Stepwise Reasoning Checkpoint Analysis: A Test Time Scaling Method to Enhance LLMs' Reasoning

Zezhong Wang[1,4*], **Xingshan Zeng**[2†], **Weiwen Liu**[3], **Yufei Wang**[2], **Liangyou Li**[2], **Yasheng Wang**[2], **Lifeng Shang**[2], **Xin Jiang**[2], **Qun Liu**[2], **Kam-Fai Wong**[1,4]
[1]The Chinese University of Hong Kong
[2]Huawei Noah's Ark Lab, [3]Shanghai Jiao Tong University
[4]MoE Key Laboratory of High Confidence Software Technologies, CUHK, China
zzwang@se.cuhk.edu.hk, zeng.xingshan@huawei.com

Abstract

Mathematical reasoning through Chain-of-Thought (CoT) has emerged as a powerful capability of Large Language Models (LLMs), which can be further enhanced through Test-Time Scaling (TTS) methods like Beam Search and DVTS. However, these methods, despite improving accuracy by allocating more computational resources during inference, often suffer from path homogenization and inefficient use of intermediate results. To address these limitations, we propose Stepwise Reasoning Checkpoint Analysis (SRCA), a framework that introduces checkpoints between reasoning steps. It incorporates two key strategies: (1) Answer-Clustered Search, which groups reasoning paths by their intermediate checkpoint answers to maintain diversity while ensuring quality, and (2) Checkpoint Candidate Augmentation, which leverages all intermediate answers for final decision-making. Our approach effectively reduces path homogenization and creates a fault-tolerant mechanism by utilizing high-quality intermediate results. Experimental results show that SRCA improves reasoning accuracy compared to existing TTS methods across various mathematical datasets.

1 Introduction

Large Language Models (LLMs) have demonstrated mathematical reasoning capabilities through Chain-of-Thought (CoT) (Wei et al., 2022). Recent studies indicate that Test Time Scaling (TTS), which expands test-time computing resources to allocate more reasoning budget through methods such as Beam Search (Snell et al., 2024) and Diverse Verifier Tree Search (DVTS) (Beeching et al., 2024), can significantly improve accuracy in mathematical reasoning tasks (Ji et al., 2025; Zhao et al., 2024; Chen et al., 2025a). These methods allow LLMs to sample multiple candidates at each reasoning step and score them using a process reward model (PRM) (Xi et al., 2024; Wu et al., 2024a; Wang et al., 2024b; Zhang et al., 2025a). According to their strategies, they select high-scoring steps to continue reasoning, thus overcoming the limitations of single-path reasoning.

However, current methods face two key challenges in practice. First, maintaining diversity in the sampled reasoning paths is both crucial and arduous (Misaki et al., 2025; Li et al., 2023). Even though the model generates multiple candidate paths, the chosen ones usually follow similar reasoning directions. This happens because the reward mechanism favors local optimal solutions, causing the search process to converge too early and fail to explore diverse reasoning patterns (Hooper et al., 2025; Zeng et al., 2025). Second, existing methods underutilize intermediate reasoning results: many intermediate branches are discarded during the search, and only a few complete paths are used in the final decision, leading to a waste of computational resources (Wang et al., 2024c; Zhang et al., 2025b).

To address these issues, we propose **Stepwise Reasoning Checkpoint Analysis (SRCA)**. We introduce reasoning checkpoints as a foundational technique and propose a searching method and a decision-enhancement strategy based on it. We inject "checkpoints" after each reasoning step. Specifically, once a step is completed, we temporarily interrupt the reasoning process and append the prompt "*So the answer is*" to the current reasoning steps, compelling the model to generate an intermediate prediction rather than continuing its reasoning process, as illustrated in the upper right corner of Figure 1. Using the intermediate answers collected at these checkpoints, we further propose an **Answer-Clustered Search** strategy. We group multiple reasoning steps sampled at the current checkpoint according to their detected answers, and

[*]Work done during internship at Huawei Noah's Ark Lab.
[†]Corresponding author

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 17135–17148
November 4-9, 2025

retain high-quality reasoning steps from each group for further extension. This approach allows us to maintain multiple potential reasoning paths leading to different answers, thus increasing the diversity of reasoning processes and mitigating the issue of path homogenization. Additionally, we introduce the **Checkpoint Candidate Augmentation** strategy. By collecting all intermediate answers generated at checkpoints, we expand the pool of candidate reasoning paths, allowing these intermediate results to participate in the final decision-making process. In this way, we fully utilize all high-quality intermediate results generated during reasoning, creating a fault-tolerant mechanism. Even if subsequent reasoning deviates, the retained high-quality intermediate predictions may still lead to the correct answer.

The contributions of this work can be summarized as follows:

- We introduce the concept of reasoning checkpoints, providing a new methodology for analyzing and improving LLM reasoning processes during test time.
- Based on this concept, we develop SRCA, a framework that effectively addresses both path diversity and computational efficiency challenges.
- We conduct extensive experiments that demonstrate the superiority of our approach and provide valuable insights for future research in Test-Time Scaling.

2 Related Works

As enthusiasm for scaling pre-training computation wanes, Test-Time Scaling (TTS) has emerged as a key research focus (Wang et al., 2024a; Wu et al., 2024b; Chen et al., 2025a). TTS allocates additional computation during inference to improve performance, significantly enhancing LLMs' problem-solving capabilities across specialized and general tasks. Some TTS approaches use training to encourage LLMs to generate more extensive outputs for deeper reasoning (Guan et al., 2025; Xi et al., 2024). These methods create synthetic data, including long chain-of-thought (Chen et al., 2025b; Xiang et al., 2025) and reflection-based examples (Bi et al., 2025; Zhang et al., 2024; Yu et al., 2024), to fine-tune LLMs, shifting their behavior from rapid responses to more deliberate reasoning.

Another category is training-free tree search, which forms the primary focus of this work (Luo et al., 2024; Wan et al., 2024; Guan et al., 2024). These methods dynamically guide the LLM's reasoning process using external verifiers or PRMs (Jiang et al., 2024; Uesato et al., 2022; Setlur et al., 2024). Snell et al. (2024) introduced Beam Search to explore the reasoning space, where PRM evaluates each reasoning step and maintains a fixed number of promising paths based on the beam width. Building upon this foundation, subsequent research (Beeching et al., 2024) proposed Diverse Verifier Tree Search (DVTS), which offers a notable improvement (Liu et al., 2025). Instead of maintaining a single search beam, DVTS operates multiple search trees simultaneously, selecting and expanding the most promising reasoning path within each tree. Tree search algorithms, however, face two crucial challenges: the diversity problem and the utilization problem.

The diversity problem arises when PRMs inadvertently suppress the LLM's sampling diversity (Chen et al., 2025b; Zheng et al., 2024). This occurs because only high-scoring paths are retained, and these paths often share similar problem-solving approaches. This issue is further exacerbated by the inherent biases in the imperfect PRMs or verifier (He et al., 2025; Zheng et al., 2024). The utilization problem arises because tree search algorithms explore numerous paths, but typically only one contributes to the final result. This leads to many branches and intermediate processes being discarded, with utilization efficiency decreasing as search scale increases. The challenge is to efficiently integrate generated reasoning overhead (Wang et al., 2024c; Zhang et al., 2025b; Sui et al., 2025). This issue has evolved into the "overthinking problem," where LLMs waste resources on simple problems, potentially leading to performance degradation through error accumulation (Li et al., 2024; Wu et al., 2025; Huang et al., 2025; Gan et al., 2025; Aggarwal and Welleck, 2025).

