Tracing the Light of Thought: A Probabilistic Self- and Cross-Consistency Verification Mechanism Improving Mathematical Reasoning in LLMs

Xiaoyang Liu, Dawei Wang, Tian Li, Huizhi Liang, Gary Ushaw, Richard Davison

Newcastle University, United Kingdom

Abstract

Reasoning capability is fundamental in enabling Large Language Models to perform complex multi-step inference. By sampling multiple reasoning paths and selecting the most frequent answer, Self Consistency (SC) remains highly effective but fails on challenging tasks where incorrect answers dominate the majority. Inspired by Metropolis Light Transport in physically-based rendering, where discovered high-contribution light paths guide subsequent sampling toward illumination sources, we propose *Metropolis Self Consistency* and its multi-LLM extension, *Metropolis Cross Consistency*, a probabilistic self- and cross-consistency verification framework for mathematical reasoning. Our approach employs an accept-reject mechanism to encourage high-quality reasoning paths, concentrating sampling in regions more likely to yield correct answers. Experiments on 9 LLMs across 4 challenging mathematical benchmarks demonstrate consistent improvements over SC. Even when combining models of vastly different capabilities, MCC maintains performance virtually matching the most capable model while significantly reducing computational cost compared to SC with the strongest model alone. While our implementation is training-free, adds minimal token overhead beyond SC, and requires no external reward model, our approach provides a flexible paradigm that can accommodate any scalar reward representing path correctness.

1 Introduction

> "We do not have government by the majority, we have government by the majority who participate."
>
> — Thomas Jefferson

Through generating a single detailed reasoning trace (Wei et al., 2022; OpenAI, 2024b), recent large language models (LLMs) have developed impressive reasoning capability that enables coherent multi-step inference beyond simple pattern recognition. Self Consistency (SC) further enhances LLMs' performance by sampling multiple reasoning paths and selecting the most frequent answer as the final output (Wang et al., 2023). Despite being an earlier and simpler method, SC continues to outperform or match more recent complex reasoning strategies, e.g., Multi-agent Debate (Liang et al., 2024), Tree of Thought (Yao et al., 2023), when given comparable or lower computational budgets (Wang et al., 2024).

However, for problems requiring complex multi-step deductions (such as challenging mathematical problems), where LLMs have a low chance of generating correct answers, the likelihood of the correct answer appearing most frequently becomes vanishingly small. This suggests that SC, while effective for problems within the model's capability range, degrades drastically for tasks that exceed it.

An analogous problem is observed in physically based rendering (PBR), where images are rendered by tracing light transport paths to estimate global illumination (Veach and Guibas, 1997): consider you are sitting in a dimly lit room where the only light source is outside, with light entering solely through a narrow gap under the door. If we randomly trace light rays from the observer's position, the vast majority of sampled light paths will contribute minimal illumination, as most rays fail to reach the light source. This mirrors the challenge faced when applying SC to difficult problems with LLMs: just as most traced rays miss the crucial light source, most generated solutions miss the correct answer. Both domains face the same fundamental issue: when the target (whether a light source or correct answer) occupies a small fraction of the sample space, random sampling becomes highly inefficient at capturing the desired outcome.

Crucially, computer graphics has developed an elegant solution to this problem: Metropolis Light Transport (MLT). The core insight of MLT is that when a high-contribution light path is discovered (e.g., when a traced ray successfully finds the gap

Findings of the Association for Computational Linguistics: ACL 2026, pages 22444–22459

under the door), subsequent samples should be concentrated in the neighborhood of this valuable path. Rather than continuing with random sampling across the entire space as SC does, MLT dramatically increases the probability of finding additional high-contribution paths by exploiting mutations of successful paths, effectively turning a rare discovery into a rich source of related solutions.

This observation leads to a natural analogy: if we map the brightness of a light path to the correctness of a reasoning path, we can see that, just as ray tracing aims to discover high-energy paths that successfully connect the observer to light sources, LLM reasoning seeks to find valid solution paths that lead to correct answers. This correspondence suggests that techniques successful in one domain may be transferable to the other. We therefore introduce **Metropolis Self Consistency (MSC)**, a MLT-inspired sampling scheme for mathematical reasoning. In essence, MSC is a sequential SC augmented with two components: for each iteration, (1) a simple accept/reject mechanism is used to filter out sampled reasoning path with low correctness; (2) previously accepted path is mutated, with controlled probability, to explore potentially better reasoning path in their vicinity. Through this, the sampling process efficiently discovers and exploits clusters of high-correctness reasoning paths, even when such paths constitute a small fraction of the overall solution space.

