

✦ STAR: Strategy-Aware Routing for Mathematical Reasoning

Anonymous ACL submission

Abstract

Existing methods usually leverage a fixed strategy, such as natural language reasoning, code-augmented reasoning, tool-integrated reasoning, or ensemble-based reasoning, to guide Large Language Models (LLMs) to perform mathematical reasoning. However, a single strategy cannot adequately handle diverse problems and thus neglects the trade-off between effectiveness and efficiency. To address these issues, we propose STAR (SStrategy-Aware Routing for mathematical reasoning), a novel framework that decouples mathematical reasoning into two stages: strategy evaluation and targeted execution. Specifically, we first construct a multi-strategy preference dataset, which we call MathStrat, capturing correctness, process quality, and computational efficiency for each problem–strategy pair. We then train a lightweight strategy adapter on the dataset to obtain a confidence distribution over the four reasoning strategies. Based on this distribution, an adaptive routing mechanism is leveraged to select the most appropriate reasoning strategy dynamically. This mechanism directs LLMs to use single-strategy execution for high-confidence predictions, dual-strategy verification for competitive scenarios, and comprehensive multi-strategy exploration for uncertain cases, respectively. Extensive experiments across five mathematical reasoning benchmarks demonstrate that PRISM consistently outperforms individual and ensemble reasoning strategies, achieving improvements ranging from 0.9% to 7.6% across different LLMs. In addition, the adaptive routing mechanism shows particularly strong benefits for mathematical reasoning tasks across various LLMs with different sizes. Our code is released at <https://anonymous.4open.science/r/STAR-2C61>.

1 Introduction

Large Language Models (LLMs), such as ChatGPT and Qwen, have achieved significant advancements across diverse natural language processing tasks

(Ma et al., 2025a,b). Notably, they have demonstrated strong performance in mathematical reasoning—a long-standing and challenging domain that requires precise logical inference, symbolic manipulation, and multi-step problem-solving (Gou et al., 2024b). Existing approaches to improving mathematical reasoning in LLMs can be broadly divided into three categories: (1) designing effective prompting methods for frozen LLMs (Trivedi et al., 2025), (2) developing strategies to enhance the capability of frozen LLMs (Didolkar et al., 2024), and (3) post-training LLMs on domain-specific data (Xia et al., 2025). Among these, the method of enhancing frozen LLMs through improving reasoning strategies such as chain-of-thought refinement (Wei et al., 2022b) and tool invocation (Xie et al., 2025) has garnered considerable attention due to its ease of deployment. Although these methods have achieved significant success, they still face two challenges.

Challenge 1: One strategy does not fit all. Existing methods primarily rely on isolated reasoning strategies, including Natural Language Reasoning (NLR) (Wang et al., 2024), Code-Augmented Reasoning (CAR) (Ye et al., 2024), Tool-Integrated Reasoning (TIR) (Li et al., 2024), and Ensemble-Based Reasoning (EBR) (Ranaldi et al., 2024), to enhance the mathematical reasoning of frozen LLMs. As illustrated in Figure 1, we evaluate the performance of these individual strategies in boosting the reasoning ability of Qwen-2.5-math across four question types sampled from MATH (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), and SVAMP (Patel et al., 2021). Our analysis reveals that no single strategy consistently outperforms others across diverse problem categories. This performance variance underscores a key limitation: rigidly adhering to a fixed reasoning paradigm fails to fully unlock the latent capabilities of frozen LLMs and hampers the adaptability to various problem types.

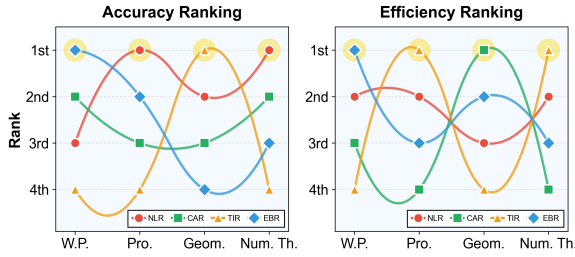


Figure 1: Accuracy and efficiency ranking of four reasoning strategies on four problem slices. Each slice (e.g., Number Theory, Geometry) comprises over 100 instances drawn from the MATH, GSM8K, and SVAMP benchmarks.

Challenge 2: Trade-offs between efficiency and effectiveness are overlooked. Current approaches (Xin et al., 2025; Zhang and Xiong, 2025) often disregard the computational cost, latency, and resource efficiency of reasoning strategies. As shown in Figure 1, we report the normalized inference efficiency of different strategies for enhancing the reasoning ability of Qwen2.5-7B across four question types. Notably, no single strategy consistently achieves the best efficiency. This observation suggests that a fixed reasoning paradigm leads to suboptimal deployments, where substantial computational expense does not yield commensurate improvements in accuracy.

To address these challenges, we introduce STAR (SStrategy-Aware Routing for mathematical reasoning), a framework that decouples mathematical reasoning into two core stages: instance-specific strategy evaluation and targeted execution via dynamic routing. Specifically, we propose a data construction approach based on multi-strategy performance profiling, which systematically evaluates diverse reasoning strategies on each problem instance to generate fine-grained suitability distributions rather than single-strategy labels. To achieve this, we execute four distinct reasoning strategies (i.e., NLR, CAR, TIR, and EBR) on problems from standard benchmarks like MATH and GSM8k. Each resulting solution trajectory is then evaluated using a multi-faceted scoring function that considers correctness, process quality, and efficiency to generate per-strategy suitability scores. These raw scores are then transformed into a soft target distribution via a temperature-scaled softmax function. We then train a lightweight Strategy Adapter by minimizing the Kullback-Leibler (KL) divergence between its output and this target distribution, which encourages the model to cap-

ture the relative suitability of strategies for each instance. At inference time, the output from the predictor drives our problem-aware strategy routing that reconciles the efficiency-effectiveness trade-off through confidence-based execution paths: high-confidence predictions trigger streamlined single-path execution; competitive scores between strategies invoke dual-path verification for robustness; and diffuse uncertainty defaults to comprehensive multi-path exploration. Through this confidence-guided orchestration, STAR achieves both strategic flexibility and computational efficiency, selecting the most suitable reasoning approach for each problem while scaling computational effort according to prediction certainty.

To verify the effectiveness and superiority, we evaluate STAR across five standard mathematical reasoning benchmarks, including MATH500, GSM8K, AQUA-RAT (Ling et al., 2017), SVAMP, and ASDiv (Miao et al., 2020). Our experiments show that STAR consistently delivers significant performance gains. Notably, on the challenging MATH benchmark, our method achieves an accuracy of 53.2%, surpassing the best-performing single-strategy baseline (TIR) by 3.1% absolute. Furthermore, it outperforms the standard ensemble-based approach (EBR) by a margin of 2.5% absolute, demonstrating the superiority of pre-execution routing over post-hoc aggregation.

Our main contributions are as follows:

1. We introduce STAR, a novel framework that decouples the mathematical reasoning process into two core stages: instance-specific strategy evaluation and targeted execution via dynamic routing. This is operationalized through a lightweight meta-predictor trained on MathStrat, our constructed dataset of $\sim 13,000$ instances that provides rich, multi-faceted supervision signals that capture the relative suitability of various reasoning strategies.
2. We design a dynamic, verifier-free routing policy for inference. This policy interprets the predictor’s output to adaptively select among single-, dual-, or multi-path execution modes, providing a principled mechanism to balance performance with computational cost.
3. We conduct extensive experiments across five standard mathematical benchmarks. The results demonstrate that STAR consistently and

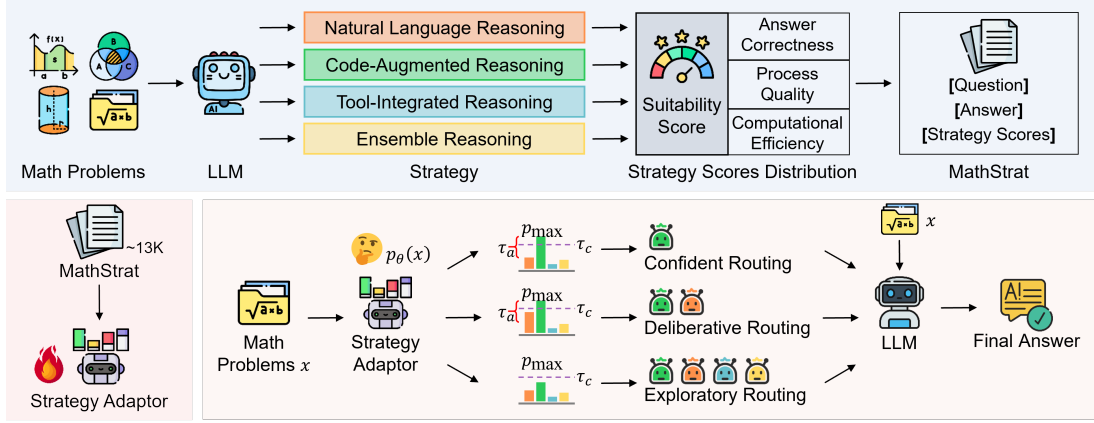


Figure 2: Overview of the STAR framework. The framework consists of two stages: Offline training (top and bottom-left), where mathematical problems are solved under multiple reasoning strategies and build a dataset called MathStrat for training the Strategy Adapter (SA); Online inference (bottom-right), where SA guides adaptive routing to produce the final answer.

significantly outperforms all single-strategy baselines and a standard ensemble method, validating the superiority of our problem-aware routing approach.