To address these dual challenges, we propose two novel strategies. First, the Answer Cluster Search algorithm is designed to enhance the diversity of tree search processes. Second, we introduce the Check Point Augmentation strategy to preserve high-quality intermediate reasoning processes for reuse in final answer decision-making, thereby addressing the low utilization problem inherent in tree search methodologies.

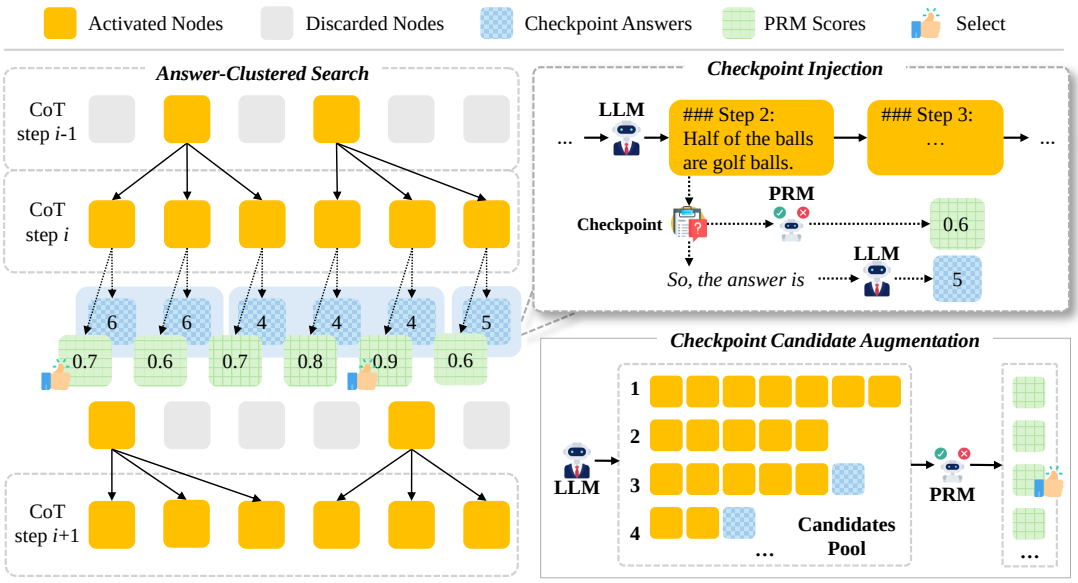

Figure 1: Overview of SRCA. Top-right: The checkpoint operation, which serves as the atomic operation in SRCA. Left: Illustration of ACS strategy at step i, where $N = 6$ and $M = 2$. Retrieved reasoning steps are clustered into three groups based on their checkpoint answers (indicated by different shades), with the highest-scoring nodes selected from clusters with answers 6 and 4 for subsequent reasoning. Bottom-right: CCA strategy, where paths 3 and 4 represent high-quality intermediate reasoning steps collected by CCA.

3 Methodology

In this section, we introduce three key techniques, Checkpoint Injection, which serves as the atomic operation in SRCA, Answer-Clustered Search (ACS), and Checkpoint Candidate Augmentation (CCA), to improve LLM's reasoning path searching.

3.1 Checkpoint Injection

We introduce a dynamic intervention mechanism to analyze the model's reasoning trajectory through checkpoint injection. As shown in the upper top-right part of Figure 1, the core procedure begins by monitoring the model's output stream for predefined step delimiter tokens (e.g., *"### Step"*), which indicate the completion of a logical reasoning unit. Upon detecting such tokens, we inject a checkpoint to temporarily suspend autoregressive generation. At each checkpoint position, a fixed prompt template x_{ckpt} = *"So, the answer is "* is inserted to force the model to generate an intermediate prediction based solely on the accumulated context up to that step. The model's immediate response to x_{ckpt} is recorded as a checkpoint answer a_t at step t, after which the LLM rolls back the generation state to the original checkpoint position. This rollback operation ensures the elimination of checkpoint influence from the ongoing reasoning process while preserving the model's KV cache for continued generation (Wang et al., 2025). The checkpoint answers subsequently serve as crucial criteria for path similarity assessment and grouping in the ACS strategy, while also being collected by the CCA method to enrich the final answer candidate pool.

3.2 Answer-Clustered Search

Similar to Beam Search (Snell et al., 2024), The Answer-Clustered Search (ACS) evaluates and retains a select few of the multiple reasoning steps sampled by the LLM for further reasoning. It enhances reasoning diversity through stepwise answer-guided clustering. In the following part, we will detail the four key steps of ACS using the running case shown on the left side of Figure 1.

1. Sampling. At each reasoning step t, we first determine the branching factor: for the initial step ($t = 1$), the LLM samples N candidate reasoning paths; for subsequent steps, each of the M surviving beams generates N/M sub-paths, maintaining a total budget of N paths. This set of paths is defined as $\{p_t^{(j)}\}_{j=1}^{N}$. Figure 1 illustrates the case where $N = 6$ and $M = 2$.

2. Clustering. All N paths undergo Checkpoint Injection at step t, yielding checkpoint answers $\{a_t^{(j)}\}_{j=1}^{N}$. These paths are clustered into groups $G = \{C_1, C_2, ..., C_k\}$ where $C_i = \{j | a_t^{(j)} = a_c\}$, forming answer-homogeneous clusters. The group-

Algorithm 1: Answer-Clustered Search

Input: Sampling budget N, beam width M, candidate set $\{p_t^{(j)}\}_{j=1}^{N}$
Output: Selected paths set $\mathcal{P}$
// Checkpoint Injection & Scoring
for $j \leftarrow 1$ **to** N **do**
  $a_t^{(j)} \leftarrow \texttt{SRCA}(p_t^{(j)})$;
  $s_t^{(j)} \leftarrow \texttt{PRM}(p_t^{(j)})$;
end
// Clustering& Sorting
$G = \{C_1, ..., C_k\}$ where $C_i = \{j | a_t^{(j)} = a_i\}$;
for $C_i \in G$ **do**
  $S_i \leftarrow \sum_{j \in C_i} s_j$;
end
$G \leftarrow \{C_i : S_i \geq S_{i+1}\}_{i=1}^{k}$;
// Round-robin Selection
$\mathcal{P} \leftarrow \emptyset$;
while $|\mathcal{P}| < M$ **do**
  for $C_i \in G$ **do**
    $j^* \leftarrow \text{argmax}_{j \in C_i} s_j$;
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{p^{(j^*)}\}$;
    $C_i \leftarrow C_i \setminus \{p^{(j^*)}\}$;
    if $|\mathcal{P}| = M$ **then**
      break;
    end
  end
end

ing results are marked with shading in Figure 1.

3. Scoring. A PRM assigns scores s_j to each path $p_t^{(j)}$, with cluster C_i's aggregate score computed as $S_i = \sum_{j \in C_i} s_j$. This approach is similar to a stepwise Weighted Best-of-N implementation.

4. Selection. Clusters are sorted by S_i in descending order, while paths within each cluster are ranked by s_j. Then we sequentially select top-ranked paths across clusters via round-robin sampling: starting from the highest cluster, we pick the top path from each cluster, cycling back when reaching the last cluster until M paths are selected.

