 5.3.2 and Appendix K) (4) How universal is SMRC, i.e., can it be directly plugged into various base models to achieve consistent performance improvements? (Appendix L) (5) What is the computational cost of SMRC and the efficiency trade-off introduced by MCTS? (Appendix J)

5.1 Experimental Setup

Benchmarks. We evaluate SMRC framework on three datasets: ProcessBench (Zheng et al., 2024), MR-GSM8K (Zeng et al., 2023b), and MSEB dataset. ProcessBench is built upon GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), OlympiadBench (He et al., 2024), and Omni-MATH (Gao et al., 2025), while MR-GSM8K is based on GSM8K. Both public datasets use LLM-generated student reasoning with human-annotated error steps. For CSRR evaluation, we treat all steps before the first annotated error as correct, since ProcessBench and MR-GSM8K datasets only label the first erroneous step.

Baselines. We compare our proposed SMRC framework with the following baseline method: **Direct Error Correction (DEC)**: Directly prompt LLMs to correct student' incorrect answer. **Thought-Preserving Error Correction (TPEC)**: Based DEC and instruct LLMs to preserve student' original reasoning method during correction. **Self-Refine** (Madaan et al., 2023): Optimizing reasoning path through multiple rounds of self-improvement. **Self-Check** (Miao et al., 2024): Generates multiple reasoning paths and selects the output with the highest self-check score. For DEC and TPEC methods, we evaluate on multiple base models, including open-source models of different scales (Team, 2024; Dubey et al., 2024) and

Method	ProcessBench			MR-GSM8K			MSEB		
	ACC	CSRR	HM	ACC	CSRR	HM	ACC	CSRR	HM
Qwen2.5-7B-Instruct									
DEC	50.7 ± 0.4	67.7 ± 0.2	58.0 ± 0.2	60.5 ± 0.9	79.6 ± 0.2	68.7 ± 0.7	18.7 ± 1.1	56.4 ± 1.8	28.1 ± 1.0
TPEC	53.6 ± 0.4	92.9 ± 0.1	68.0 ± 0.3	63.1 ± 0.5	94.7 ± 0.2	75.7 ± 0.4	17.9 ± 2.6	84.7 ± 1.1	29.4 ± 3.3
Llama-3.1-8B-Instruct									
DEC	30.9 ± 3.7	66.0 ± 1.5	42.7 ± 3.2	37.4 ± 0.7	81.1 ± 0.5	51.1 ± 0.6	6.75 ± 1.1	38.3 ± 2.2	11.4 ± 1.4
TPEC	29.7 ± 2.8	80.8 ± 0.7	43.4 ± 2.1	37.4 ± 0.5	92.0 ± 0.5	53.2 ± 0.5	8.03 ± 2.1	65.2 ± 2.8	14.2 ± 3.3
Qwen2.5-14B-Instruct									
DEC	56.8 ± 0.4	93.6 ± 0.2	70.7 ± 0.3	76.1 ± 0.6	88.8 ± 0.2	82.0 ± 0.4	21.4 ± 1.6	81.7 ± 1.1	34.0 ± 2.0
TPEC	57.9 ± 0.2	$\mathbf{96.3 \pm 0.1}$	72.3 ± 0.1	80.7 ± 0.2	93.5 ± 0.2	86.7 ± 0.2	25.0 ± 0.6	88.2 ± 0.9	39.1 ± 0.8
MuduoLLM									
DEC	52.0 ± 0.1	30.3 ± 0.2	38.3 ± 0.1	21.4 ± 0.5	25.4 ± 0.3	23.2 ± 0.4	17.1 ± 0.9	28.3 ± 1.2	21.3 ± 1.0
TPEC	56.8 ± 0.7	84.0 ± 0.2	67.8 ± 0.6	71.9 ± 0.5	86.4 ± 0.3	78.5 ± 0.4	19.8 ± 2.4	57.0 ± 2.5	29.3 ± 2.7
Llama-3.1-70B-Instruct									
DEC	48.3 ± 0.1	29.2 ± 0.3	36.4 ± 0.2	52.6 ± 1.2	46.7 ± 0.6	49.5 ± 0.8	10.4 ± 0.6	27.6 ± 2.8	15.0 ± 1.0
TPEC	41.0 ± 0.4	78.5 ± 0.5	53.8 ± 0.3	72.5 ± 0.8	91.3 ± 0.3	80.9 ± 0.4	11.3 ± 0.9	82.3 ± 0.8	20.0 ± 1.4
Qwen2.5-72B-Instruct									
DEC	64.4 ± 0.5	82.1 ± 0.2	72.1 ± 0.3	78.7 ± 0.7	91.0 ± 0.3	84.4 ± 0.5	26.5 ± 0.5	69.4 ± 3.5	38.3 ± 0.9
TPEC	61.6 ± 0.9	92.5 ± 5.3	73.9 ± 2.4	82.2 ± 0.4	$\mathbf{96.4 \pm 0.1}$	88.7 ± 0.2	27.2 ± 1.6	82.1 ± 0.8	40.8 ± 1.7
Self-Check	62.5 ± 0.3	95.3 ± 0.6	$\mathbf{75.5 \pm 0.3}$	83.9 ± 0.3	95.2 ± 0.1	89.2 ± 0.2	27.2 ± 0.1	$\mathbf{91.2 \pm 1.6}$	41.9 ± 1.1
Self-Refine	60.3 ± 0.4	85.9 ± 0.9	70.8 ± 0.6	86.9 ± 1.5	90.4 ± 0.6	88.6 ± 0.5	35.2 ± 2.1	76.6 ± 4.2	48.1 ± 1.8
SMRC (ours)	$\mathbf{64.9 \pm 0.4}$	89.3 ± 0.4	75.2 ± 0.2	$\mathbf{91.4 \pm 0.5}$	94.5 ± 0.2	$\mathbf{92.9 \pm 0.2}$	$\mathbf{40.1 \pm 1.7}$	73.7 ± 2.1	$\mathbf{51.8 \pm 1.1}$