2 Methodology

The STAR framework shown in Figure 2 is designed in two stages that decouple strategy awareness from execution. The first stage involves an offline training of a Strategy Adapter (SA). It learns to map a given problem instance to a suitability distribution over a set of reasoning strategies. The second stage is the online inference, where the prediction of SA guides an adaptive routing policy to select an execution pathway dynamically.

2.1 Strategy Adapter

As outlined in Section 1, existing reasoning strategies can be broadly categorized into four paradigms: NLR, CAR, TIR, and EBR. To achieve dynamic strategy selection, we propose the strategy adaptation pipeline: (1) collection of strategy preference data and (2) training of the strategy suitability assessment model.

Collection of Strategy Preference Data. To generate effective training signals for strategy selection, we evaluate each strategy across multiple aspects. We construct supervision signals using three complementary dimensions that capture the essential trade-offs: (1) answer correctness, which determines whether the strategy produces the correct solution to the given mathematical problem; (2) process quality, which evaluates whether the strategy follows mathematically valid reasoning

steps and avoids logical errors or redundant operations; and (3) computational efficiency, which measures whether the strategy achieves results within reasonable time and resource consumption. We quantify computational efficiency using two metrics: the actual wall-clock inference time and the number of generated tokens, both normalized via within-instance min-max scaling. To implement this evaluation framework, we execute all four reasoning strategies (NLR, CAR, TIR, and EBR) on each problem instance x using identical base models and decoding configurations. This yields three measurements per strategy s : (i) binary correctness $f_c(s, x) \in \{0, 1\}$ indicating successful problem solving; (ii) process quality $f_q(s, x) \in [0, 1]$ from an automated evaluator that penalizes invalid steps and redundant reasoning (Xia et al., 2025); and (iii) efficiency score $f_e(s, x) \in [0, 1]$ computed from raw timing and output length metrics. Specifically, we define the efficiency score $f_e(s, x)$ as:

$$f_e(s, x) = 1 - \frac{1}{2}(\hat{t}(s, x) + \hat{\ell}(s, x)), \quad (1)$$

where the normalized timing and length components $\hat{t}(s, x)$ and $\hat{\ell}(s, x)$ are obtained through min-max scaling within each problem instance. We compute an overall score $\phi(s, x)$ for strategy s on problem instance x by linearly combining three interpretable signals:

$$\phi(s, x) = \mathbf{w}^\top \mathbf{f}(s, x), \quad (2)$$

where $\mathbf{f}(s, x) = [f_c(s, x), f_q(s, x), f_e(s, x)]^\top$, and $\mathbf{w} = [w_c, w_q, w_e]^\top$ is the corresponding weight vector. The sensitivity analysis of weight

configurations is detailed in Appendix A.3. We then form a soft supervision target distribution $\mathbf{y}(x)$ by applying a temperature-scaled softmax function over the four scores within the same instance, where $y(s, x)$ represents the target probability for strategy s on instance x :

$$y(s, x) = \frac{\exp(\phi(s, x)/\tau)}{\sum_{s' \in \mathcal{S}} \exp(\phi(s', x)/\tau)}, \tau = 0.5. \quad (3)$$

Strategy Adapter Training. Given the training set $\{x_i\}_{i=1}^N$ in MathStrat, we train the SA f_θ to output logits $z_\theta(x_i) \in \mathbb{R}^{|\mathcal{S}|}$ and predict a probability distribution $p_\theta(x_i) = \text{softmax}(z_\theta(x_i))$ over strategies for each problem instance. Our training objective is designed to match the full target distribution while explicitly stabilizing the ranking of the top strategy. The primary objective minimizes the Kullback–Leibler (KL) divergence between the target distribution $\mathbf{y}(x_i)$ and the predicted distribution $p_\theta(x_i)$:

$$\mathcal{L}_{\text{dist}}(\theta) = \frac{1}{N} \sum_{i=1}^N \text{KL}(\mathbf{y}(x_i) \parallel \mathbf{p}_\theta(x_i)). \quad (4)$$

To reinforce learning of the top-ranked strategy, we add an auxiliary cross-entropy loss. Let $s_i^* = \arg \max_{s \in \mathcal{S}} S(s, x_i)$ be the best strategy for instance x_i . The auxiliary loss is:

$$\mathcal{L}_{\text{ord}}(\theta) = -\frac{1}{N} \sum_{i=1}^N \log p_{\theta, s_i^*}(x_i). \quad (5)$$

This auxiliary loss explicitly enforces top-1 selection, which is essential for our routing mechanism, and it also prevents the predictor from producing overly soft distributions when trained with the KL term alone. The final loss combines these objectives:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{dist}}(\theta) + \lambda \mathcal{L}_{\text{ord}}(\theta), \quad (6)$$

where λ is a hyperparameter that balances the two objectives. This combined distributional and ranking-aware objective encourages the model to capture the relative suitability of strategy families, which guides the inference-time routing policy. We implement the SA as a lightweight language model (e.g., 1.5B parameters) trained on MathStrat of approximately 13,000 problem instances with multi-strategy performance evaluations. Upon completion of training, the adapter demonstrates effective suitability assessment capabilities, with a representative example of its prediction behavior provided in Appendix A.6.

2.2 Adaptive Routing Policy at Inference

The SA produces a probability distribution $p_\theta(x)$ over strategies for each problem instance x . Simply selecting the highest-probability strategy, however, would result in uniform single-strategy execution regardless of prediction confidence. This approach neither leverages high confidence for computational efficiency nor exploits uncertainty for enhanced robustness through multi-strategy verification.

Our adaptive routing policy interprets the predictor output through a confidence-based framework that dynamically selects among three execution modes: *Confident*, *Deliberative*, and *Exploratory* routing. The mode selection depends on two calibrated thresholds: a confidence threshold τ_c and an ambiguity margin τ_a , which are optimized through grid search on a validation set (see Appendix A.4 for details), applied to the top two predicted probabilities p_{max} and $p_{2\text{nd}}$. *Confident Routing* ($p_{\text{max}} \geq \tau_c$ and $(p_{\text{max}} - p_{2\text{nd}}) \geq \tau_a$) executes only the single best-ranked strategy when the predictor exhibits high confidence with a clear preference. *Deliberative Routing* ($p_{\text{max}} \geq \tau_c$ and $(p_{\text{max}} - p_{2\text{nd}}) < \tau_a$) executes the top two strategies when confidence is high but rankings are close, selecting the answer with agreement or, in case of disagreement, the answer from the higher-confidence strategy. *Exploratory Routing* ($p_{\text{max}} < \tau_c$) executes all available strategies when predictor confidence is insufficient, again using majority voting for final answer selection. For multi-strategy modes, answers from different strategies first undergo format standardization before voting. When votes are tied, the answer from the strategy with the highest predicted probability is selected. This mechanism provides principled computational resource allocation without requiring external verification components.

3 Experiment

3.1 Setup

Datasets and Evaluation We use as diverse mathematical datasets as possible for experiments. In addition to the widely used MATH (Hendrycks et al., 2021) and GSM8K (Cobbe et al., 2021) dataset, we also adopt AQUA-RAT (Ling et al., 2017), SVAMP (Patel et al., 2021) and ASDiv (Miao et al., 2020). These datasets cover multiple fields of mathematics, such as elementary arithmetic problems, mathematical algebra, inferential

Table 1: Performance comparison of different mathematical reasoning strategies across three base language models and five datasets. CoT refers to Chain-of-Thought reasoning, PAL to Program-Aided Language models, ToRA to Tool-integrated Reasoning Agent, and Hybrid to ensemble-based approaches. STAR represents our proposed adaptive routing framework. Pass@k denotes the percentage of problems for which at least one correct solution appears in the top-k generated outputs. Blue highlighting indicates the best performance for each model-dataset combination.