Recent evidence (YAO et al., 2024) has demonstrated that in multi-step reasoning paths generated by LLMs, low logprobabilities of the initial tokens within a step are notably correlated with the presence of errors in that step. Motivated by this observation, we offer a novel lens through which to interpret error detection in multi-step LLM reasoning: as we have viewed reasoning paths as light paths, each reasoning step can be analogous to a reflection event in the scene. When the initial tokens of a step exhibit low probabilities, the path encounters numerous possible reasoning branches at that juncture, potentially forking to divergent conclusions. This high uncertainty at a given step drastically reduces the probability of sampling the correct reasoning path. We can conceptualize this instability arising from the model uncertainty as analogous to the reflection on a diffuse surface, where incident energy scatters across multiple directions, diluting the contribution along any single path. Conversely, when the model demonstrates high confidence in correct reasoning steps (evidenced by high logprobabilities for the initial tokens), this resembles specular reflection, which preserves the energy along a well-defined direction and stably guides the reasoning toward a determinate conclusion.

Guided by this insight, the correctness of LLM reasoning paths is quantified by a lightweight correctness function that requires only the logprobabilities of the generated tokens as input. Importantly, this function provides a probabilistic measure indicating which solutions are more likely to be correct, rather than definitively identifying correct answers: instead of simply selecting the answer with the highest correctness score (which may still be incorrect), we concentrate our sampling efforts around regions of high correctness values. This approach acknowledges the uncertainty inherent in the correctness estimation while leveraging it to guide more effective exploration of the solution space.

We also extend MSC into a multi-LLM workflow called **Metropolis Cross Consistency (MCC)**. By orchestrating language models of comparable capabilities trained on different knowledge corpora to cross-verify each other's reasoning paths, MCC further improves performance over MSC. Remarkably, we find that even when MCC incorporates models with significant capability disparities, its performance virtually matches that of using only the strongest model.

Our contributions can be summarized as follows: **(1)** We propose MSC, a novel sampling scheme that bridges ray tracing and LLM reasoning, consistently improving accuracy over SC on challenging mathematical tasks. **(2)** We extend MSC into a multi-LLM workflow called MCC, which either further improves mathematical reasoning performance or virtually maintains the SC performance of the stronger model at reduced cost. **(3)** We provide a novel angle to see LLM error analysis as material reflectance analysis. **(4)** While our implementation provides a zero-overhead correctness function for mathematical reasoning, it is designed to be hot-swappable with any mechanism that produces a scalar proportional to correctness.

2 Methodology

We refer to an LLM that samples reasoning paths as a *generator*, and the sequential sampling process of a generator as a *chain*. Our framework supports N generators (and thus, N parallel chains) for $N \in \{1, 2\}$, which we call MSC ($N = 1$) and MCC ($N = 2$). When $N = 2$, a *judging* module is