Table 1: Results of SMRC and other frontier LLMs on three benchmarks. Methods without model specification use Qwen2.5-72B-Instruct. **Bold** denotes the best, and underlined the second-best results.

education-specific models (from BNU and TAL, 2025). Self-Refine and Self-Check methods are evaluated using the Qwen2.5-72B-Instruct model.

Implementation details. We utilize Qwen2.5-14B-Instruct fine-tuned on our reward dataset to estimate Q-values, while Qwen2.5-72B-Instruct serves as the primary generator for node expansion in the Monte Carlo tree. The remaining hyperparameter settings are detailed in Appendix M.

Metrics. To evaluate the efficacy of our approach, we adopt three key performance metrics:

(1) **Reasoning Accuracy (ACC):**

$$ACC = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \mathbb{I}[\text{IsValid}(A_i^*, Q_i, RA_i)] \quad (5)$$

(2) **Correct Step Retention Rate (CSRR):**

$$CSRR = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \frac{|A_i^* \cap A_i^{\text{correct}}|}{|A_i^{\text{correct}}|} \quad (6)$$

The two metrics measure error correction accuracy and correct step retention, respectively, where $|D|$ denotes the dataset size, A_i^* is the corrected solution, Q_i is the i-th question, RA_i is the reference answer, $A_i^{correct}$ is the set of correct steps in the original solution, and $IsValid(\cdot)$ is the validation function. We adopt the harmonic mean **(HM)** as a unified metric to jointly evaluate, defined as:

$$HM = \frac{2 \cdot CSRR \cdot ACC}{CSRR + ACC}$$

Following prior work (Ho et al., 2025; Zhang et al., 2025), we use Qwen2.5-72B-Instruct (Team, 2024) with structured prompts for evaluation.

5.2 Main Results

Table 1 presents a comprehensive performance comparison between our SMRC framework and baseline methods across three datasets. The experimental results reveal several key findings:

SMRC shows versatility and robustness. It performs consistently across multiple error correction datasets. Although not the top method on ProcessBench, which emphasizes complex reasoning, SMRC achieves leading results on MR-GSM8K and MSEB. These results suggest strong adaptability to diverse correction settings and good generalization.

SMRC addresses the limitations of generation-based filtering through structured search. Guided by an external reward model, its MCTS framework explores correction paths more thoroughly and avoids local optima compared with methods such as Self-Check and Self-Refine. The advantage is especially evident on datasets like **MSEB**, which contain multiple valid solution paths.

We observe that in some cases SMRC achieves lower CSRR than other methods. This is because our framework optimizes answer correctness, while step retention is only encouraged. When these objectives conflict, SMRC prioritizes a correct final

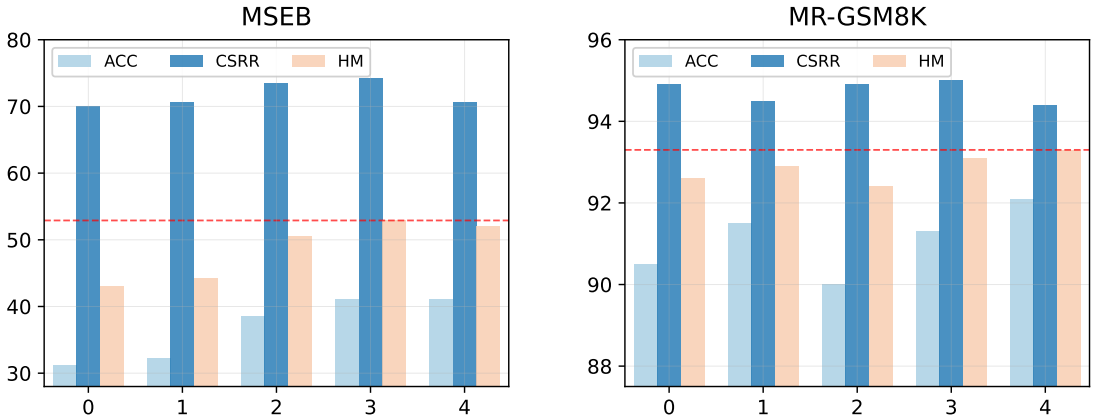

Figure 4: Impact of feedback iterations on performance in SMRC. The red line denotes the maximum HM value.