Model	Approach	MATH500		GSM8K		AQUA-RAT		SVAMP		ASDiv		Average	
		Pass@1	Pass@5	Pass@1	Pass@5	Pass@1	Pass@5	Pass@1	Pass@5	Pass@1	Pass@5	Pass@1	Pass@5
Qwen2.5-Math-7B	CoT	21.2	50.8	78.1	93.5	37.0	57.5	85.7	94.6	82.8	92.5	61.0	77.8
	PAL	30.4	55.4	84.8	95.3	18.1	44.1	13.5	45.2	86.3	93.7	46.6	66.7
	ToRA	41.4	62.0	69.4	94.5	47.2	64.6	75.8	96.2	75.7	93.6	61.9	82.2
	Hybrid	37.2	53.2	24.0	68.2	15.4	44.5	21.8	63.9	15.9	50.1	22.9	56.0
	STAR (ours)	46.2	64.4	86.7	96.0	42.1	64.8	91.8	96.9	86.5	93.6	70.7	83.1
Deepseek-math-7b-v1	CoT	43.0	57.2	87.6	93.3	33.5	56.7	83.7	92.8	85.6	91.2	68.6	79.2
	PAL	38.0	53.2	83.9	91.8	47.6	55.1	84.8	90.2	84.2	89.7	67.7	76.0
	ToRA	32.2	49.2	78.3	93.0	38.6	58.7	76.5	92.4	79.7	91.5	61.1	77.0
	Hybrid	12.6	30.8	60.0	90.1	26.3	50.0	71.9	93.2	68.2	90.2	47.8	70.9
	STAR (ours)	52.2	61.6	87.8	93.6	45.3	62.2	88.0	94.3	88.6	92.3	72.4	79.9
Llama-3-8B	CoT	13.6	29.8	45.6	76.8	17.7	36.6	64.6	88.6	22.6	60.1	32.8	58.4
	PAL	10.4	22.9	54.3	78.4	16.5	35.4	72.4	89.3	13.5	31.7	33.4	51.5
	ToRA	11.0	25.0	43.9	75.1	14.6	33.1	65.6	89.2	21.9	60.0	31.4	56.5
	Hybrid	11.8	26.2	44.6	88.9	10.9	37.7	22.0	62.7	25.1	62.7	22.9	55.6
	STAR (ours)	15.2	36.2	53.0	78.5	14.6	33.1	66.1	89.6	63.7	83.1	42.5	64.1

counting, and probability number theory. They also span a wide range of difficulty levels, including simple elementary school math problems, intermediate-level questions, and even Olympiad-style competition problems. For evaluation metrics, we report Pass@k accuracy, where a problem is considered solved if the correct answer appears in the top-k generated solutions. Specifically, Pass@1 reflects single-shot correctness, while Pass@5 captures the probability of producing at least one correct solution among five independent generations.

Base Models and Reasoning Baselines Our experiments use three base language models: Qwen2.5-Math-7B (Yang et al., 2024), Deepseek-math-7b-v1 (Lu et al., 2024a), and Llama-3-8B. This selection covers both math-specialized and general-purpose models at comparable scales (7B–8B) to evaluate generalizability across varying model capabilities. We compare against four representative reasoning approaches as baselines. Chain-of-Thought (CoT; Wei et al., 2022a) prompts LLMs to generate step-by-step rationales before producing a final answer. Program-Aided Language Models (PAL; Gao et al., 2023a) introduce a neuro-symbolic approach that offloads the reasoning and calculation logic to an external tool. Tool-Augmented Reasoning Agent (ToRA; Zhang et al., 2023) can interleave natural language reasoning steps with calls to different tools, such as a cal-

culator, a symbolic solver, or retrieval APIs. Hybrid (Yue et al., 2023) approaches aim to combine the strengths of different reasoning paradigms to achieve superior performance and robustness.

3.2 Main Results

To evaluate the overall effectiveness and generalizability of STAR, we conduct comprehensive experiments across three base models and five mathematical reasoning benchmarks. Table 1 shows performance across three base models and five mathematical reasoning benchmarks. STAR achieves average improvements of 0.9% on Qwen2.5-Math-7B, 2.9% on Deepseek-math-7b-v1, and 7.6% on Llama-3-8B over the best single strategies. The inverse relationship between relative improvement and base model capability suggests that strategic routing provides greater value when addressing model limitations. The results confirm our central observation that no single strategy dominates across all benchmarks—while ToRA excels on MATH500, PAL leads on GSM8K for Qwen (95.3%), and performance varies dramatically across datasets. STAR effectively handles this heterogeneity through adaptive strategy selection. The method substantially outperforms the Hybrid baseline, particularly on complex reasoning tasks like AQUA-RAT, demonstrating that pre-execution routing is more effective than post-hoc strategy aggregation. Individual strategies exhibit high vari-

ance (PAL ranges from 13.5% to 95.3%), while STAR maintains consistent performance across all test conditions.

3.3 Ablation Study On Routing Components

To understand the contribution of each routing component, we conduct progressive ablation experiments across GSM8K, MATH500, and Hungarian Math datasets. As detailed in Table 2, adding confident routing shows mixed results—76.0% on GSM8K (below the 78.1% CoT baseline) but improvements on MATH500 (28.6% vs 21.2%) and Hungarian Math (50.0% vs 40.6%). Incorporating deliberative routing provides continued gains on MATH500 (32.2%) with variable performance elsewhere. The complete STAR system achieves substantial improvements across all datasets (86.7%, 46.2%, and 53.1%, respectively), with dramatic jumps from the previous configuration demonstrating that the full adaptive routing policy is essential for optimal performance. These results validate that individual routing modes provide limited benefits, while the intelligent coordination of all components through problem-aware strategy selection delivers significant performance gains.

Table 2: Ablation study of adaptive routing components on three benchmarks using Qwen2.5-Math-7B.

Setting	GSM8K		MATH500		Hun. Math	
	p@1	p@5	p@1	p@5	p@1	p@5
Vanilla CoT	78.1	93.5	21.2	50.8	40.6	56.3
Conf.	76.0	95.0	28.6	56.1	50.0	62.5
Conf. + Delib.	75.8	95.6	32.2	57.3	43.8	50.0
STAR	86.7	96.0	46.2	64.4	53.1	71.9

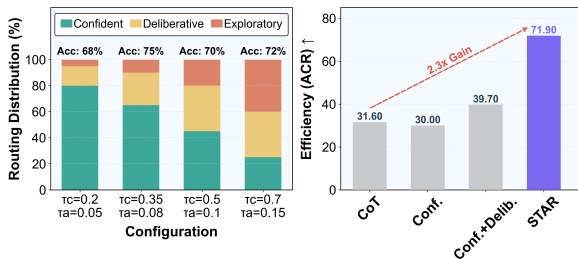


Figure 3: Routing configuration and efficiency analysis. **Left:** Routing mode distribution across different threshold configurations. The stacked bars show the percentage of problems routed to each execution mode (Confident, Deliberative, Exploratory) for varying τ_c and τ_a values, with accuracy annotated above each bar. **Right:** Efficiency comparison using ACR.

3.4 Routing Configuration and Efficiency Analysis

We analyze threshold sensitivity and efficiency progression across routing configurations to understand the interplay between routing behavior and computational efficiency.

Routing Behavior under Different Configurations.

To investigate how threshold parameters affect routing decisions, we vary the confidence threshold τ_c and ambiguity margin τ_a and examine the resulting routing mode distributions. Figure 3 (left) illustrates how the confidence threshold τ_c and ambiguity margin τ_a jointly influence the distribution of routing modes. As τ_c increases from 0.2 to 0.7, the proportion of Confident routing rises substantially from 25% to 80%, while Exploratory routing decreases from 40% to 5%. Notably, accuracy does not monotonically improve with higher confident routing ratios—the configuration $\tau_c = 0.5, \tau_a = 0.1$ achieves the best accuracy (75%) with a balanced distribution. Overly aggressive confident routing ($\tau_c = 0.7$) leads to performance degradation (68%), as the system loses the robustness benefits of multi-strategy verification on uncertain predictions.

Efficiency Analysis. To jointly evaluate accuracy and computational cost, we introduce the Accuracy-to-Cost Ratio (ACR), which measures reasoning performance normalized by log-scaled token overhead:

$$\text{ACR} = \frac{\text{Acc}}{\log_{10}(1+C)}, \quad C = \frac{\eta}{\eta_{\text{CoT}}}, \quad (7)$$

where η denotes the average output token count and η_{CoT} is the baseline token count from standard chain-of-thought. The log-scaled denominator dampens cost penalties for configurations requiring longer outputs. As shown in Figure 3 (right), the baseline CoT achieves an ACR of 31.6. Adding Confident routing alone yields minimal improvement (30.0), as single-strategy selection without deliberative fallback lacks robustness. Incorporating Deliberative routing increases ACR to 39.7, demonstrating the value of dual-strategy verification. The complete STAR system achieves an ACR of 71.9, representing a $2.3\times$ improvement over the baseline, validating that adaptive routing improves both accuracy and computational efficiency.

3.5 Scalability Analysis

To evaluate the scalability of our approach, we conducted experiments across Qwen2.5 models

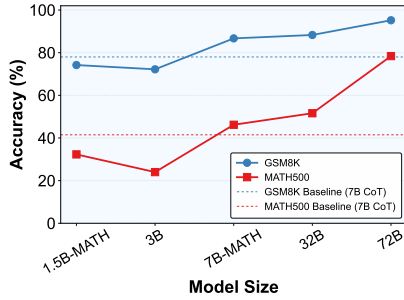


Figure 4: Scalability analysis of STAR across Qwen2.5 models of varying sizes on GSM8K and MATH500 benchmarks. Dotted lines indicate the baseline performance of Qwen2.5-7B with standard CoT prompting.

ranging from 1.5B to 72B parameters on GSM8K and MATH500 benchmarks, using Qwen2.5 7B with chain-of-thought prompting as our baseline. As illustrated in Figure 4, STAR demonstrates consistent improvements over the baseline across all model scales, achieving accuracy from 74.2% to 95.2% on GSM8K and 32.3% to 78.4% on MATH500. The framework exhibits distinct scaling patterns across benchmarks: steady improvements on GSM8K with notable gains from 7B to 32B parameters, and more dramatic scaling effects on MATH500 where performance nearly doubles from smallest to largest models. These scaling results validate that adaptive strategy selection provides robust benefits across different model capacities. The sustained improvements across parameter scales demonstrate that the framework generalizes effectively and does not depend on specific model characteristics to achieve performance gains. Importantly, since STAR operates as a training-free approach that works purely through inference-time strategy selection, it can be readily applied to any pre-trained model without requiring additional fine-tuning or domain-specific training, making it broadly applicable across different model families and computational budgets.