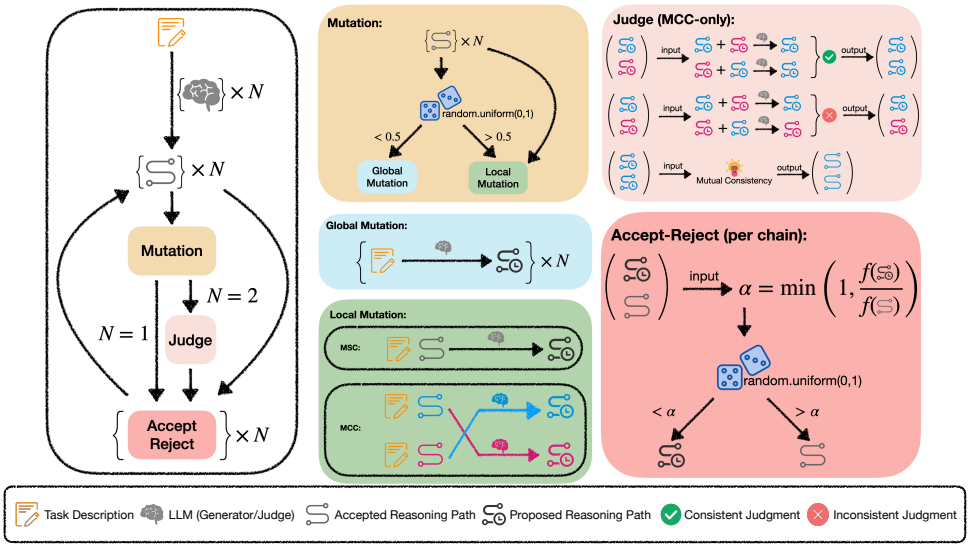

Figure 1: General workflow of MSC/MCC (left), with each of its components detailed on the right.

involved, where a third LLM is used as a judge. As shown in the leftmost box of Figure 1, given a task, the N generators first generate N initial reasoning paths (Section 2.1). Then, our framework iteratively undergoes Mutation (Section 2.2), an optional Judging step that is inserted only for $N = 2$ (Section 2.3), and Accept-Reject (Section 2.4), producing a distribution of predefined sample size. The final answer is selected by majority voting on the answers of the sampled paths.

2.1 Initialization

Our framework conducts N chains in parallel, each powered by a distinct generator. To start, those generators are prompted with the task description to produce a total of N reasoning paths.

2.2 Mutation

Inspired by classic implementations of MLT (Kelemen et al., 2002; Veach and Guibas, 1997), we introduce two complementary mutation strategies: local and global mutation. At the beginning of each iteration, a single random number $u \sim \text{Uniform}(0, 1)$, served as a probability check shared across all N chains, determines whether to apply local or global mutation, with the probability of choosing global controlled by a predefined hyperparameter called *large step probability* p_{large}. All N generators then execute the same chosen mutation type synchronously, producing N new proposed paths (one per chain). In global mutation, each generator is prompted with only the task description to propose a fresh solution. In local mutation, each generator receives the task description along with a previously accepted solution, and is asked to propose a refined solution. For $N = 2$ (MCC), we couple the two chains during local mutation: each generator refines the other chain's previously accepted solution, rather than its own. As LLMs may have limited ability to evaluate their own response quality (Huang et al., 2024a), this cross-refinement mechanism naturally leverages cross-model knowledge to mitigate bias in self-refinement. The mixed global/local mutation scheme is loosely reminiscent of the system 1/2 dual-process theories of cognition (Wason and Evans, 1974; Kahneman, 2011), which we note as an inspirational remark.

2.3 Judge (MCC-only)

The judging step is bypassed entirely when $N = 1$ (MSC); the rest of this section therefore describes the $N = 2$ (MCC) case: after mutation, the two proposals from the two chains are passed to an additional judging module. A third LLM acting as the judge is given the task description together with the two proposed paths and asked to determine which one better solves the task. To mitigate the positional bias inherent in LLM-as-a-Judge, we run this pairwise comparison twice in parallel, each time with the two proposals in swapped order, producing two pairwise decisions. If the two pairwise decisions agree, we replace the losing proposal with the winning one; both chains then temporarily hold the same favored proposal as they enter the Accept-Reject phase (Section 2.4). If the two decisions disagree, the judge cannot reliably distinguish the two proposals; it takes no action, and

each chain retains its own proposal. Importantly, we do not treat the judge as an oracle: were the framework to unconditionally accept the judge's preference as final, MCC's overall performance would be rigidly bottlenecked by the judge's individual capability. Instead, each chain still runs Accept-Reject independently against its own previously accepted solution: the judge's preference merely increases the probability that the favored proposal will be sampled into the final distribution, but does not guarantee absolute acceptance. This explains why, as we show in Section 3.1 (Table 1), MCC maintains top-tier performance even when paired with a weaker and cheaper judge. Nevertheless, when both proposed paths independently yield the exact same final answer, we reach *mutual consistency* and directly accept both into the final sample distribution, bypassing both the judging and the Accept-Reject phases, as agreement between distinct LLMs is a strong indicator of correctness (Qi et al., 2025).

2.4 Accept-Reject

As with MLT, our Accept-Reject mechanism is grounded in a well-established principle called the Metropolis-Hastings (MH) algorithm (Metropolis et al., 1953; Hastings, 1970). The theoretical rationale is as follows.