		ProcessBench			MR-GSM8K			MSEB			
Reward Model	Size	ACC	CSRR	HM	ACC	CSRR	HM	ACC	CSRR	HM	Avg. HM
Qwen2.5-Instruct	7B	50.2	76.8	60.7	48.0	88.3	62.2	23.4	62.1	34.1	52.3
Qwen2.5-Instruct	14B	58.2	84.4	68.9	52.1	91.0	66.3	26.5	**75.1**	39.2	58.1
Qwen2.5-Instruct	72B	52.6	75.3	62.0	52.1	91.1	66.3	22.7	67.7	34.1	54.1
Llama-3.1-Instruct	8B	44.3	79.5	56.9	61.9	91.7	73.9	16.4	61.1	25.9	52.2
Llama-3.1-Instruct	70B	46.5	80.3	58.9	36.5	88.9	51.7	20.2	72.7	31.6	47.4
SMRC (ours)	14B	**64.9**	**89.3**	**75.2**	**91.4**	**94.5**	**92.9**	**40.1**	73.7	**51.8**	**73.3**

Table 2: Results of using open-source foundation models directly as reward models through prompt engineering. Among these, our proposed SMRC method, which is obtained by fine-tuning the Qwen2.5-14B model, demonstrates performance advantages. **Bold** indicates the best performance, and underline indicates the second-best performance.

answer, which may lead to lower CSRR.

5.3 Ablation Study

5.3.1 Base LM vs. Fine-tuned Reward Model

To validate the necessity of fine-tuning the reward model, we compared several instruction-tuned base models (Qwen2.5-Instruct 7B/14B/72B and Llama-3.1-Instruct 8B/70B) used directly as reward models against our fine-tuned 14B SMRC model.

Results demonstrate that our fine-tuned 14B model achieves superior performance. While base models struggle to capture fine-grained process-level distinctions—showing insufficient accuracy in identifying reasoning step correctness and considerable performance volatility—our task-specific fine-tuned model provides substantially more reliable assessments of reasoning step quality, with stable and balanced performance across all metrics.

5.3.2 Impact of the Feedback Mechanism

We conduct an ablation study on the feedback mechanism of SMRC across iterations on two benchmark datasets (Fig. 4). Performance improves with iterations and converges after 3–4 rounds. This suggests that moderate iterations balance computational cost and performance, while the increase in harmonic mean demonstrates the feedback mechanism's effectiveness in improving correction quality and preserving correct reasoning.

6 Conclusion

This work introduces SMRC framework to address the misalignment between large language models and students' reasoning pathways in mathematical error correction. We formulate ***student-aware mathematical reasoning correction*** as a dual-objective optimization problem that jointly optimizes answer correctness and preservation of students' cognitive processes, and solve it using a MCTS algorithm with guided exploration, multi-step verification, and dynamic pruning.

We further construct the **MSEB** dataset of real-world student solutions with annotated errors and introduce **Correct Step Retention Rate (CSRR)** to evaluate educational alignment. Experiments show that SMRC achieves strong and robust performance across multiple benchmarks.

Limitations

Despite the strong performance of SMRC, several limitations remain.

First, the current uniform reward backtracking strategy may not fully capture the heterogeneous contributions of individual reasoning steps in complex reasoning processes. Designing more fine-grained credit assignment mechanisms is an interesting direction for future work.

Second, SMRC relies on MCTS-based search, which introduces additional computational overhead as the number of reasoning steps increases, even with pruning and reward guidance. Improving the efficiency of search-based correction frameworks remains an important direction for future research.

Third, while the MSEB dataset is constructed from authentic high school student solutions, its scale is still relatively limited due to the difficulty of collecting and annotating real student reasoning traces. Expanding the dataset with broader coverage of mathematical topics and student reasoning patterns is an important direction for future work.

Finally, some evaluations rely on LLM-based automatic judges, which may introduce model-specific biases. Future work could incorporate additional human evaluation to further strengthen the reliability of the evaluation.

References

Johan Boye and Birger Moëll. 2025. Large language models and mathematical reasoning failures. *ArXiv*, abs/2502.11574.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Abhimanyu Dubey, Abhinav Jauhri, and Abhinav Pandey et al. 2024. The llama 3 herd of models. *ArXiv*.