3.6 Strategy Adapter Behavior Analysis

We analyze the prediction behavior patterns of our SA across different mathematical reasoning datasets to validate its learned strategy selection characteristics. This analysis examines both the confidence levels in predictions and the competitive landscape among strategies. Figure 5 presents the distributions of prediction confidence (p_{\max}) and strategy competition gaps ($p_{\max} - p_{2\text{nd}}$) across four mathematical reasoning datasets. The results reveal several important patterns that validate our frame-

work design. First, the SA exhibits prediction confidence patterns that correlate with dataset complexity. On MATH500, which contains competition-level mathematical problems, the predictor shows notably conservative behavior with prediction confidence (p_{\max}) concentrated in the 0.0 to 0.2 range. This low confidence reflects both the inherent difficulty of these problems and the SA’s learned caution when dealing with competition-level mathematics, where strategy effectiveness is less predictable.

In contrast, datasets containing more elementary mathematical problems show progressively higher prediction confidence. GSM8K demonstrates moderate confidence levels with p_{\max} distributed across $0.2 \sim 0.6$, while ASDiv and SVAMP exhibit relatively higher confidence with peaks around $0.3 \sim 0.4$. This graduated confidence pattern indicates that the SA has successfully learned to associate problem complexity with prediction uncertainty, demonstrating sophisticated meta-reasoning about strategy applicability. The adaptive confidence calibration also suggests that the SA effectively captures the inherent variability in strategy effectiveness across different mathematical domains, with higher uncertainty appropriately assigned to problems where multiple strategies might yield similar performance. Furthermore, the distribution shapes themselves provide insight into the SA’s decision-making process: sharp peaks indicate clear strategy preferences for certain problem types, while flatter distributions suggest scenarios where multiple strategies remain viable options. The strategy competition analysis reveals consistently small gaps between top strategies across all datasets. This competitive landscape validates our adaptive routing design rationale: the narrow margins between strategy preferences require nuanced confidence-based decision making rather than simple winner-take-all selection. Additionally, we provide a detailed analysis of strategy performance patterns and inter-strategy correlations across datasets in Appendix A.5.

3.7 Problem Difficulty and Subject Analysis

To understand how problem characteristics influence routing effectiveness, we conduct a fine-grained analysis on MATH500 across two dimensions: difficulty levels (1-5, as annotated in the original dataset) and mathematical subjects. Figure 6 presents the performance breakdown and routing behavior patterns.

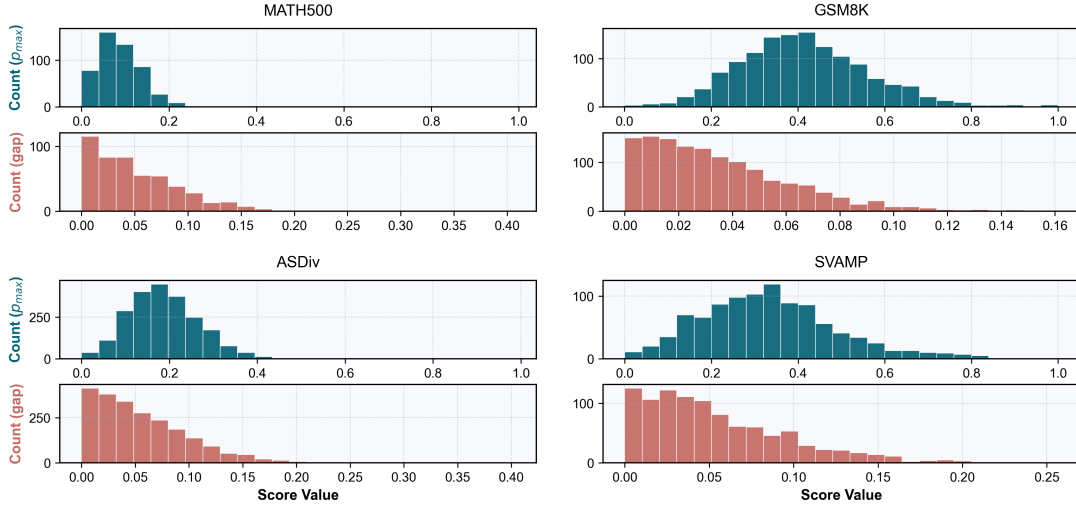


Figure 5: Strategy Adapter behavior across mathematical reasoning datasets. Top row shows prediction confidence (p_{\max}) distributions; bottom row shows strategy competition gaps ($p_{\max} - p_{2\text{nd}}$). The SA exhibits dataset-appropriate confidence levels: conservative predictions on competition problems (MATH500) and higher confidence on elementary problems (ASDiv, SVAMP), while maintaining competitive strategy landscapes across datasets.

Difficulty-Stratified Performance. Figure 6(a) shows that both Pass@1 and Pass@5 decrease monotonically as difficulty increases, with Pass@1 dropping from 68% (Level 1) to 15% (Level 5) and Pass@5 from 92% to 43%. The shaded gap between them widens at moderate difficulty (Levels 2-4), indicating that sampling diversity provides greater benefit for intermediate problems. Figure 6(b) reveals distinct accuracy patterns across routing modes: Exploratory routing maintains the highest accuracy across all levels, while Confident routing shows steeper degradation at higher difficulty. This validates our adaptive design, as the system appropriately shifts toward multi-strategy exploration when problems become challenging.

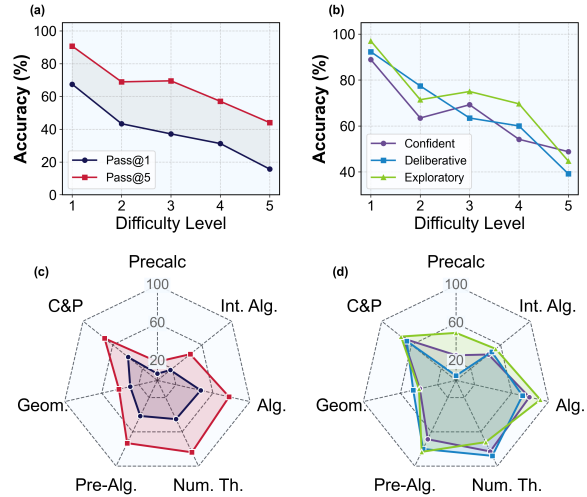


Figure 6: Problem difficulty and subject analysis. (a-d) Performance and routing analysis across levels and domains.

Subject-Specific Performance. The radar charts in Figure 6(c-d) reveal distinct cross-domain patterns. In (c), the Pass@5 benefit over Pass@1 varies substantially by subject: Pre-Algebra and Number Theory show large gaps, suggesting high solution path variability where sampling diversity helps, while Geometry exhibits smaller gaps. In (d), routing mode effectiveness varies by domain: Exploratory routing excels in Algebra and Pre-Algebra, while Confident routing performs comparably in Counting & Probability. Geometry shows the most uniform performance across all routing modes, suggesting that no single strategy holds a clear advantage for this subject.

4 Conclusion

We introduce STAR, a problem-aware strategy routing framework that decouples mathematical reasoning into strategy evaluation and targeted execution. It leverages multi-strategy performance profiling to construct MathStrat, a 13K strategy preference dataset. A strategy adaptor trained on this dataset performs policy routing at inference time. Extensive experiments with three LLMs across five datasets, combined with ablation studies and efficiency analysis, demonstrate the effectiveness, superiority, and scalability of STAR.

591 Limitations

592 While STAR demonstrates consistent improve-
593 ments across multiple benchmarks, some aspects
594 merit further investigation. The current routing
595 thresholds are calibrated for mathematical reason-
596 ing tasks, and exploring automated threshold adap-
597 tation methods could further enhance the frame-
598 work’s flexibility. Additionally, while our exper-
599 iments focus on mathematical reasoning, extend-
600 ing the strategy preference learning paradigm to
601 other reasoning domains presents a promising di-
602 rection. Finally, incorporating new reasoning strate-
603 gies would benefit from efficient adapter updating
604 mechanisms.

605 References

606 Wenhui Chen, Xueguang Zhao, Chang Shu, Hongjin
607 Chen, Zeqiu Lin, Yulei Wang, William Yang Wang,
608 and 1 others. 2023. Program of thoughts prompting:
609 Disentangling computation from reasoning for nu-
610 merical reasoning tasks. *Transactions on Machine
611 Learning Research*.