This resource-aware branching prioritizes high-quality clusters while maintaining inter-cluster diversity. The cyclic selection mechanism prevents dominance by single-answer clusters and enables early identification of divergent reasoning trajectories. We provide a more rigorous process in Algorithm 1.

3.3 Checkpoint Candidate Augmentation

The proposed Checkpoint Candidate Augmentation (CCA) aims to maximize the use of reasoning resources and enhance the diversity of candidate answers by integrating the checkpoint answers from intermediate reasoning steps. Traditional Beam Search methods retain only a fixed number, i.e., M, of complete reasoning paths as the final candidate set, which leads to the discard of many unfinished intermediate branches. To address this issue, our method continuously collects intermediate answers generated at all checkpoints during the ACS and reconstructs the corresponding truncated reasoning paths into valid candidate paths. Specifically, for each intermediate answer $a_t^{(j)}$ produced at a checkpoint, we concatenate it with the current reasoning path $p_t^{(j)}$ to form a candidate path with a complete logical chain:

$$\hat{p}_t^{(j)} = p_t^{(j)} \oplus x_{\text{ckpt}} \oplus a_t^{(j)} \tag{1}$$

where $\oplus$ represents string concatenation.

All candidate paths, including the original complete paths and the newly added intermediate paths, are uniformly scored by the PRM, and the path with the highest score is selected as the model output. This method offers two main advantages: first, by incorporating prediction results from the intermediate inference process into the candidate set, it significantly improves the utilization of computational resources already spent; second, by retaining intermediate answers at various stages, it establishes an effective fault tolerance mechanism. Even if the LLM makes mistakes in subsequent steps, it may still arrive at the correct answer through the retained high-quality intermediate predictions. On the other hand, CCA can effectively mitigate issues such as overthinking, increasingly erroneous reasoning, and repetitive outputs in LLMs.

4 Experiments

We conducted comparative experiments on four mathematical reasoning datasets and against four Test-Time Scaling baselines.

4.1 Settings

In the experiments, we used four datasets: GSM8K, MATH500, AIME, and OlympiadBench. Two different-sized LLMs were tested in total, specifically `Llama-3.2-1B-Instruct` (MetaAI, 2024) and `Qwen3-0.6B` (Yang et al., 2025). For the PRM, we adopted the model fine-tuned by DeepSeek, `Llama3.1-8B-PRM-Deepseek-Data` (Xiong et al., 2024) and `Skywork-o1-Open-PRM-Qwen-2.5-7B` released by Skywork (o1 Team, 2024).

We compared our method against several TTS algorithms, including Greedy Search, Self-Consistency (Wang et al., 2023), Best-of-N (BoN) (Brown et al., 2024), Weighted BoN (Brown et al., 2024), Beam Search (Snell et al., 2024), and

Models & TTS	GSM8K	MATH500	AIME	OlympiadBench
Independent Sampling				
Llama-3.1-70B-Instruct (Grattafiori et al., 2024)	**95.10**	65.00	36.66	27.70
Llama-3.2-1B-Instruct (MetaAI, 2024)	43.75	24.40	3.22	4.59
w. Self-Consistency (Wang et al., 2023)	57.70	39.80	8.57	11.70
TTS Llama-3.2-1B-Instruct w. Llama3.1-8B-PRM-Deepseek-Data				
BoN (Brown et al., 2024)	80.36	46.20	11.04	13.48
Weighted BoN (Brown et al., 2024)	65.50	46.40	10.50	13.63
Beam Search (Snell et al., 2024)	84.84	52.00	19.07	18.07
DVTS (Beeching et al., 2024)	83.47	52.60	20.68	19.40
SRCA (Ours)	85.60	53.40	24.97	20.74
TTS Llama-3.2-1B-Instruct w. Skywork-o1-Open-PRM-Qwen-2.5-7B				
BoN (Brown et al., 2024)	80.74	55.20	25.08	18.22
Weighted BoN (Brown et al., 2024)	76.72	52.60	28.08	18.67
Beam Search (Snell et al., 2024)	84.99	63.20	26.82	23.89
DVTS (Beeching et al., 2024)	84.00	64.80	29.03	25.82
SRCA (Ours)	85.97	**65.20**	**39.71**	**27.75**

Table 1: Comparison of TTS results. The upper section shows the greedy search results for 1B and 70B models, and we additionally report the self-consistency performance of the 1B model with $N = 128$. The lower section shows results from the 1B model combined with various TTS methods and two PRMs, also with $N = 128$. Numbers indicate accuracy (%). Best overall performance on each dataset is marked in **bold**, while best performance within each group is underlined.

Diverse Verifier Tree Search (DVTS) (Beeching et al., 2024). Among these methods, Beam Search maintains N paths and selects M highest-scoring ones for expansion, with each generating N/M sub-paths. DVTS extends this by initializing M subtrees and sampling N/M paths per step within each subtree, enhancing path diversity through subtree isolation. Other baseline methods are standard approaches in the field; their detailed descriptions can be found in Appendix A.

For all the sampling-based methods, set temperature = 0.8 and top_p = 0.9. We use $N = 16$ and $N = 64$ for sampling times to assess the effect of sampling scale on reasoning performance. For the methods involving path selection, such as Beam Search, DVTS, and SRCA, the beam width M is fixedly set to 4, that is, the 4 candidate paths with the highest scores are retained at each reasoning step for subsequent expansion.

Since the PRM can collect the step-level scores of the complete reasoning path to form a score sequence, there are various ways to determine the final score of the path, such as taking the sum, accumulation, minimum value of the sequence scores, and the score of the last step. In this experiment, the score of the last step in the path is used as the path score. The effects of these configurations on experimental results are discussed in Appendix B.2.

4.2 Results

Table 1 shows the performance of `Llama-3.2-1B-Instruct` with various TTS methods on four mathematical datasets ($N = 128$, $M = 4$). We also include results from `Llama-3.1-70B-Instruct` for comparison.

4.2.1 Scaling with SRCA: Small Models Can Outperform Larger Ones

SRCA consistently outperforms other TTS methods across all datasets, regardless of the PRM used. With DeepSeek PRM, SRCA achieves approximately 10% absolute improvement over the BoN baseline. The improvement is particularly notable on AIME, where SRCA shows a 43% relative performance gain over DVTS. Remarkably, when using Skywork PRM, our 1B model with SRCA outperforms the 70B model on MATH500, AIME, and OlympiadBench, only falling behind on the simpler GSM8K dataset. This demonstrates SRCA's effectiveness in enabling smaller models to compete with larger ones. On the other hand, The choice of PRM also impacts performance, with Skywork PRM generally yielding better results than DeepSeek PRM across all TTS methods. This suggests that future advances in PRM development could lead to further performance improvements in TTS methods. More results of SRCA on Qwen3-0.6B are presented in Appendix B.3.