SC can be seen as sampling multiple solution sequences $\{x\}$ for a given question q with respect to the distribution $p(x \mid q) = \prod_{t \in x} p(t)$, where $p(x \mid q)$ denotes the conditional probability of generating the solution sequence x given the task description q, and $p(t)$ denotes the probability assigned to the generated token t in the solution sequence x (for brevity, we omit writing the condition on the preceding context). Given a hard task q, the probability mass of $p(x \mid q)$ for incorrect x often far outweighs correct ones, causing SC to fail. Hence, we aim to sample x directly with respect to their correctness, turning correct answers into the majority. This is achieved by the MH algorithm: it iteratively generates a sequence of samples that distribute towards a target distribution f that is difficult to sample from directly. Specifically, given a sample x that was accepted during iteration t_{i-1}, a sample y is proposed, through a proposal distribution Q, which is then accepted probabilistically according to the acceptance rate in iteration t_i:

$$a(y \mid x) = \min\left(1, \frac{f(y)Q(x \mid y)}{f(x)Q(y \mid x)}\right). \quad (1)$$

In our context, x and y are solution sequences generated by LLM. f is our correctness function. $Q(\cdot \mid \cdot)$ denotes the probability of proposing a new solution sequence at iteration t_i, conditional on the solution sequence accepted at iteration t_{i-1}. It is often referred to as *transition probability*. We find that, when grounded in LLM, the performance peaks with the assumption $Q(x|y) = Q(y|x)$ (see Section 3.3). This assumption not only provides us with a conceptually simple accept/reject mechanism, but eliminates the significant overhead for computing the bidirectional transition probabilities. Our acceptance rate thus becomes:

$$\alpha(y \mid x) = \min\left(1, \frac{f(y)}{f(x)}\right). \quad (2)$$

That is, a proposal is accepted if it is at least as good as the previously accepted solution; otherwise, the worse the quality of the proposed solution, the more likely the previous solution is to be retained.

While MH algorithm provides a principled pillar to our work, our work is neither directly motivated by nor solely based on it. As shown in Section 3.4, directly applying MH algorithm (i.e. Equation 1) to a sequential SC yields catastrophic results. For MH to take effect, the MLT-inspired mutation framework described in Section 2.2 is crucial.

2.5 The Correctness Function

For a multi-step solution sequence x, our correctness function f combines confidence measures at multiple granularities and is expressed as:

$$f(x) = e^{\frac{E(x)+D(x)+P(x)}{T}} \quad (3)$$

We show later in Section 3.2 that such a mixing drastically increases the stability of our algorithms.

Step-level Confidence. As discussed in Section 1, erroneous reasoning steps are characterized by low initial-token probabilities. While YAO et al. (2024) focus on the first 3 tokens for error detection, our preliminary experiments show that low probabilities often persist across the first 3 to 10 tokens in error steps (see Appendix B). Motivated by those observations, for a step s containing l tokens in x, we define the step correctness $C(s) = \frac{\sum_{i=1}^{l} w_i p(t_i)}{\sum_{i=1}^{l} w_i}$, a weighted average probability of the l tokens, using an exponential decay weighting $w_i = \left(\frac{1}{2}\right)^{\frac{i}{\lambda}}$ with half-life $\lambda = 3$. This value ensures that the weighting concentrates on approximately the first 10 tokens, where error signals are predominantly

observed (Figure 4), while subsequent tokens contribute negligibly. This design accounts for the uncertainty for the whole step while shifting the attention towards the initial tokens. We intentionally avoid uniformity measures, e.g., KL divergence used by YAO et al. (2024), as we observed that tokens in many correct steps exhibit large probability fluctuations (see Appendix B). Step-level confidence is then defined as the minimum correctness among all steps $E(x) = \min(\{C(s) \mid s \in x\})$, a measure of the most error-prone step within x.

Token-level Confidence. Although most error steps assign low probability to their initial tokens, we observed that key information (e.g., intermediate result values) often appears later in the step. As a result, focusing only on initial tokens may fail to capture sudden hallucinations that could alter the final answer. To address this issue, we measure token-level confidence spikes by computing $G(x) = \min(\{\log p(t_i) - \log p(t_{i-1}) \mid t \in x\})$, the largest drop in log probabilities between any two successive tokens t_{i-1}, t_i in sequence x. We normalize $G(x)$ through a sigmoid function $D(x) = \frac{1}{1+e^{-kG(x)}}$, where we set $k = \frac{ln19}{16}$ so that D maps to $(0, 0.05]$ when $G \leq -16$, a well-established empirical threshold for error detection (Yin et al., 2024).