612 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian,
613 Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias
614 Plappert, Jerry Tworek, Jacob Hilton, Reiichiro
615 Nakano, Christopher Hesse, and John Schulman.
616 2021. Training verifiers to solve math word prob-
617 lems. *arXiv preprint arXiv:2110.14168*.

618 Aniket Didolkar, Anirudh Goyal, Nan Rosemary Ke,
619 Siyuan Guo, Michal Valko, Timothy Lillicrap, Danilo
620 Rezende, Yoshua Bengio, Michael Mozer, and San-
621 jeev Arora. 2024. Metacognitive capabilities of llms:
622 An exploration in mathematical problem solving. In
623 *NeurIPS*, pages 19783–19812.

624 Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon,
625 Pengfei Liu, Yiming Yang, Jamie Callan, and Gra-
626 ham Neubig. 2023a. Pal: Program-aided language
627 models. In *ICML*, pages 10764–10799.

628 Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon,
629 Yiming Yang, Jamie Callan, and Graham Neubig.
630 2023b. Pal: Program-aided language models. In
631 *ICML*, pages 10764–10799.

632 Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang,
633 Minlie Huang, Nan Duan, and Weizhu Chen. 2024a.
634 Tora: A tool-integrated reasoning agent for mathe-
635 matical problem solving. In *ICLR*.

636 Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang,
637 Minlie Huang, Nan Duan, Weizhu Chen, and 1 others.
638 2024b. Tora: A tool-integrated reasoning agent for
639 mathematical problem solving. In *ICLR*.

640 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul
641 Arora, Steven Basart, Eric Tang, Dawn Song, and

Jacob Steinhardt. 2021. Measuring mathematical
problem solving with the math dataset. In *NeurIPS*.

Aili Jin, Zhi-Yao Tang, Hong-Yu Zhang, Chen-Yu Wang,
and Ming Wan. 2024. T-rex: A new frontier for tool-
augmented reasoning with large language models. In
ICLR.

Zenan Li, Zhi Zhou, Yuan Yao, Yu-Feng Li, Chun
Cao, Fan Yang, Xian Zhang, and Xiaoxing Ma. 2024.
Neuro-symbolic data generation for math reasoning.
In *NeurIPS*, pages 23488–23515.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blun-
som. 2017. Program induction by rationale genera-
tion: Learning to solve and explain algebraic word
problems. In *ACL*, pages 158–167.

Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai
Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhu-
oshu Li, Yaofeng Sun, Chengqi Deng, Hanwei Xu,
Zhenda Xie, and Chong Ruan. 2024a. Deepseek-vl:
Towards real-world vision-language understanding.
arXiv preprint arXiv:2403.05525.

Zimu Lu, Aojun Zhou, Houxing Ren, Ke Wang,
Weikang Shi, Juntong Pan, Mingjie Zhan, and Hong-
sheng Li. 2024b. Mathgenie: Generating synthetic
data with question back-translation for enhancing
mathematical reasoning of llms. In *ACL*.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jian-
guang Lou, Chongyang Tao, Xiubo Geng, Qingwei
Lin, Shifeng Chen, and Dongmei Zhang. 2025. Wiz-
ardmath: Empowering mathematical reasoning for
large language models via reinforced evol-instruct.
In *ICLR*.

Jie Ma, Zhitao Gao, Qi Chai, Wangchun Sun, Pinghui
Wang, Hongbin Pei, Jing Tao, Lingyun Song, Jun Liu,
Chen Zhang, and 1 others. 2025a. Debate on graph:
a flexible and reliable reasoning framework for large
language models. In *AAAI*, pages 24768–24776.

Jie Ma, Ning Qu, Zhitao Gao, Rui Xing, Jun Liu,
Hongbin Pei, Jiang Xie, Linyun Song, Pinghui
Wang, Jing Tao, and 1 others. 2025b. Delibera-
tion on priors: Trustworthy reasoning of large lan-
guage models on knowledge graphs. *arXiv preprint
arXiv:2505.15210*.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su.
2020. A diverse corpus for evaluating and developing
english math word problem solvers. In *ACL*, pages
975–984.

Ha Nguyen and Vicki Allan. 2024. Using gpt-4 to
provide tiered, formative code feedback. In *ACM*,
page 958–964.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal.
2021. Are nlp models really able to solve
simple math word problems? *arXiv preprint
arXiv:2103.07191*.

695	Leonardo Ranaldi, Giulia Pucci, Barry Haddow, and Alexandra Birch. 2024. Empowering multi-step reasoning across languages via program-aided language models. In <i>EMNLP</i> .	750
696		751
697		
698		
699	Prashant Trivedi, Souradip Chakraborty, Avinash Reddy, Vaneet Aggarwal, Amrit Singh Bedi, and George K. Atia. 2025. Align-pro: A principled approach to prompt optimization for llm alignment. In <i>AAAI</i> , pages 27653–27661.	752
700		753
701		754
702		755
703		
704	Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. 2024. Measuring multimodal mathematical reasoning with math-vision dataset. In <i>NeurIPS</i> , pages 95095–95169.	756
705		757
706		758
707		759
708		760
709	Xuezhi Wang, Jason Wei, Dale Schuurmans, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In <i>ICLR</i> .	761
710		762
711		763
712		764
713	Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022a. Chain-of-thought prompting elicits reasoning in large language models. In <i>NeurIPS</i> .	765
714		766
715		767
716		768
717		769
718	Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In <i>NeurIPS</i> , pages 24824–24837.	770
719		771
720		772
721		773
722		774
723	Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, and Li Jiang. 2024. Autogen: Enabling next-gen llm applications via multi-agent conversation. In <i>ICLR</i> .	775
724		776
725		777
726		778
727	Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. 2025. Evaluating mathematical reasoning beyond accuracy. In <i>AAAI</i> .	779
728		780
729		781
730	Wenjing Xie, Xiaobo Liang, Juntao Li, Wanfu Wang, Kehai Chen, Qiaoming Zhu, and Min Zhang. 2025. From awareness to adaptability: Enhancing tool utilization for scientific reasoning. In <i>ACL</i> .	782
731		783
732		784
733		785
734	Huajian Xin, ZZ Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, and 1 others. 2025. Deepseek-prover-v1.5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. In <i>ICLR</i> .	786
735		787
736		788
737		789
738		
739		
740	An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. <i>arXiv preprint arXiv:2409.12122</i> .	790
741		791
742		792
743		793
744		794
745		795
746		796
747	Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with	797
748		798
749		799
	large language models. In <i>NeurIPS</i> , pages 11809–11822.	800
		801
		802
	Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. Tree of thoughts: Deliberate problem solving with large language models. In <i>NeurIPS</i> , pages 11809–11822.	803
		804
		805
	Jiacheng Ye, Shansan Gong, Liheng Chen, Lin Zheng, Jiahui Gao, Han Shi, Chuan Wu, Xin Jiang, Zhenguo Li, Wei Bi, and Lingpeng Kong. 2024. Diffusion of thought: Chain-of-thought reasoning in diffusion language models. In <i>NeurIPS</i> , pages 105345–105374.	806
		807
		808
	Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. In <i>ICLR</i> .	809
		810
		811
	Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhua Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. <i>arXiv preprint arXiv:2309.05653</i> .	812
		813
		814
	Beichen Zhang, Kun Zhou, Xilin Wei, Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. 2023. Evaluating and improving tool-augmented computation-intensive math reasoning. In <i>NeurIPS</i> , pages 23570–23589.	815
		816
		817
	Di Zhang, Jianbo Wu, Jingdi Lei, Tong Che, Jiatong Li, Tong Xie, Xiaoshui Huang, Shufei Zhang, Marco Pavone, Yuqiang Li, Wanli Ouyang, and Dongzhan Zhou. 2025. Llama-berry: Pairwise optimization for olympiad-level mathematical reasoning via o1-like monte carlo tree search. In <i>NAACL</i> , pages 7315–7337.	818
		819
		820
	Shaowei Zhang and Deyi Xiong. 2025. Backmath: Towards backward reasoning for solving math problems step by step. In <i>COLING</i> , pages 466–482.	821
		822
		823
	Zhehao Zhang, Jiaao Chen, and Diyi Yang. 2024. Darg: Dynamic evaluation of large language models via adaptive reasoning graph. In <i>NeurIPS</i> , pages 135904–135942.	824
		825
		826
	James Zhao, Yuxi Xie, Kenji Kawaguchi, Junxian He, and Michael Xie. 2023. Automatic model selection with large language models for reasoning. In <i>EMNLP</i> , pages 758–783.	827
		828
		829
		830
		831
		832
		833
		834
		835
		836
		837
		838
		839
		840
		841
		842
		843
		844
		845
		846
		847
		848
		849
		850
		851
		852
		853
		854
		855
		856
		857
		858
		859
		860
		861
		862
		863
		864
		865
		866
		867
		868
		869
		870
		871
		872
		873
		874
		875
		876
		877
		878
		879
		880
		881
		882
		883
		884
		885
		886
		887
		888
		889
		890
		891
		892
		893
		894
		895
		896
		897
		898
		899
		900
		901
		902
		903
		904
		905
		906
		907
		908
		909
		910
		911
		912
		913
		914
		915
		916
		917
		918
		919
		920
		921
		922
		923
		924
		925
		926
		927
		928
		929
		930
		931
		932
		933
		934
		935
		936
		937
		938
		939
		940
		941
		942
		943
		944
		945
		946
		947
		948
		949
		950
		951
		952
		953
		954
		955
		956
		957
		958
		959
		960
		961
		962
		963
		964
		965
		966
		967
		968
		969
		970
		971
		972
		973
		974
		975
		976
		977
		978
		979
		980
		981
		982
		983
		984
		985
		986
		987
		988
		989
		990
		991
		992
		993
		994
		995
		996
		997
		998
		999
		1000