MuduoLLM Contributors from BNU and TAL. 2025. Muduollm: A high-performance llm for intelligent education solutions.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. 2025. Omni-MATH: A universal olympiad level mathematic benchmark for large language models. In *Proc. of ICLR*.

Leo Gao, John Schulman, and Jacob Hilton. 2022. Scaling laws for reward model overoptimization. In *Proc. of ICML*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proc. of ACL*, pages 3828–3850.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Xanh Ho, Jiahao Huang, Florian Boudin, and Akiko Aizawa. 2025. Llm-as-a-judge: Reassessing the performance of llms in extractive qa. *ArXiv*.

Xinnuo Li, Yunxiang Zhang, and Lu Wang. 2024. Improving language model self-correction capability with meta-feedback.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. 2025. Limr: Less is more for rl scaling. *Preprint*, arXiv:2502.11886.

Jiayu Liu, Zhenya Huang, Tong Xiao, Jing Sha, Jinze Wu, Qi Liu, Shijin Wang, and Enhong Chen. 2024. SocraticLM: Exploring socratic personalized teaching with large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Proc. of NeurIPS*.

Ning Miao, Yee Whye Teh, and Tom Rainforth. 2024. Selfcheck: Using LLMs to zero-shot check their own step-by-step reasoning. In *Proc. of ICLR*.

Susanne Narciss and K. Huth. 2004. How to design informative tutoring feedback for multi-media learning.

Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. 2024. Easy-to-hard generalization: Scalable alignment beyond human supervision. In *Proc. of NeurIPS*.

Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. 2023. Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, pages 2497–2562.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Ante Wang, Linfeng Song, Ye Tian, Baolin Peng, Dian Yu, Haitao Mi, Jinsong Su, and Dong Yu. 2024. Litesearch: Efficacious tree search for llm. *ArXiv*.

Pengyuan Wang, Tian-Shuo Liu, Chenyang Wang, Yidi Wang, Shu Yan, Cheng-Xing Jia, Xu-Hui Liu, Xin-Wei Chen, Jia-Cheng Xu, Ziniu Li, and Yang Yu. 2025. A survey on large language models for mathematical reasoning. *ArXiv*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *Proc. of ICLR*.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2023. Generating sequences by learning to self-correct. In *Proc. of ICLR*.

Zhiheng Xi, Dingwen Yang, Jixuan Huang, Jiafu Tang, Guanyu Li, Yiwen Ding, Wei He, Boyang Hong, Shihan Do, Wenyu Zhan, Xiao Wang, Rui Zheng, Tao Ji, Xiaowei Shi, Yitao Zhai, Rongxiang Weng, Jingang Wang, Xun-Ye Cai, Tao Gui, and 5 others. 2024. Enhancing llm reasoning via critique models with test-time and training-time supervision. *ArXiv*.

Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie Wang, and Wenjie Li. 2025. Tokenskip: Controllable chain-of-thought compression in llms. *ArXiv*.

Wenkai Yang, Shuming Ma, Yankai Lin, and Furu Wei. 2025. Towards thinking-optimal scaling of test-time compute for llm reasoning. *ArXiv*.

Dun Zeng, Yong Dai, Pengyu Cheng, Tianhao Hu, Wanshun Chen, Nan Du, and Zenglin Xu. 2023a. On diversified preferences of large language model alignment. In *Proc. of EMNLP*.

Zhongshen Zeng, Pengguang Chen, Haiyun Jiang, and Jiaya Jia. 2023b. Challenge llms to reason about reasoning: A benchmark to unveil cognitive depth in llms. *Preprint*, arXiv:2312.17080.

Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B. Tenenbaum, and Chuang Gan. 2023. Planning with large language models for code generation. In *Proc. of ICLR*.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. In *Proc. of ACL Findings*, pages 10495–10516.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2024. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*.

Jialun Zhong, Wei Shen, Yanzeng Li, Songyang Gao, Hua Lu, Yicheng Chen, Yang Zhang, Wei Zhou, Jinjie Gu, and Lei Zou. 2025. A comprehensive survey of reward models: Taxonomy, applications, challenges, and future. *ArXiv*.

Appendix

A The Use of Large Language Models (LLMs)

The authors leveraged LLMs solely as a writing aid to improve textual expression. The conceptual development, research execution, and final validation were entirely conducted and overseen by the authors, who take full responsibility for the work's integrity.

B Ethical Considerations

The data used in this study was collected in an educational setting with approval from the participating school. Students participated voluntarily and were informed that their responses would be used for research purposes. No personally identifiable information was collected or released. The collected data consists solely of task-oriented mathematical problem-solving responses and poses minimal risk to participants.

C The MSEB Dataset Example and Description

This appendix illustrates the structure, design principles, and utility of the MSEB (Multi-Solution Error Benchmark) dataset through a concrete example, demonstrating its role in evaluating the task of *Student Aware Mathematical Reasoning Correction*.