Path-level Confidence. To account for cases where LLMs may occasionally self-correct errors from early steps in its later steps, we include a measure for global confidence to offset the bias in the correctness score introduced by local errors. This path-level confidence is defined as the geometric mean of the reciprocals of perplexity for all steps $P(x) = \exp\left(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{|s_i|}\sum_{j=1}^{|s_i|}\log p\left(t_j^{(i)}\right)\right)$ where x is a reasoning path that has n steps, each denoted as s.

Proposal Tolerance. The positive hyperparameter T controls the "tolerance" on the sampled reasoning path: as $T \to 0$, a new path with lower confidence than the previously accepted path will be increasingly likely to be rejected.

Calibration Considerations. LLMs' token probabilities are known to not being very well calibrated. Thus, our correctness function design tries to avoid absolute probability thresholds. Instead, it relies on relative drops between consecutive tokens and relative decay weightings.

3 Experiments

Models. Our work is applicable to any LLM whose log probabilities are accessible. In this paper, a total of 9 LLMs ranging from open-source to proprietary are used. We divide them into 2 groups by the generators' sizes: for *small-scale* group, Mistral Small 3 24B (Mistral team, 2025) and Gemma 2 27B (Team et al., 2024) are used as generators; for *large-scale* group, generators are chosen to be GPT 4.1 (OpenAI, 2025a) and Virtuoso Large (Arcee AI, 2025). To isolate the performance improvement attributable to the judge, each group includes both a strong and a weak judge: GPT 4o (OpenAI, 2024) and GPT 4o mini (OpenAI, 2024) for small-scale group; Gemini 2.0 Flash, Gemini 2.0 Flash-Lite (Google, 2024) for large-scale group. GPT-OSS-20B (OpenAI, 2025b), a popular reasoning model, is also used to evaluate MSC. All inferences were performed via proprietary APIs or the Together Inference Endpoint (Together AI, 2025).

Datasets. We evaluate on 4 datasets consisting of diverse mathematical tasks: GSM-hard (Gao et al., 2023), AQuA (Ling et al., 2017), MATH 500 (Lightman et al., 2024), AIME (AI-MO, 2024). These datasets were deliberately chosen from a wide range of reasoning-intensive benchmarks for their great difficulty for both LLMs and humans.

Baselines and Metrics. We use Self Consistency (SC) as the main baseline to compare with MSC/MCC for two reasons: (1) Consistency-based method with reasonably large sample size delivers stable performance across different program runs; (2) SC **outperforms/matches** recent complex approaches (e.g., Multi-agent Debate, Tree of Thought) when given **comparable/lower** computational budgets (Wang et al., 2024). Due to their tree- and graph-based structures, methods like Tree of Thought and Graph of Thought demand exponentially higher token budgets. Comparing our lightweight, linear-chain framework against such heavily-budgeted methods would introduce massive computational disparity as a confounding variable, resulting in an unfair evaluation. Thus, SC represents the strongest budget-equivalent baseline in the literature. We also include CoT with complex exemplars (Fu et al., 2023), as it is widely adopted for its efficiency and effectiveness on reasoning tasks. We choose to report all the results using absolute accuracy over ten independent runs ($\frac{\sum \text{\# Correct}}{10\times|\text{Benchmark}|}$). Both SC, MSC and MCC use