4.2.2 Expanding Sampling Times: SRCA Has Higher Efficiency.

We test various TTS methods with sampling times $N = 16$, 32, 64, and 128. The results are shown in Figure 2. SRCA demonstrates superior efficiency by requiring fewer samples to achieve comparable accuracy. With DeepSeek PRM on MATH500, SRCA achieves 51.2% accuracy at $N = 16$, out-

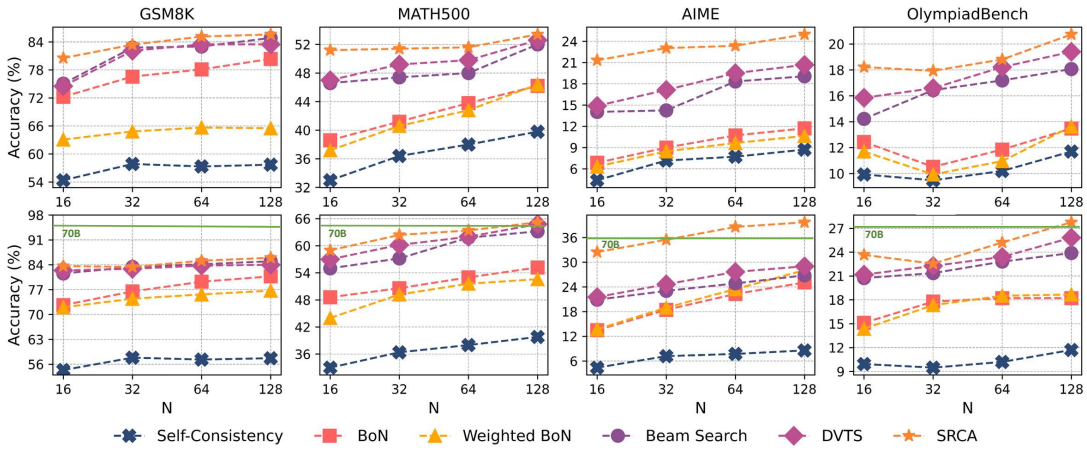

Figure 2: Performance trends of TTS methods with DeepSeek PRM (top row) and Skywork PRM (bottom row) and as the sampling number N increases from 16 to 128. In the bottom row, we additionally mark the performance of the 70B model with a green line for comparison.

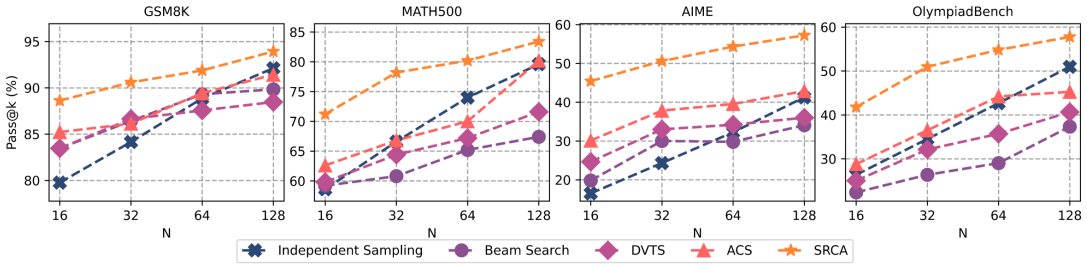

Figure 3: Pass@K trends of the 1B model with different TTS methods and DeepSeek PRM as the sampling number increases from 16 to 128. Note that for Pass@K calculation, Self-Consistency, BoN, and Weighted BoN degrade to Independent Sampling.

performing DVTS's 49.8% at $N = 64$. This advantage is more pronounced on AIME, where SRCA's accuracy at $N = 16$ exceeds all TTS methods' performance at $N = 128$. Using Skywork PRM further amplifies this gap: SRCA reaches 32.48% at $N = 16$, while the best baseline (DVTS) only achieves 29.03% at $N = 128$. Performance improvements show diminishing returns as N increases, with $N = 16 \rightarrow 32$ gains being larger than $N = 64 \rightarrow 128$. This pattern holds across different PRMs, suggesting convergence to an upper bound. Further analysis regarding the computational overhead and efficiency of SRCA is provided in Appendix B.1.

5 Analysis

5.1 Pass Rate Test: SRCA Improves Answer Discovery

The ability to sample at least one correct reasoning path is crucial for policy models, as it determines the effectiveness of PRM guidance. If a policy model does not sample any correct paths, even a perfect PRM cannot select the correct one. We conducted **Pass@k** tests on 4 datasets comparing different TTS methods, including SRCA without CCA to understand each component's contribution. Results are shown in Figure 3. SRCA demonstrates superior pass rates across datasets. Ablation studies show that CCA contributes approximately 10% improvement through answer pool expansion, while ACS alone still outperforms DVTS and Beam Search by 3% on average. Independent sampling achieves higher Pass@k on simpler datasets due to its unconstrained randomness generating more diverse solutions. However, for challenging datasets like AIME, this approach performs poorly as random exploration is ineffective when precise reasoning is required. On the other hand, the TTS method produces a candidate set of better quality, resulting in a higher pass rate.

5.2 Early Stopping: Efficient Computing with No Performance Loss.

Recent research shows that LLMs often suffer from overthinking, conducting unnecessary analysis that

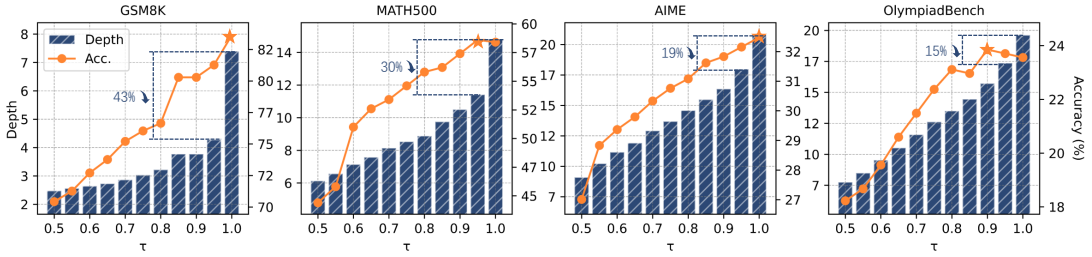

Figure 4: The average accuracy and search depth of SRCA with early stopping strategies under different values of tau. The left y-axis represents the search depth, while the right y-axis represents the accuracy (%). The dashed line in the figure annotates the reduction rate of tree depth, i.e., the number of reasoning steps, when $tau = 0.95$. The pentagon represents the best performance.

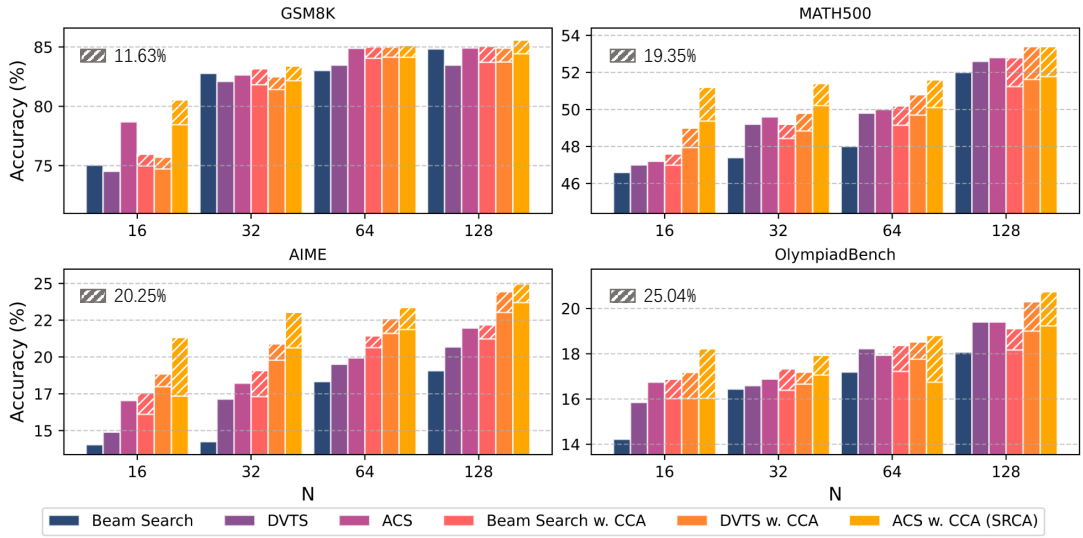

Figure 5: Ablation study results on four datasets, grouped by different values of N. For the bars corresponding to methods incorporating CCA, the Checkpoint Answer Rate (CAR) is additionally marked with slashes shading. The average CAR for each dataset is indicated in the top-left corner of each subplot.