C.1 Data Format

Each instance in the MSEB dataset is stored as a JSON object containing the following core fields:

- `"question"`: The statement of the mathematical problem.
- `"answer"`: The standard correct solution and complete solving process for the problem.
- `"student_answer"`: A solution attempt collected from a real student, containing common errors.
- `"correct_step"`: A sequence of reasoning steps from the student's attempt that are judged to be correct. This field is used to

calculate the **Correct Step Retention Rate (CSRR)**.

C.2 Data Instance

The following example showcases a complete data instance. It exemplifies a typical scenario where a student demonstrates correct initial reasoning but makes a critical error in a later transformation.

MSEB Dataset Instance

Question: Given real numbers a, b, c such that $a+b+c=0$ and $a^2+b^2+c^2=1$, the maximum value of a is _____.

Standard Answer:

1. Substituting $c=-(a+b)$ into $a^2+b^2+c^2=1$ yields $2a^2+2b^2+2ab=1$.
2. That is, $2a^2-1=-2b(a+b)\leq 2(\frac{-b+a+b}{2})^2=\frac{a^2}{2}$.
3. Therefore, $a^2\leq\frac{2}{3}$, i.e., $-\frac{\sqrt{6}}{3}\leq a\leq\frac{\sqrt{6}}{3}$.
4. Hence, the maximum value of a is $\frac{\sqrt{6}}{3}$.

Student Answer:

1. Express c as $-(a+b)$ and substitute into $a^2+b^2+c^2=1$, obtaining $a^2+b^2+(a+b)^2=1$.
2. Expand and simplify: $a^2+b^2+a^2+2ab+b^2=1$; $2a^2+2b^2+2ab=1$; $a^2+b^2+ab=\frac{1}{2}$.
3. ... (subsequent erroneous reasoning) ...
4. In conclusion, the maximum value of a is $\boxed{\frac{\sqrt{6}}{6}}$.

Correct Steps:

1. Express c as $-(a+b)$ and substitute into $a^2+b^2+c^2=1$, obtaining $a^2+b^2+(a+b)^2=1$.
2. Expand and simplify the equation: $a^2+b^2+a^2+2ab+b^2=1$; $2a^2+2b^2+2ab=1$; $a^2+b^2+ab=\frac{1}{2}$.
3. $a^2+b^2+ab=(a+\frac{b}{2})^2+\frac{3}{4}b^2=\frac{1}{2}$.

D Dataset Coverage Analysis of MSEB

To better understand the coverage of the MSEB dataset, we analyze the distributions of question formats, knowledge areas, and error types. These statistics help illustrate the diversity of mathematical problems and reasoning errors represented in the dataset.

D.1 Question Format Distribution

Question Format	Count	Percentage
Solution-based Questions	73	46.20%
Proof Questions	41	25.95%
Fill-in-the-Blank Questions	35	22.15%
Multiple-Choice Questions	9	5.70%

Table 3: Distribution of question formats in MSEB.

The dataset covers multiple assessment formats commonly used in high-school mathematics exams, with solution-based and proof questions forming the majority of instances.

D.2 Knowledge Area Distribution

Knowledge Area	Count	Percentage
Inequalities	38	24.05%
Analytic Geometry	26	16.46%
Trigonometric Functions	16	10.13%
Sequences and Series	13	8.23%
Probability and Statistics	8	5.06%
Plane Geometry	7	4.43%
Sequences and Inequalities	7	4.43%
Functions and Derivatives	14	8.86%
Other Algebra and Function Problems	29	18.35%

Table 4: Distribution of knowledge areas in MSEB.

MSEB spans a diverse set of mathematical topics commonly taught in high-school curricula, including algebra, geometry, trigonometry, and probability.

D.3 Error Type Distribution

Error Type	Count	Percentage
Computational Errors	76	48.10%
Logical Reasoning Errors	67	42.41%
Formula Misuse	5	3.16%
Improper Bounding Method	7	4.43%
Other Error Types	3	1.90%

Table 5: Distribution of error types in MSEB.

The error distribution shows that computational errors and logical reasoning mistakes account for

the majority of cases. This pattern reflects common error types observed in real student work rather than synthetic error distributions generated by language models.

Discussion Overall, MSEB covers a wide range of question formats, mathematical topics, and reasoning error types. Although the dataset is relatively small, it captures realistic student reasoning patterns.

In addition, SMRC demonstrates consistent improvements on larger benchmarks such as PROCESSBENCH and MR-GSM8K, suggesting that the performance gains are not tied to a particular dataset distribution.