et al., 2024; Lu et al., 2024b) and evolutionary rewriting of instructions (Luo et al., 2025). While effective for symbolic deduction, NLR’s reliance on unstructured text makes it prone to arithmetic and logical errors in computationally intensive problems. **Code-Augmented Reasoning** To address the computational limitations of NLR, CAR reframes mathematical problems as program generation tasks, offloading calculations to a deterministic code interpreter (Zhang et al., 2024). This is often implemented through prompting paradigms like Program-of-Thoughts (PoT) (Chen et al., 2023) or by fine-tuning models on interleaved text and code, as in Program-Aided Language Models (PAL) (Gao et al., 2023b). CAR excels at numerical precision but remains dependent on the initial natural language understanding for problem decomposition and program planning. **Tool-Integrated Reasoning** TIR extends the role of LLMs from solvers to agents that dispatch tasks to external tools like calculators or symbolic solvers. This approach, exemplified by works such as ToRA (Gou et al., 2024a), creates a call-verify-iterate loop that enhances robustness on high-difficulty problems. Recent studies (Jin et al., 2024; Wu et al., 2024) have also demonstrated strong performance by leveraging advanced integrated environments like the GPT-4 Code Interpreter for complex problem-solving (Nguyen and Allan, 2024). The primary trade-off for TIR is the increased complexity and latency associated with tool selection and orchestration. **Ensemble-Based Reasoning** To improve robustness with minimal engineering, lightweight ensemble methods aggregate multiple solution trajectories. The most prominent example is Self-Consistency, which samples multiple CoT paths and selects the answer via majority voting (Wang et al., 2023). Other works extend this by exploring more complex reasoning structures like a Tree of Thoughts (Yao et al., 2023b). These methods (Zhang et al., 2025; Yao et al., 2023a; Xia et al., 2025), however, are fundamentally forms of post-hoc selection, requiring the generation of multiple costly trajectories before aggregation and failing to identify the optimal strategy in advance. Related work has also explored selecting among reasoning modes before inference, such as routing between CoT and PAL using an external selector and assigning problems to predefined reasoning types via supervised classification. (Yue et al., 2023);(Zhao et al., 2023)

Algorithm 1 Adaptive Routing Policy at Inference

Require: Problem P ; Model M ; Strategies $\mathcal{S} = \{\sigma_1, \dots, \sigma_k\}$; Thresholds τ_c, τ_a
Ensure: Answer A ; Routing mode R ; Executed strategies Σ^*

```

1:  $\mathbf{p} \leftarrow M(P)$  ▷ Predict  $p(\sigma_i | P)$ 
2:  $i_{\max} \leftarrow \arg \max_i \mathbf{p}_i$ ;  $p_{\max} \leftarrow \mathbf{p}_{i_{\max}}$ ;  $\sigma_{\max} \leftarrow \sigma_{i_{\max}}$ 
3:  $i_2 \leftarrow \arg \max_{i \neq i_{\max}} \mathbf{p}_i$ ;  $p_2 \leftarrow \mathbf{p}_{i_2}$ ;  $\sigma_2 \leftarrow \sigma_{i_2}$ 
4: if  $p_{\max} \geq \tau_c \wedge (p_{\max} - p_2) \geq \tau_a$  then
5:    $R \leftarrow \text{CONFIDENT}$  ▷ Confident Routing
6:    $\Sigma^* \leftarrow \{\sigma_{\max}\}$ ;  $A \leftarrow \sigma_{\max}(P)$ 
7: else if  $p_{\max} \geq \tau_c \wedge (p_{\max} - p_2) < \tau_a$  then
8:    $R \leftarrow \text{DELIBERATIVE}$  ▷ Deliberative Routing
9:    $\Sigma^* \leftarrow \{\sigma_{\max}, \sigma_2\}$ 
10:   $A_1 \leftarrow \sigma_{\max}(P)$ ;  $A_2 \leftarrow \sigma_2(P)$ 
11:   $A \leftarrow \text{Vote}(\{A_1, A_2\})$ 
12: else
13:   $R \leftarrow \text{EXPLORATORY}$  ▷ Exploratory Routing
14:   $\Sigma^* \leftarrow \mathcal{S}$ ;  $\mathcal{A} \leftarrow \emptyset$ 
15:  for  $\sigma_i \in \mathcal{S}$  do
16:     $A_i \leftarrow \sigma_i(P)$ ;  $\mathcal{A} \leftarrow \mathcal{A} \cup \{A_i\}$ 
17:  end for
18:   $A \leftarrow \text{Vote}(\mathcal{A})$ 
19: end if
20: return  $(A, R, \Sigma^*)$ 

```

A.1 Adaptive Strategy Routing algorithm 854

This section provides the detailed pseudo-code for the STAR adaptive routing policy, as referenced in Section 2.2. 855
856
857

A.2 Preference Data Example 858

Figure 7 presents a representative case of our multi-strategy performance evaluation process for collecting training data. The case shows a trigonometric function analysis problem where we execute all four reasoning strategies and collect comprehensive performance metrics. 859
860
861
862
863
864

The data collection captures three complementary dimensions as described in our methodology: answer correctness (validity scores), process quality (redundancy measures), and computational efficiency (execution time and output length). As illustrated in this case, different strategies exhibit distinct performance profiles: NLR achieves moderate validity (0.23) but suffers from high redundancy (0.68), while TIR demonstrates better process quality with higher validity (0.29) and lower redundancy (0.72). CAR shows the poorest validity (0.18) with the highest redundancy (0.73), and EBR balances the highest validity (0.35) with acceptable redundancy (0.66). 865
866
867
868
869
870
871
872
873
874
875
876
877
878

The condensed output summaries reveal how each strategy approaches the same trigonometric problem differently: NLR relies on numerical approximation methods, CAR attempts algebraic sim- 879
880
881
882

plification, TIR uses analytical identities, and EBR combines multiple approaches. This diversity in solution paths, combined with varying performance across the three evaluation dimensions, provides rich supervision signals that enable our SA to learn nuanced associations between problem characteristics and strategy effectiveness. The multi-faceted evaluation ensures that our training data captures the essential trade-offs between correctness, reasoning quality, and computational cost.

A.3 Sensitivity Analysis of Score Aggregation Weights

The suitability score in Equation 2 aggregates three evaluation dimensions through weighted combination. To determine the optimal weight configuration, we conducted systematic ablation experiments on the same 200-problem validation set used for threshold optimization. We evaluated seven configurations representing different design choices: correctness-prioritized (0.70, 0.15, 0.15), balanced-moderate (0.60, 0.20, 0.20), fully uniform (0.33, 0.33, 0.33), quality-prioritized (0.20, 0.60, 0.20), efficiency-prioritized (0.20, 0.20, 0.60), quality-extreme (0.15, 0.70, 0.15), and efficiency-extreme (0.15, 0.15, 0.70). For each configuration, we measure Rank-1 Accuracy (percentage of problems where the top-ranked strategy matches the ground-truth best) and Spearman Correlation (rank correlation between predicted and ground-truth strategy rankings).

Figure 8 presents the results across the accuracy-stability space. The correctness-prioritized configuration (0.70, 0.15, 0.15) achieves the highest Rank-1 accuracy at 61.0%, outperforming quality-prioritized weighting (52.0%) by 9 percentage points. This substantial difference validates emphasizing correctness, which directly determines whether the selected strategy successfully solves the problem. Higher correctness weights improve Top-1 selection accuracy but reduce ranking stability. This reflects our focus on selecting the best strategy for deployment rather than achieving perfect ranking consistency. The moderate Spearman correlations (0.30-0.43) are expected given the limited ranking space with only 4 strategies and the similarity in strategy performance on individual problems. Configurations maintaining correctness weights between 0.33-0.70 demonstrate robust performance (56-61%), indicating stability within reasonable parameter ranges while exhibiting a clear optimal region. We select $w_c = 0.70$, $w_q = 0.15$,

$w_e = 0.15$ based on its empirical superiority and theoretical alignment with mathematical reasoning evaluation practices that prioritize correctness.