wastes computational resources and can even lead to incorrect answers (Li et al., 2024; Wu et al., 2025; Sui et al., 2025). We implement **early stopping** in SRCA by introducing a threshold τ: reasoning stops when a checkpoint answer's score exceeds τ. We tested various τ values (0.5-1.0), measuring both accuracy and average reasoning steps, with $\tau = 1$ (no early stopping) as baseline. Results are shown in Figure 4. This experiment uses Skywork PRM with $N = 16$ samples.

Low thresholds like 0.5 hurt performance, causing a 14% accuracy drop on MATH500. While this reduces reasoning steps, the performance trade-off is unacceptable. Higher thresholds require more steps but yield better accuracy, as expected. At $\tau = 0.95$, early stopping reduces reasoning steps by 27% across datasets while only losing 0.58% accuracy. Notably, it sometimes improves accuracy: both MATH500 and OlympiadBench show better results at $\tau = 0.95$ than without early stopping. This confirms that overthinking can harm performance, due to either model reasoning limitations or PRM imperfections (Zheng et al., 2024; He et al., 2025). Early stopping is particularly effective for simpler tasks: GSM8K shows a 43% reduction in reasoning steps at $\tau = 0.95$, compared to 19% for the more challenging AIME problems.

5.3 Ablation Study

SRCA combines ACS for diverse path searching and CCA for expanding the answer candidate pool. We conducted ablation studies by: (1) testing ACS alone without CCA, and (2) combining CCA with baseline methods (Beam Search and DVTS). SRCA combines ACS for diverse path searching and CCA for expanding the answer candidate pool. We conducted ablation studies by: (1) testing ACS alone without CCA, (2) combining CCA with baseline methods (Beam Search and DVTS), and (3) replacing ACS with a random baseline.

Reasoning	Step Score	Checkpoint Answer	Final Score
Question: What is the smallest positive perfect cube that can be written as the sum of three consecutive integers?	-	-	-
Answer: ## Step 1: Let's denote the first of the three consecutive integers as n. We know that the next two consecutive integers are n+1 and n+2.	0.8774	$\boxed{9}$	0.0313
## Step 2: The sum of these three consecutive integers is n + (n+1) + (n+2). This simplifies to 3n + 3.	0.8676	$\boxed{3n + 3}$	0.0459
## Step 3: We are looking for the smallest positive perfect cube that can be expressed in this form. Let's rewrite the expression 3n + 3 as 3(n + 1).	0.8359	$\boxed{{3(1)^3 + 3(2)^3 = 3(1 + 2^3) = 3(1 + 8) = 3(9) = 27}$	0.6762
## Step 4: A perfect cube is the cube of an integer. So, we need to find the smallest value of n for which 3(n + 1) is a perfect cube.	0.7822	$\boxed{27}$	0.6837
## Step 5: To find the smallest perfect cube, we can start by checking small values of n. Let's start with n = 1: 3(1 + 1) = 3(2) = 6, which is not a perfect cube.	0.6674	$\boxed{27}$	0.7192
## Step 6: Now, let's try n = 2: 3(2 + 1) = 3(3) = 9, which is a perfect cube (3^2).	0.3876	$\boxed{9}$	0.0054
## Step 7: Since we found a perfect cube in the first step, we don't need to check further values of n.	0.0764	$\boxed{9}$	0.1356
The final answer is: $\boxed{9}$	-	-	0.0212

Table 2: A real running case of SRCA during reasoning. The first column shows the question and the model's reasoning process. The Step Score represents PRM's evaluation of the current step, while the Final Score indicates the PRM score when using the current step and its checkpoint answer as the reasoning endpoint.

5.3.1 Impact of CCA

Results are shown in Figure 5. We also track the Checkpoint Answer Rate (CAR) - the percentage of final answers selected from checkpoint, which is marked with slashes on the bar chart. After removing the CCA strategy, SRCA degrades to ACS strategy, resulting in a notable performance decline. However, on relatively simple datasets like GSM8K and MATH500, increasing the sampling size (e.g., to $N = 128$) minimizes this performance gap to less than 1%. Notably, ACS consistently outperforms both Beam Search and DVTS baselines across most configurations, demonstrating its robust effectiveness.

The integration of CCA with baseline methods yields substantial improvements in accuracy (3-4%). Analysis shows that 19.07% of final answers originate from CCA's expanded candidate pool, underscoring its important contribution to solution generation. The impact varies by problem difficulty: CAR is 11.63% for GSM8K but rises to 25.04% for OlympiadBench, indicating that CCA's influence is more pronounced in solving complex problems.

5.3.2 Effectiveness of ACS

We define this random baseline as follows: (1) For each sampling of n paths, first randomly shuffle them; (2) Then randomly select an integer k from 1 to 16, representing the number of clusters to form; (3) Randomly select k unique numbers from 0 to n as cluster boundaries. This completes the cluster division, and we continue with Round-robin selection according to Algorithm 1. For example, if $n = 16$ and $k = 3$, the boundary indices could be: 3, 5, 10. The final clusters would be divided into $[0, 1, 2]$, $[3, 4]$, $[5, 6, 7, 8, 9]$, $[10, 11, 12, 13, 14, 15]$. We report the results of this baseline in Table 3.

TTS	GSM8K	MATH500	AIME	Olym.
Beam Search	84.99	63.20	26.82	23.89
DVTS	84.00	64.80	29.03	25.82
SRCA	85.97	65.20	39.71	27.75
SRCA *Rand.*	83.78	65.00	28.62	25.96

Table 3: Results of Replacing ACS with a Random Baseline.

We set the sampling number N=128 and choose Skywork PRM. The performance of the random baseline is not significantly different from DVTS, but lower than SRCA. This indicates that clustering based on intermediate answers can bring substantial improvements.

5.4 Case Study

Table 2 shows a real running case of SRCA during reasoning. Since the complete search tree is too large, we only showcase how SRCA uses Checkpoint Answers to backup correct answers from incorrect branches. The model's reasoning process can be explained manually in three phases:

- Early Stage (Steps 1-2): During initial reasoning, the model produces either incorrect answers or incomplete expressions instead of

proper numerical values, indicating insufficient reasoning depth.

- Answer Formation Stage (Steps 3-5): Starting from Step 3, the model attempts brief reasoning in the answer box and first obtains the correct answer 27. Although reasoning in the answer box is not ideal behavior, the model successfully reaches the correct answer this way. This correct answer is maintained until Step 5.
- Error Stage (Step 6): A critical reasoning error occurs when the model incorrectly identifies 9 as a perfect cube instead of a perfect square. This error leads to an incorrect checkpoint answer that affects the reasoning until the end.