E SMRC framework node generates prompt

SMRC framework node generates prompt

You are a high-precision mathematical problem-solving and error-correction engine. Your task is to receive a [Question] and a [Student's Solution Process], and generate a "minimally corrected" correct solution.
[Core Principles]
1. **Identify the Method**: First, analyze and understand the problem-solving strategy chosen by the student (e.g., substitution method or elimination by addition/subtraction for systems of equations? Formula method or completing the square for quadratic equations?).
2. **Preserve Correct Parts**: Retain, exactly as written,**all correct steps** in the student's solution process that occur**before the first error**.
3. **Correct Within the Chosen Method**: Starting from the first error, you**must continue and complete the solution using the strategy already chosen by the student**.**It is forbidden** to switch to a different solution method.
4. **Clean Output**: Directly output the complete, correct solution process without any explanations, titles, or extraneous text.
Please strictly imitate the following examples when handling the final [Formal Task].
Example 1: Algebraic Operation Error
[Question]&[Student's Solution Process]
Solve the equation: 5(x + 1) - 2 = 23
Solution:
5(x + 1) - 2 = 23
5x + 1 - 2 = 23
5x - 1 = 23
5x = 24
x = 4.8
[Your Output]
Solution:
5(x + 1) - 2 = 23
5x + 5 - 2 = 23
5x + 3 = 23
5x = 20
x = 4
Example 2: Geometric Concept Confusion
[Question]&[Student's Solution Process]
What is the area of a square inscribed in a circle with a radius of 5cm?
Solution:
The radius of the circle r = 5cm.
The diameter of the circle d = 10cm.
The side length of the square a = d = 10cm.
The area of the square $S = a^2 = 10^2 = 100$ cm².
[Your Output]
Solution:
The radius of the circle r = 5cm.
The diameter of the circle d = 10cm.
The diagonal length of this inscribed square is equal to the diameter of the circle, which is 10cm.
Let the side length of the square be a, then $a^2 + a^2 = 10^2$.
$2a^2 = 100$
$a^2 = 50$
The area of the square $S = a^2 = 50$ cm².
Example 3: Must Adhere to the Student's Chosen Strategy
[Question]&[Student's Solution Process]
Solve the system of equations:
(1) x + y = 3
(2) 2x - y = 6
(The student chose the "substitution method") Solution:
From (1), we get x = 3 + y
Substituting x into (2) gives:
2(3 + y) - y = 6
6 + 2y - y = 6
y = 0
Substituting y=0 into (1) gives x = 3.
So the solution is x=3, y=0.
[Your Output]
*(The model must also use the "substitution

method" for correction, not switch to the simpler "elimination by addition/subtraction method")*
Solution:
From (1), we get x = 3 - y
Substituting x into (2) gives:
2(3 - y) - y = 6
6 - 2y - y = 6
6 - 3y = 6
-3y = 0
y = 0
Substituting y=0 into x = 3 - y gives x = 3.
So the solution to the system is x=3, y=0.

F Other prompts

F.1 Multi-turn Feedback Prompt for the SMRC Framework

Multi-turn Feedback Prompt for the SMRC Framework

It seems there might be some issues with your answer. Please review it and provide a new response.

F.2 Student Solution Decomposition Prompt

Student Solution Decomposition Prompt

Please help me break down the steps in the student's answer and provide them in the following format:
Steps 1:...
Steps 2:...
Steps 3:...
The student's answer is as follows:
{student_answer}

G The case of init for Monte Carlo Tree in SMRC

The figure 5 illustrates the answer combination tree for question Q with value scores (v) computed by the Reward Model. Red dashed edges denote pruned paths where values decrease relative to parent nodes ($a_1 \rightarrow a_3$ and $Q \rightarrow a_3$). The path $a_1 \rightarrow a_2 \rightarrow a_3$ achieves the highest score (v:0.6), representing the optimal sequence.

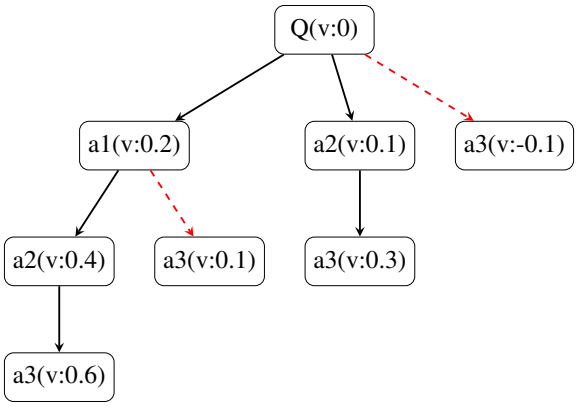

Figure 5: Initialization tree for Question Q with Student Answers A = $\{a_1, a_2, a_3\}$

H Example of Reward Data Construction

As illustrated in Figure 6, we employ a breadth-first search (BFS) strategy to expand multiple reasoning paths for problem Q, constructing a complete search tree structure. For value assessment, leaf nodes corresponding to reasoning paths that yield correct answers are assigned a value of +1, while leaf nodes associated with reasoning paths that produce incorrect answers are assigned a value of -1.