A.4 Threshold Parameter Optimization

To determine the optimal confidence threshold τ_c and ambiguity margin τ_a for our adaptive routing policy, we conducted a comprehensive grid search on a validation set comprising 200 problems sampled from MATH, GSM8K, AQUA-RAT, SVAMP, and ASDiv datasets to ensure coverage across different mathematical domains and difficulty levels. We evaluated $\tau_c \in [0.1, 0.7]$ and $\tau_a \in [0.02, 0.20]$ with step sizes of 0.05 and 0.01 respectively. Figure 9 shows the parameter optimization results across the two-dimensional parameter space. The contour plot reveals a well-defined optimal region at $\tau_c = 0.4$ and $\tau_a = 0.08$, achieving 78.0% Pass@1 accuracy on the validation set. The performance landscape exhibits several notable characteristics:

Confidence Threshold Sensitivity: The τ_c parameter shows an inverted-U relationship with performance (Figure 9, top right panel). Very low confidence thresholds ($\tau_c < 0.2$) result in over-conservative routing that fails to leverage high-confidence predictions effectively, achieving only 65% accuracy. Conversely, excessively high thresholds ($\tau_c > 0.5$) force the system into single-strategy execution even for uncertain predictions, degrading performance to 71%.

Ambiguity Margin Sensitivity: The τ_a parameter demonstrates a sharp optimum (Figure 9, bottom right panel). The optimal value of 0.08 creates an appropriate balance for distinguishing competitive strategy scenarios. Lower values ($\tau_a < 0.06$) cause excessive deliberative routing even when strategy preferences are clear, while higher values ($\tau_a > 0.12$) prevent beneficial dual-strategy verification in genuinely competitive cases.

The relatively narrow optimal region indicates that careful parameter tuning is essential for achieving peak performance. However, the clear convex structure around the optimum suggests stable convergence during hyperparameter search. We use these validated parameters ($\tau_c = 0.4$, $\tau_a = 0.08$) across all experimental settings to ensure fair comparison with baseline methods.

Example of Collected Preference Data					
Strategy	Time	Length	Validity	Redundancy	Condensed Output Summaries
NLR	7.6281	2151	0.23	0.68	<ol style="list-style-type: none"> The equation $\frac{x}{100} = \sin x$ is analyzed by comparing the linear function $f(x) = x/100$ and the periodic function $g(x) = \sin x$. Since $\sin x \leq 1$, intersections occur only for $x \leq 100$, and within this range, the approximately 31.83 periods of $\sin x$ suggest about 64 intersections (two per period plus endpoints). A Python script numerically checks for sign changes between the functions over the interval $[-100, 100]$ using a high-resolution grid of 10,000 points. The numerical computation yields 63 intersections, correcting the initial analytical estimate and providing the final solution count.
CAR	12.0192	3197	0.18	0.73	<ol style="list-style-type: none"> The function is first simplified using trigonometric identities to the core form $f(x) = \sqrt{2}\sin(2x + \frac{\pi}{4})$ for easier analysis. The period is calculated directly from the simplified function's angular frequency, yielding a smallest positive period of π. By analyzing the sine function over the transformed interval of its argument, the maximum of $\sin(u)\sin(u)$ is found to be 1 and the minimum to be $-\frac{\sqrt{2}}{2}$. The extreme values are scaled by the amplitude $\sqrt{2}$ to give the final results: a maximum of $\sqrt{2}$ and a minimum of -1 for $f(x)$.
TIR	7.0319	1907	0.29	0.72	<ol style="list-style-type: none"> The original trigonometric expression is simplified using sum-to-product and double-angle identities to obtain $f(x) = \sin 2x + \cos 2x$. The simplified function $\sqrt{2}\sin(2x + \frac{\pi}{4})$ has a period of π, derived from the standard sine period formula $2\pi/ B$. On the interval $x \in [-\frac{\pi}{4}, \frac{\pi}{4}]$, the argument $2x + \frac{\pi}{4}$ ranges from $-\frac{\pi}{4}$ to $\frac{3\pi}{4}$, covering key monotonic segments of the sine function. The maximum value of $f(x)$ is $\sqrt{2}$ (when $\sin(\cdot) = 1$) and the minimum is -1 (at the left endpoint where $\sin(-\frac{\pi}{4}) = -\frac{\sqrt{2}}{2}$).
EBR	15.5082	4367	0.35	0.66	<ol style="list-style-type: none"> The equation is analyzed by comparing the linear function $f(x) = x/100$ with the periodic sine function $g(x) = \sin x$. Intersections occur only when $x \leq 100$, since the sine function is bounded between -1 and 1. Within this range, $\sin x \sin x$ completes about 31.83 periods, suggesting approximately 64 intersections (two per period plus endpoints). Python code confirms the actual number of intersections is 63, refining the initial estimate.

Figure 7: Strategy preference data collection showing multi-strategy performance evaluation for a trigonometric function problem. Each strategy exhibits distinct profiles across the three evaluation dimensions: correctness, process quality, and computational efficiency.

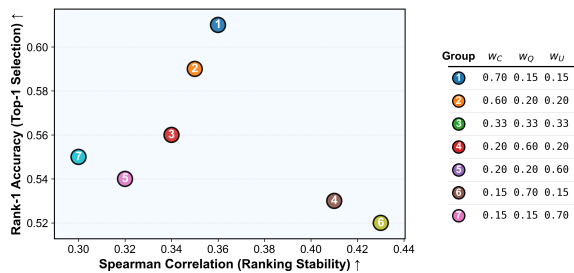


Figure 8: Sensitivity analysis of score aggregation weights on the validation set. The scatter plot shows Rank-1 accuracy versus Spearman correlation across seven weight configurations.

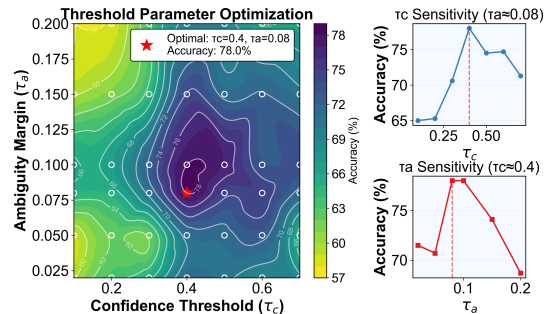


Figure 9: Threshold parameter optimization on the validation set. Left: Contour plot showing accuracy across the $\tau_c - \tau_a$ space with the optimum marked by a red star. Right: Sensitivity analysis showing accuracy as each parameter varies while holding the other at its optimum. The validation set contains 200 problems sampled from MATH, GSM8K, AQUA-RAT, SVAMP, and ASDiv.

981 A.5 Strategy Performance and Correlation

982 Analysis

983 Figure 10 presents the mean performance scores and inter-strategy correlations across four mathe-
 984 matical reasoning datasets. The left panel shows that all strategies achieve similar average suitabil-
 985 ity scores (ranging from 0.18 to 0.31), with no single strategy demonstrating clear dominance across
 986 datasets. The right panel displays correlation matrices revealing predominantly low or negative correla-
 987 tions between strategies, indicating complementary rather than redundant capabilities. Notably,
 988
 989
 990
 991
 992

993 TIR exhibits consistent negative correlations with
 994 other strategies across most datasets, suggesting its
 995 specialized applicability to distinct problem char-
 996 acteristics. These patterns validate the necessity of
 997 adaptive strategy selection, as the low inter-strategy
 998 correlations demonstrate that different approaches
 999 excel on different problem subsets.

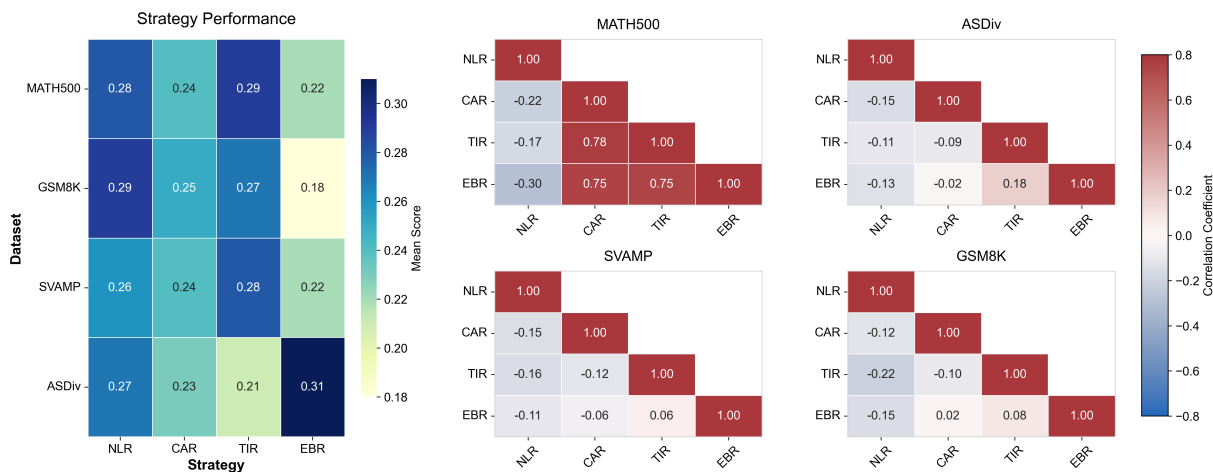


Figure 10: Strategy performance and correlation analysis. Left: Mean suitability scores by strategy and dataset. Right: Inter-strategy correlation matrices for each dataset. Low correlations indicate complementary strategy capabilities.