When all checkpoint answers are evaluated as reasoning endpoints by PRM, Step 5 receives the highest score of 0.7192, exceeding the natural ending's score of 0.0212. If no other branch has a score higher than 0.7192, this score will be selected as the final answer, effectively correcting the wrong answer from the natural reasoning endpoint to the correct one.

6 Conclusion

In this paper, we introduced SRCA, a novel framework that enhances LLM reasoning by introducing checkpoints between reasoning steps. Our Answer-Clustered Search strategy effectively maintains path diversity while ensuring reasoning quality, and the Checkpoint Candidate Augmentation approach efficiently utilizes intermediate predictions for final decision-making. Experimental results demonstrate that SRCA outperforms baseline methods like Beam Search and DVTS across various mathematical reasoning datasets. The success of SRCA suggests that leveraging intermediate checkpoints is a promising direction for improving LLM reasoning capabilities.

Limitations

SRCA faces two main limitations. First, while it requires checkpoints between reasoning steps, defining precise step boundaries is challenging. Although Llama-series models exhibit relatively clear step demarcations with characteristic delimiters, others, particularly the emerging "slow thinking" models, often generate outputs without distinct structural patterns and sometimes in a more conversational style. Second, the reasoning steps augmented by the CCA strategy are often incomplete. While models can still generate correct answers based on these partial reasoning paths, this incompleteness reduces the interpretability of the reasoning process. Compared to naturally completed reasoning chains, these occasionally truncated paths represent a shortcoming in terms of explanation quality and transparency.

Acknowledgments

This work is supported by Hong Kong RGC GRF No. 14206324, CUHK Knowledge Transfer Project Fund No. KPF23GWP20, and National Natural Science Foundation of China (62502310).

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *Preprint*, arXiv:2503.04697.

Edward Beeching, Lewis Tunstall, and Sasha Rush. 2024. Scaling test-time compute with open models.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. 2025. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning. *Preprint*, arXiv:2412.09078.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *Preprint*, arXiv:2407.21787.

Yanxi Chen, Xuchen Pan, Yaliang Li, Bolin Ding, and Jingren Zhou. 2025a. Simple and provable scaling laws for the test-time compute of large language models. *Preprint*, arXiv:2411.19477.

Zhipeng Chen, Yingqian Min, Beichen Zhang, Jie Chen, Jinhao Jiang, Daixuan Cheng, Wayne Xin Zhao, Zheng Liu, Xu Miao, Yang Lu, and 3 others. 2025b. An empirical study on eliciting and improving r1-like reasoning models. *Preprint*, arXiv:2503.04548.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 2 others. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Zeyu Gan, Yun Liao, and Yong Liu. 2025. Rethinking external slow-thinking: From snowball errors to probability of correct reasoning. *Preprint*, arXiv:2501.15602.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 551 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Xinyan Guan, Yanjiang Liu, Xinyu Lu, Boxi Cao, Ben He, Xianpei Han, Le Sun, Jie Lou, Bowen Yu, Yaojie Lu, and Hongyu Lin. 2024. Search, verify and feedback: Towards next generation post-training paradigm of foundation models via verifier engineering. *Preprint*, arXiv:2411.11504.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 4 others. 2024. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, Bangkok, Thailand. Association for Computational Linguistics.

Yancheng He, Shilong Li, Jiaheng Liu, Weixun Wang, Xingyuan Bu, Ge Zhang, Zhongyuan Peng, Zhaoxiang Zhang, Zhicheng Zheng, Wenbo Su, and Bo Zheng. 2025. Can large language models detect errors in long chain-of-thought reasoning? *Preprint*, arXiv:2502.19361.

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. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Coleman Hooper, Sehoon Kim, Suhong Moon, Kerem Dilmen, Monishwaran Maheswaran, Nicholas Lee, Michael W. Mahoney, Sophia Shao, Kurt Keutzer, and Amir Gholami. 2025. Ets: Efficient tree search for inference-time scaling. *Preprint*, arXiv:2502.13575.

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. 2025. Efficient test-time scaling via self-calibration. *Preprint*, arXiv:2503.00031.

Yixin Ji, Juntao Li, Hai Ye, Kaixin Wu, Kai Yao, Jia Xu, Linjian Mo, and Min Zhang. 2025. Test-time compute: from system-1 thinking to system-2 thinking. *Preprint*, arXiv:2501.02497.

Jinhao Jiang, Zhipeng Chen, Yingqian Min, Jie Chen, Xiaoxue Cheng, Jiapeng Wang, Yiru Tang, Haoxiang Sun, Jia Deng, Wayne Xin Zhao, and 5 others. 2024. Enhancing llm reasoning with reward-guided tree search. *Preprint*, arXiv:2411.11694.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333, Toronto, Canada. Association for Computational Linguistics.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. 2024. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. *Preprint*, arXiv:2401.10480.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. 2025. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling. *Preprint*, arXiv:2502.06703.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, and 2 others. 2024. Improve mathematical reasoning in language models by automated process supervision. *Preprint*, arXiv:2406.06592.

MetaAI. 2024. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models. https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/. 2025-03-01.

Kou Misaki, Yuichi Inoue, Yuki Imajuku, So Kuroki, Taishi Nakamura, and Takuya Akiba. 2025. Wider or deeper? scaling llm inference-time compute with adaptive branching tree search. *Preprint*, arXiv:2503.04412.

Skywork o1 Team. 2024. Skywork-o1 open series. https://huggingface.co/Skywork.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2024. Rewarding progress: Scaling automated process verifiers for llm reasoning. *Preprint*, arXiv:2410.08146.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and Xia Hu. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *Preprint*, arXiv:2503.16419.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback. *Preprint*, arXiv:2211.14275.

Ziyu Wan, Xidong Feng, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. 2024. Alphazero-like tree-search can guide large language model decoding and training. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Jun Wang, Meng Fang, Ziyu Wan, Muning Wen, Jiachen Zhu, Anjie Liu, Ziqin Gong, Yan Song, Lei Chen, Lionel M. Ni, and 3 others. 2024a. Openr: An open source framework for advanced reasoning with large language models. *Preprint*, arXiv:2410.09671.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, Bangkok, Thailand. Association for Computational Linguistics.

Xiyao Wang, Linfeng Song, Ye Tian, Dian Yu, Baolin Peng, Haitao Mi, Furong Huang, and Dong Yu. 2024c. Towards self-improvement of llms via mcts: Leveraging stepwise knowledge with curriculum preference learning. *Preprint*, arXiv:2410.06508.

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 *The Eleventh International Conference on Learning Representations*.

Zezhong Wang, Xingshan Zeng, Weiwen Liu, Yufei Wang, Liangyou Li, Yasheng Wang, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. 2025. Chain-of-probe: Examining the necessity and accuracy of cot step-by-step. *Preprint*, arXiv:2406.16144.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Jinyang Wu, Mingkuan Feng, Shuai Zhang, Feihu Che, Zengqi Wen, and Jianhua Tao. 2024a. Beyond examples: High-level automated reasoning paradigm in in-context learning via mcts. *Preprint*, arXiv:2411.18478.

Siwei Wu, Zhongyuan Peng, Xinrun Du, Tuney Zheng, Minghao Liu, Jialong Wu, Jiachen Ma, Yizhi Li, Jian Yang, Wangchunshu Zhou, and 7 others. 2024b. A comparative study on reasoning patterns of openai's o1 model. *Preprint*, arXiv:2410.13639.

Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka, and Yisen Wang. 2025. When more is less: Understanding chain-of-thought length in llms. *Preprint*, arXiv:2502.07266.

Zhiheng Xi, Dingwen Yang, Jixuan Huang, Jiafu Tang, Guanyu Li, Yiwen Ding, Wei He, Boyang Hong, Shihan Do, Wenyu Zhan, and 14 others. 2024. Enhancing llm reasoning via critique models with test-time and training-time supervision. *Preprint*, arXiv:2411.16579.

Violet Xiang, Charlie Snell, Kanishk Gandhi, Alon Albalak, Anikait Singh, Chase Blagden, Duy Phung, Rafael Rafailov, Nathan Lile, Dakota Mahan, and 4 others. 2025. Towards system 2 reasoning in llms: Learning how to think with meta chain-of-thought. *Preprint*, arXiv:2501.04682.

Wei Xiong, Hanning Zhang, Nan Jiang, and Tong Zhang. 2024. An implementation of generative prm. `https://github.com/RLHFlow/RLHF-Reward-Modeling`.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 50 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Xiao Yu, Baolin Peng, Vineeth Vajipey, Hao Cheng, Michel Galley, Jianfeng Gao, and Zhou Yu. 2024. Improving autonomous ai agents with reflective tree search and self-learning. In *The Thirteenth International Conference on Learning Representations*.

Weihao Zeng, Yuzhen Huang, Lulu Zhao, Yijun Wang, Zifei Shan, and Junxian He. 2025. B-star: Monitoring and balancing exploration and exploitation in self-taught reasoners. *Preprint*, arXiv:2412.17256.

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

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2025a. Generative verifiers: Reward modeling as next-token prediction. *Preprint*, arXiv:2408.15240.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Zhihan Guo, Yufei Wang, Irwin King, Xue Liu, and Chen Ma. 2025b. What, how, where, and how well? a survey on test-time scaling in large language models. *Preprint*, arXiv:2503.24235.

Yu Zhao, Huifeng Yin, Bo Zeng, Hao Wang, Tianqi Shi, Chenyang Lyu, Longyue Wang, Weihua Luo, and Kaifu Zhang. 2024. Marco-o1: Towards open reasoning models for open-ended solutions. *Preprint*, arXiv:2411.14405.

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. *Preprint*, arXiv:2412.06559.

A Experiment Settings

A.1 Datasets

The following are the four datasets used in the experiment:

- **GSM8K** (Cobbe et al., 2021) is an evaluation set consisting of 8,500 high-quality primary school mathematics problems. It is mainly used to assess the language comprehension and mathematical reasoning abilities of models in basic mathematical problems.
- **MATH500** (Lightman et al., 2024) is a subset of the MATH dataset (Hendrycks et al., 2021) containing 500 questions. It covers seven mathematical domains and five difficulty levels. It is designed to test the performance of LLMs in solving advanced mathematical problems.
- **AIME**[1] offers a rich collection of challenging problems from the American Invitational Mathematics Examination and contains 933 high-difficulty mathematical problems.
- **OlympiadBench** is an Olympiad-level bilingual multimodal scientific benchmark (He et al., 2024). In this experiment, only the subset of English mathematical problems without images was tested, with a total of 674 questions.

A.2 Baselines

- Greedy Search: A decoding method based on the principle of local optimality. It always selects the token with the highest current probability as the output.
- Self-Consistency (Wang et al., 2023): LLMs generates N independent reasoning paths. Finally, the most frequently occurring output is counted as the answer.
- Best-of-N (BoN) (Brown et al., 2024): Similar to self-consistency, the LLM generates N independent reasoning paths. According to the scores given by the reward model, the path with the highest score is selected as the answer.
- Weighted BoN (Brown et al., 2024): It is a combination of Self-Consistency and BoN. The reward model scores the N independent reasoning paths generated by the LLM. Then, the paths are clustered according to the answers, and the sum of the path scores within each cluster is taken as the answer's score. The answer with the highest score is selected.
- Beam Search (Snell et al., 2024): N reasoning paths are maintained at each reasoning step. According to the scores given by the PRM for the current paths, M paths are selected to continue the reasoning and expand downward. Each selected path can expand into N/M subpaths.
- Diverse Verifier Tree Search (DVTS) (Beeching et al., 2024): DVTS is an extension of the beam search. It first initializes M subtrees. Each subtree samples N/M paths at every step. Those paths are then scored by the PRM. The path with the highest score within the subtree is selected for further reasoning. It is similar to Beam Search, as in each step, M paths are selected from N paths for further reasoning. Due to the scope limitation of the subtrees, it prevents some locally optimal branches from early elimination, thereby enhancing the path diversity.

A.3 Testing Prompt

```
Solve the following math problem efficiently and clearly:

- For simple problems (2 steps or fewer):
Provide a concise solution with minimal explanation.

- For complex problems (3 steps or more):
Use this step-by-step format:

## Step 1: [Concise description]
[Brief explanation and calculations]

## Step 2: [Concise description]
[Brief explanation and calculations]

...

Regardless of the approach, always conclude with:

Therefore, the final answer is: $\boxed{answer}$. I hope it is correct.

Where [answer] is just the final number or expression that solves the problem.
```

Table 4: System prompt used in the experiment.

A.4 Details of PRM Score

The calculation of the PRM score strictly follows the official guidelines provided by the PRM developers. Specifically:

[1] www.kaggle.com/datasets/aime-problem-set-1983-2024

Model	Type	N	FLOPs
Llama-3.2-1B-Instruct	Auto Regressive	128	1.31×10^{18}
Llama3.1-8B-PRM-Deepseek-Data	Reward	128	9.03×10^{15}
Llama-3.1-70B-Instruc	Auto Regressive	1	3.04×10^{18}

Table 5: Computational Cost Analysis (in FLOPs) for Different Model Configurations during Inference.

- For DeepSeek PRM, following its official usage guidelines, we organize the existing reasoning process into a dialogue format, input it into the PRM, and use the probability of outputting "+" as the PRM score.

- For Skywork PRM, it adds and trains a value head on top of a conventional LLM. This head outputs a 0-1 score for each token. We select the score corresponding to the last token output in a step as the PRM score.

B Supplementary Experimental Results

B.1 Computational Cost Analysis in FLOPs

Table 5 compares the computational cost in FLOPs for processing a single query across different models. We assume an input length of 256 tokens (prefill) and an output length of 4096 tokens (decode). The policy model generates tokens sequentially in an auto-regressive manner, requiring multiple forward passes, while the PRM requires only one forward pass for scoring.

The results demonstrate that SRCA with $N = 128$, combining a 1B policy model and an 8B PRM, requires only 43.01% of the computational cost compared to the 70B model for processing a single sample. Considering the experimental results reported in Table 1, the 1B model enhanced with SRCA achieves higher accuracy than the 70B model, indicating that our approach not only reduces computational overhead but also yields superior performance.

B.2 Evaluation of Scoring Methods and Selection Strategies

We analyzed how different scoring methods for reasoning paths and answer selection strategies affect the accuracy of TTS methods. We employ `Llama3.1-8B-PRM-Deepseek-Data` as the PRM in this experiment. The PRM assigns scores to each reasoning step, generating a sequence of scores for each path. We examined two primary methods for computing the final path score:

- **Last**: Using the final step's score as the path score, where PRM functions similarly to an Outcome Reward Model (ORM).
- **Mean**: Taking the average of the score sequence to reflect the overall reliability of the reasoning process.