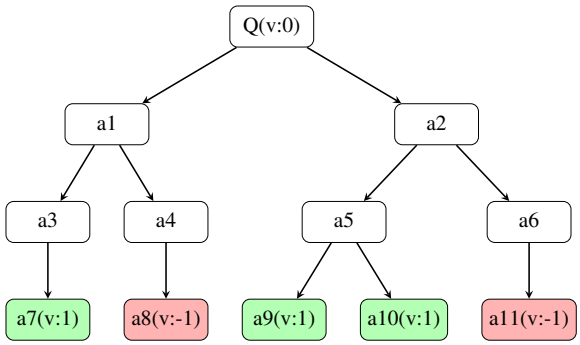

Figure 6: Initial reasoning tree structure with leaf node value assignments. Green nodes indicate correct answers (+1), red nodes indicate incorrect answers (-1).

The computational workflow for step-level rewards in the reasoning tree is as follows:

1. Identify the longest correct reasoning sequence (using node a_7 as an example), backtracking from the leaf node to the root node until the first value-annotated node is found. Nodes along the path equally distribute the total reward value R of the reasoning path, with each node receiving reward:

$$r_i = \frac{R}{n} \tag{7}$$

where n is the number of nodes in the path.

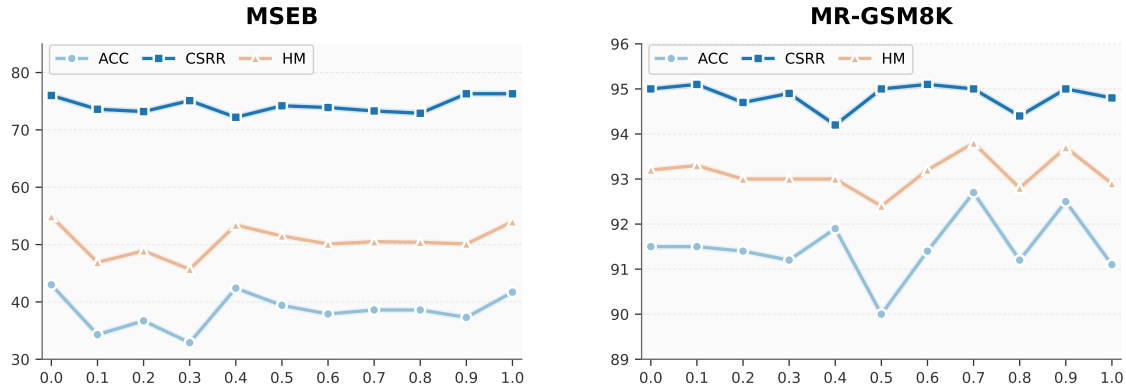

Figure 7: Effect of MCTS exploration parameter c on error correction performance.

2. Node values are computed using cumulative calculation:

$$V(n_i) = V(\text{parent}(n_i)) + r_i \tag{8}$$

3. Consequently, for a path with $R = 1$ and $n = 3$:

$$V(a_1) = 0.33 \tag{9}$$
$$V(a_3) = 0.33 + 0.33 = 0.66 \tag{10}$$
$$V(a_7) = 0.66 + 0.33 = 1.0 \tag{11}$$

demonstrating the incremental contribution of the reasoning pathway.

4. Then select the next node, a_9 and a_{10}, to apply the same approach.

5. Finally, for incorrect reasoning paths: Select a leaf node from an erroneous path (e.g., a_8). Backtrack upward to locate the first value-assigned node a_1 with $V(a_1) = 0.33$. The path nodes share the negative contribution:

$$\begin{aligned} \text{shared contribution} &= -1 - V(a_1) \\ &= -1 - 0.33 \\ &= -1.33 \end{aligned} \tag{12}$$

Each node (a_4 and a_8) receives $r = \frac{-1.33}{2} = -0.665$. Therefore:

$$\begin{aligned} V(a_4) &= V(a_1) + r = 0.33 + (-0.665) \\ &= -0.335 \end{aligned} \tag{13}$$

$$\begin{aligned} V(a_8) &= V(a_4) + r = -0.335 + (-0.665) \\ &= -1.0 \end{aligned} \tag{14}$$

The final constructed result is illustrated in Figure 8.

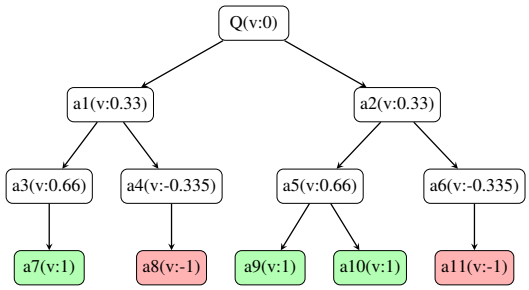

Figure 8: Reasoning tree after step-level reward propagation. Node values reflect cumulative rewards computed through backtracking from leaf nodes to the root.