A.6 Case Study: Strategy Adapter Prediction

Figure 11 presents a representative example demonstrating how SA evaluates different reasoning approaches for a logarithmic geometry problem. The problem requires finding the x -coordinate where a horizontal line intersects the curve $f(x) = \ln x$, involving both coordinate geometry concepts and logarithmic calculations.

The Strategy Adapter assigns suitability scores that align well with actual strategy performance: TIR receives the highest score (0.63) and successfully solves the problem through tool-assisted computation, while NLR and CAR receive lower scores (0.28 and 0.16 respectively) and both fail to produce correct solutions. EBR achieves a moderate score (0.58) and succeeds through ensemble reasoning. This case exemplifies how the adapter learns to associate problem characteristics—such as the need for precise numerical computation in logarithmic contexts—with appropriate reasoning strategies.

The correlation between prediction scores and actual outcomes validates our approach of using suitability scores to guide adaptive routing decisions, demonstrating that higher SA scores generally correspond to better strategy performance on specific problem instances.

A.7 Error Case Analysis

To better understand STAR’s limitations, we analyze representative failure cases that reveal two primary error patterns. Figure 12 illustrates a high-confidence routing error where the SA assigns the highest score to TIR (0.357) for a trigonometric product calculation problem. Despite this confident

prediction, TIR produces an incorrect numerical result (0.0195 instead of $\frac{9}{256}$), demonstrating that high discriminator scores do not always guarantee correct solutions. This case reveals a calibration issue where the adapter overestimates strategy suitability based on problem surface features (presence of trigonometric functions suggesting tool usage) without anticipating execution failures in symbolic manipulation.

Figure 13 presents an aggregation error in dual-path routing. For a complex number rotation problem, the adapter correctly identifies CAR (0.330) and NLR (0.285) as competitive strategies. However, while both strategies are executed, CAR produces a formatting error (6,-5.0000000000000001) and NLR generates the correct answer (6-5i). The routing policy selects CAR’s output based on higher predicted confidence, resulting in failure. This case highlights a limitation in our confidence-based aggregation mechanism: when strategies produce inconsistent answers, prediction scores alone may not reliably indicate correctness. These failure patterns suggest that future work could benefit from incorporating runtime validation signals or more sophisticated answer consistency checking beyond confidence-weighted selection.

Strategy-Specific Responses and Preference Scores on a Benchmark Math Problem	
<p>Problem: Two points $A(x_1, y_1)$ and $B(x_2, y_2)$ are chosen on the graph of $f(x) = \ln x$, with $0 < x_1 < x_2$. The points C and D trisect \overline{AB}, with $AC < CB$. Through C a horizontal line is drawn to cut the curve at $E(x_3, y_3)$. Find x_3 if $x_1 = 1$ and $x_2 = 1000$.</p> <p>Golden Answer: 10</p>	
NLR	Suitability Score : 0.28
<p>Text: The coordinates of A and B are $(1, 0)$ and $(1000, \ln 1000)$, respectively...Substituting the values, we get: $6(x + \frac{1}{2}) + (y - \frac{3}{2}) + 10(z + \frac{1}{2}) = 0$ $6x + 3 + y - \frac{3}{2} + 10z + 5 = 0$ $6x + y + 1$</p>	✘
CAR	Suitability Score : 0.16
<p>Text: Two points $A(x_1, y_1)$ and $B(x_2, y_2)$ are chosen on the graph of $f(x) = \ln x$, with $0 < x_1 < x_2$. ...So, $y_3 = \ln(x_3) = y_c$ # Therefore, $x_3 = \exp(y_c)$ $x_3 = \text{math.exp}(y_c)$ result = x_3 return result ````</p>	✘
TIR	Suitability Score : 0.63
<p>Text: To solve this problem, we need to find the x-coordinate of the point $(E(x_3, y_3))$ where a horizontal line through point (C) intersects the curve $f(x) = \ln x$...we get: $x_3 = e^{\frac{\ln 1000}{3}} = \left(e^{\ln 1000}\right)^{\frac{1}{3}} = 1000^{\frac{1}{3}} = 10$ \ So, the x-coordinate (x_3) is (10).Let's confirm this with Python code...</p>	✔
EBR	Suitability Score : 0.58
<p>Text: To solve this problem, we need to find the x-coordinate of the point $(E(x_3, y_3))$ where a horizontal line through point (C) intersects the curve $f(x) = \ln x$...Therefore, we need to solve $(\ln x_3 = \ln 10)$, which gives $(x_3 = 10)$. Let's confirm this with Python code...</p>	✔

Figure 11: Case study showing SA suitability scores and corresponding strategy outcomes for a logarithmic geometry problem. Higher suitability scores correlate with successful problem-solving performance.

Error Case Analysis – High-Confidence Routing Errors
<p>Problem: Find $\sin 20^\circ \sin 40^\circ \sin 60^\circ \sin 80^\circ \sin 100^\circ \sin 120^\circ \sin 140^\circ \sin 160^\circ$.</p> <p>Golden Answer: $\frac{9}{256}$</p> <p>Mode: prism</p> <p>Chosen_path: prism-single-path</p> <p>Strategies_actually_executed_in_path: [TIR]</p> <p>Discriminator_scores: { NLR: 0.236978, CAR: 0.221464, TIR: 0.357294, EBR: 0.184264 }</p> <p>Raw_llm_output: \n\nTo solve the problem, we can use trigonometric identities and properties. The given expression is:\n\n$\sin 20^\circ \sin 40^\circ \sin 60^\circ \sin 80^\circ \sin 100^\circ \sin 120^\circ \sin 140^\circ \sin 160^\circ$\n\nWe can use the identity ...Now we need to find $(\sin 20^\circ)^2 (\sin 40^\circ)^2$. We can use the product-to-sum identities. Specifically, ...Now we can multiply this by $(\frac{3}{4})$:\n\n$(\sin 20^\circ)^2 (\sin 40^\circ)^2 (\sin 60^\circ)^2 = \left(\frac{1}{2}\right) \cos(20^\circ) - \left(\frac{1}{4}\right)^2 \cdot \left(\frac{3}{4}\right)$\n\nThis expression is quite complex to simplify further without numerical evaluation. However, we can use Python to evaluate this expression accurately. Let's use Python to compute the value.\n\npython\nimport math\n# Define the angles in degrees\nangles = [20, 40, 60, 80, 100, 120, 140, 160]\n# Convert degrees to radians\nangles_rad = [math.radians(angle) for angle in angles]\n\n# Calculate the product of sines,...,\n\nprint(product_of_sines)</p> <p>Extracted_answer: 0.01953125</p>

Figure 12: High-confidence routing error: the SA assigns the highest score to TIR, which executes but produces an incorrect numerical result.

Error Case Analysis – Aggregation Errors
<p>Problem: Let $z = 2 + \sqrt{2} - (3 + 3\sqrt{2})i$, and let $c = 2 - 3i$. Let w be the result when z is rotated around c by $\frac{\pi}{4}$ counter-clockwise. In units of 0.6 cm, pair C, W, Z; $Z = (2 + \sqrt{2}, -3 - 3\sqrt{2})$; $C = (2, -3)$; $W = \text{rotate}(45, C)(Z)$. Draw $Z-C-W$. Label C, N; W, SE; Z, S; $\frac{\pi}{4}$; $C + (0.6, -1)$. Find w.</p> <p>Golden Answer: 6 - 5i</p>
<p>Mode: prism Chosen_path: prism-dual-path Strategies_actually_executed_in_path: [CAR, NLR]</p>
<p>Discriminator_scores: { NLR: 0.284763, CAR: 0.329766, TIR: 0.20969, EBR: 0.175781 }</p>
<p>Strategy_name: CAR Raw_llm_output: <pre>import cmath\n\n # Given complex numbers\n z = 2 + cmath.sqrt(2) - (3 + 3 * cmath.sqrt(2)) * 1j\n c = 2 - 3 * 1j\n\n # Rotate z around c by pi/4 counter-clockwise\n angle = cmath.pi / 4\n w = c + (z - c) * cmath.exp(1j * angle)\n\n result = w.real, w.imag\n return result\n```\n</pre> Extracted_answer: (6,-5.000000000000001) Is_correct_intermediate_answer: FALSE</p>
<p>Strategy_name: NLR Raw_llm_output: Let's think step by step\nWe write z as $2 + \sqrt{2} - (3 + 3\sqrt{2})i$ and c as $2 - 3i$. The difference $z - c$ is ..., To rotate this complex number by $\frac{\pi}{4}$, we multiply by ..., This gives..., we add c to get $w = (4 - 2i) + (2 - 3i) = \boxed{6 - 5i}$. The answer is 6-5i\n</p> <p>Extracted_answer: 6-5i Is_correct_intermediate_answer: TRUE</p>

Figure 13: Aggregation error: in dual-path routing, CAR (higher score) produces malformed output while NLR (lower score) yields the correct answer, but score-based selection fails.