Furthermore, BoN and Weighted BoN can be combined with other tree search algorithms as answer selection strategies. Specifically, after the tree search algorithm generates multiple candidate paths:

- **BoN**: Selects the path with the highest score
- **Weighted BoN**: First clusters answers, then selects the answer with the highest sum of path scores within its cluster

The experimental results (Table 6) demonstrate that the Last scoring method consistently outperforms Mean, while BoN generally yields better results than Weighted BoN. This pattern holds across all four datasets and three TTS methods.

Notably, the superiority of Last over Mean suggests that some correct reasoning paths have high final scores but lower average scores. This indicates that even when reaching the correct answer, the intermediate reasoning steps may not be entirely accurate. Developing TTS methods that ensure both process and outcome accuracy remains a future research direction.

The choice between BoN and Weighted BoN reflects a balance between policy model and reward model decision-making. BoN relies primarily on PRM's judgment by selecting the highest-scoring path, while Weighted BoN considers the sampling frequency of the policy model through score aggregation. In our experiments, using a 1B parameter policy model and an 8B parameter PRM, the PRM-dominated BoN strategy achieved superior results, likely due to the PRM's stronger discriminative ability.

B.3 Experiments on Qwen3-0.6B

To validate the generalizability of SRCA, we conducted additional experiments on Qwen3-0.6B (Yang et al., 2025), following the same set-

Selection	N	Method	GSM8k	MATH500	AIME	Olympiad	Avg.	GSM8k	MATH500	AIME	Olympiad	Avg.
			Last					Mean				
BoN	16	Beam	0.7505	0.4660	0.1404	0.1422	0.3748	0.7475	0.4600	0.1200	0.1247	0.3631
		DVTS	0.7452	0.4700	0.1489	0.1585	0.3807	0.7331	0.4620	0.1307	0.1525	0.3696
		SRCA	0.8054	0.5120	0.2133	0.1822	0.4282	0.7869	0.4940	0.1747	0.1718	0.4069
		Avg.	0.7670	0.4827	0.1675	0.1610	0.3946	0.7558	0.4720	0.1418	0.1497	0.3798
	32	Beam	0.8278	0.4740	0.1425	0.1644	0.4022	0.8043	0.4780	0.1457	0.1377	0.3914
		DVTS	0.8210	0.4920	0.1714	0.1659	0.4126	0.8241	0.5020	0.1758	0.1629	0.4162
		SRCA	0.8340	0.5140	0.2304	0.1793	0.4394	0.8317	0.5120	0.1908	0.1793	0.4285
		Avg.	0.8276	0.4933	0.1814	0.1699	0.4181	0.8200	0.4973	0.1708	0.1600	0.4120
	64	Beam	0.8302	0.4800	0.1833	0.1719	0.4164	0.8392	0.4960	0.1758	0.1659	0.4192
		DVTS	0.8347	0.4980	0.1951	0.1822	0.4275	0.8484	0.5140	0.1907	0.1733	0.4316
		SRCA	0.8514	0.5160	0.2337	0.1881	0.4473	0.8491	0.5160	0.2144	0.1837	0.4408
		Avg.	0.8388	0.4980	0.2040	0.1807	0.4304	0.8456	0.5087	0.1936	0.1743	0.4305
	128	Beam	0.8484	0.5200	0.1907	0.1807	0.4350	0.8340	0.5160	0.1832	0.1807	0.4285
		DVTS	0.8347	0.5260	0.2068	0.1940	0.4404	0.8499	0.5180	0.1843	0.1866	0.4347
		SRCA	0.8560	0.5340	0.2497	0.2074	0.4618	0.8514	0.5240	0.2197	0.1896	0.4462
		Avg.	0.8464	0.5267	0.2157	0.1940	0.4457	0.8451	0.5193	0.1957	0.1856	0.4365
Weighted BoN	16	Beam	0.7369	0.4600	0.1200	0.1303	0.3618	0.7194	0.4460	0.1189	0.1229	0.3518
		DVTS	0.7422	0.4760	0.1446	0.1526	0.3789	0.7111	0.4620	0.1125	0.1496	0.3588
		SRCA	0.7597	0.4800	0.1714	0.1688	0.3950	0.7187	0.4680	0.1393	0.1674	0.3734
		Avg.	0.7463	0.4720	0.1453	0.1506	0.3785	0.7164	0.4587	0.1236	0.1466	0.3613
	32	Beam	0.7740	0.4760	0.1446	0.1659	0.3901	0.7520	0.4480	0.1404	0.1348	0.3688
		DVTS	0.7877	0.4780	0.1661	0.1718	0.4009	0.7491	0.4680	0.1425	0.1644	0.3810
		SRCA	0.7937	0.4900	0.1822	0.1778	0.4109	0.7832	0.4700	0.1704	0.1762	0.4000
		Avg.	0.7851	0.4813	0.1643	0.1718	0.4007	0.7614	0.4620	0.1511	0.1585	0.3833
	64	Beam	0.8036	0.4780	0.1886	0.1733	0.4109	0.7771	0.4860	0.1939	0.1615	0.4046
		DVTS	0.7915	0.4860	0.1897	0.1825	0.4124	0.7839	0.4900	0.1961	0.1719	0.4105
		SRCA	0.8173	0.5060	0.2068	0.1854	0.4289	0.7945	0.4940	0.2208	0.1778	0.4218
		Avg.	0.8041	0.4900	0.1950	0.1804	0.4174	0.7852	0.4900	0.2036	0.1704	0.4123
	128	Beam	0.8014	0.5000	0.1907	0.1854	0.4194	0.7574	0.4820	0.1951	0.1911	0.4064
		DVTS	0.8195	0.5020	0.2079	0.1899	0.4298	0.7680	0.4920	0.1994	0.1940	0.4134
		SRCA	0.8173	0.5100	0.2262	0.1943	0.4370	0.7786	0.5120	0.2444	0.2030	0.4345
		Avg.	0.8127	0.5040	0.2083	0.1899	0.4287	0.7680	0.4953	0.2130	0.1960	0.4181

Table 6: Performance comparison of TTS methods with different scoring methods (Last/Mean) and selection strategies (BoN/Weighted BoN) on four benchmark datasets. Numbers indicate accuracy. Higher scores indicate better performance. Red cells denote group averages for each N value.

tings as our main experiments. We set the sampling number N to 16 to expedite the experimental process. The experimental results are presented in Table 7. Experimental results indicate that SRCA maintains superior performance compared to other TTS approaches. The observed trends align with the findings from our primary experiments, thereby confirming the general applicability of SRCA across different LLMs.

Models & TTS	GSM8K	MATH500	AIME	Olympiad
Greedy Search	42.61%	34.40%	3.54%	13.63%
Self-Consistency	52.62%	47.00%	4.07%	20.59%
BoN	68.69%	51.20%	6.54%	23.56%
Weighted BoN	63.91%	53.60%	7.29%	23.41%
Beam Search	72.10%	54.00%	16.4%	25.07%
DVTS	74.91%	55.80%	17.36%	25.67%
SRCA	**79.45%**	**56.60%**	**21.33%**	**27.89%**

Table 7: Comparative results of TTS with Qwen3-0.6B. Numerical values indicate accuracy rates, with bold figures denoting the best performance. Experimental parameters: N=16, utilizing DeepSeek PRM.