I Performance Comparison of Search Algorithms in the SMRC Framework: A Case Study of BFS, DFS, and MCTS

Table 7 compares the performance of three search algorithms. Although BFS and DFS achieve higher CSRR (BFS: 98.6%/97.6%; DFS: 96.9%/92.0%), they lack effective node quality assessment mechanisms and tend to retain all explored steps, resulting in lower accuracy (BFS: 15.0%/8.86%; DFS: 83.2%/27.2%).

MCTS, guided by the Reward Model for pruning, achieves slightly lower CSRR (94.5%/73.7%) but significantly higher accuracy (91.4% and 40.1%), yielding the best harmonic mean scores (92.9/51.8). This demonstrates that value-driven pruning effectively balances step retention and answer accuracy.

Method	Harmonic Mean	Avg. Total Token	Avg. Time / Case	Time Overhead
DEC	38.3	1.2k	44.88 s	1.00×
TPEC	40.8	1.4k	52.53 s	1.17×
Self-Check	41.9	68.6k	959.28 s	21.37×
Self-Refine	48.1	7.4k	84.39 s	1.88×
SMRC	**51.8**	51.2k	118.37 s	2.64×

Table 6: Performance-cost trade-off on the MSEB dataset. SMRC achieves the best harmonic mean while incurring moderate computational overhead compared with lightweight baselines.

	MR-GSM8K			MSEB		
Method	ACC	CSRR	HM	ACC	CSRR	HM
BFS	15.0	**98.6**	26.1	8.86	**97.6**	16.2
DFS	83.2	96.9	89.5	27.2	92.0	42.0
MCTS	**91.4**	94.5	**92.9**	**40.1**	73.7	**51.8**

Table 7: Performance comparison of search algorithms in the SMRC framework on MR-GSM8K and MSEB datasets.

	MR-GSM8K			MSEB		
Method	ACC	CSRR	HM	ACC	CSRR	HM
DEC	37.1	80.7	50.8	6.31	38.6	10.8
TPEC	6.30	38.6	10.8	10.8	63.6	18.5
Self-Check (Miao et al., 2024)	48.5	91.2	63.3	8.01	60.3	14.4
Self-Refine (Madaan et al., 2023)	21.6	79.8	34.1	6.96	**77.1**	12.7
SMRC(ours)	**67.8**	**92.8**	**78.4**	**11.6**	53.2	**19.1**

Table 8: Comparison of SMRC against baseline methods on MR-GSM8K and MSEB using Llama-3.1-8B-Instruct as the generator. Best results are in **bold** and second-best are underlined.

J Efficiency Analysis

Table 6 shows the performance–cost trade-off on the MSEB dataset. SMRC achieves the best harmonic mean with about 2–3× higher runtime than lightweight baselines like DEC. Although it uses more tokens than DEC and TPEC, it is still far more efficient than Self-Check, which relies on large-scale re-sampling. Moreover, MCTS runs with a fixed rollout budget, keeping inference cost explicitly bounded.

K Sensitivity Analysis of MCTS Exploration Parameter

As shown in Fig. 7, our framework demonstrates *acceptable but limited* robustness to the exploration parameter c ($c \in [0.0, 1.0]$). On MSEB, metric variation remain manageabl: ACC varies by $\sim$7%, CSRR by $\sim$6%, and HM by $\sim$8%. MR-GSM8K shows better stability with $<$2% variation across all metrics.

Despite measurable sensitivity, the framework maintains usable performance throughout the parameter range. Consistently high CSRR values confirm reliable preservation of correct reasoning steps. These results indicate that while parameter tuning yields marginal improvements, the framework remains practically applicable even with suboptimal c values, reducing dependence on hyperparameter optimization.

L Generality of SMRC Across Base Models

As shown in Table 8, a key strength of our SMRC framework lies in its model-agnostic nature and consistent effectiveness across different base models. To verify that the performance improvements are attributable to our algorithmic design rather than model-specific properties, we evaluated our approach on Llama-3.1-8B, a model that possesses significantly fewer parameters than the Qwen2.5-72B model used in our main experiments. Despite this substantial reduction in model capacity, SMRC maintains superior performance over all baseline methods on both benchmarks. This result underscores the efficacy of its structured Monte Carlo Tree Search and external reward guidance, enabling it to achieve the highest harmonic mean scores.

M Implementation Details

In our experimental setup for the SMRC framework, we employ the hyperparameter configuration: the exploration parameter c for Monte Carlo Tree Search is set to 0.4, with a maximum of 30 exploration attempts and a reward threshold of 0.95 for termination. The self-correction mechanism is constrained to 4 rounds. For model specifications, we utilize Qwen2.5-14B-Instruct fine-tuned on our reward dataset to estimate Q-values, while Qwen2.5-72B-Instruct serves as the generator for node expansion in the Monte Carlo tree.