Model	Method	GSM-hard	AQuA	MATH 500	AIME	Avg.
Mistral Small 3 24B	CoT@6-shot	67.37	78.66	67.16	4.67	54.46
	SC@maj32	68.09	75.12	66.04	2.89	53.04
	MSC	**71.08**	77.05	67.38	6.89	55.60
Gemma 2 27B	CoT@6-shot	64.82	68.86	55.28	4.33	48.32
	SC@maj32	65.91	74.45	62.04	5.56	51.99
	MSC	68.23	76.81	66.04	**7.44**	54.63
Mistral Small 3 24B + Gemma 2 27B	MCC (Judge: GPT 4o)	69.08	79.61	**68.32**	5.22	55.56
	MCC (Judge: GPT 4o mini)	69.76	**80.35**	67.08	5.78	**55.74**
GPT 4.1	CoT@6-shot	74.18	87.40	86.70	30.22	69.63
	SC@maj32	75.34	88.31	91.74	44.89	75.07
	MSC	75.90	88.94	92.82	**48.00**	**76.41**
Virtuoso Large	CoT@6-shot	64.84	83.19	81.64	13.67	60.84
	SC@maj32	68.95	87.56	87.44	17.11	65.27
	MSC	71.79	86.93	86.36	20.44	66.38
GPT 4.1 + Virtuoso Large	MCC (Judge: Gemini 2.0 Flash)	**76.38**	88.31	**93.04**	35.67	73.35
	MCC (Judge: Gemini 2.0 Flash-Lite)	74.43	**89.13**	91.14	41.33	74.01

Table 1: Main results. The best performance for each dataset is highlighted in **bold** for small-scale (upper-half) and large-scale (lower-half) LLMs, respectively.

the same sample size of 32. We used few-shot prompting with 6 exemplars for CoT pass@1, as in-context learning offers little benefit beyond this.

For MCC, we deliberately compare against the standalone SC of the stronger model in each pair, which provides a more rigorous upper bound than a simple ensemble of both models' outputs: when generators possess different capacities, the weaker model introduces noise to the standard majority voting on combined outputs, and inherently dilutes the stronger model's correct answers, making a simple ensemble a predictably weaker baseline.

Implementation Details. For CoT, greedy decoding is used, aligning with its original work (Wei et al., 2022). For all samplings in SC, MSC and MCC, decoding temperature is set to 1, top-p to 1, and top-k to the LLM's vocabulary size, preserving the original Softmax probability distribution. All inferences of the chosen open-source models were performed in FP16 precision, which helps preserve the original relative probability differences, mitigating calibration distortion. Judges use greedy decoding for pairwise comparisons. In our main experiments, the large step probability is set to 0.5, aligning with the optimal value found in classic implementation of MLT (Kelemen et al., 2002) which balances global and local search. The proposal tolerance T is set to 0.7; preliminary experiments confirmed that varying T (e.g., to 0.5) yields no noticeable difference, suggesting low sensitivity to this parameter as long as the resulting correctness scores remain sufficiently discriminative across reasoning paths. We deliberately select literature-established values or vanilla settings for each hyperparameter (including equal weighting of E, D, and P in Equation 3) without any model- or task-specific tuning.

Prompt Consistency. We use similar prompts for all LLMs across all benchmarks, the only major difference is that the exemplars were created to respect the specific type of tasks in each dataset. Our prompt templates are provided in Appendix C.

3.1 Main Results

Table 1 presents results of our main experiments. Generators in small-scale group were selected to demonstrate similar performance across most tasks, while generators in large-scale group exhibit substantial gap in their general capabilities. This setup enables us to examine how differences in paired generator capabilities affect MCC's behavior.

Begin with small-scale group, significant improvements by MSC are consistently observed for Gemma. For Mistral, while MSC's improvements over SC are relatively moderate on AQuA and MATH 500, the improvements on GSM-Hard and AIME are remarkable: MSC outperforms SC by 238% on AIME. We find that, for some tasks where all the 32 samples obtained by SC are incorrect, MSC can find the correct solutions and successfully pick the right ones during majority voting. Note that, according to its official documentation, Mistral performs better with low decoding temperature (Mistral AI, 2025). Thus, some of the Mistral's results produced by CoT (greedy decoding) outperformed those produced by SC (decoding tempera-

ture = 1). MCC further achieves substantial gains thanks to its cross-consistency architecture. Its performance on AQuA and MATH 500 surpasses MSC and greatly outperforms the SC baselines for both generators. While MCC is marginally below MSC on GSM-hard and AIME, it still exceeds the best baseline. More interestingly, the overall performance remains similar when switching between strong and weak judges, indicating that the performance gains are robust across judges with varying levels of capability. This can be attributed to the fact that judges are not required to generate solutions from scratch, so their individual reasoning ability does not dominate MCC's outcome. Our multi-LLM architecture effectively lowers individual model reasoning demands while enhancing the system's overall reasoning capability.

For large-scale group, MSC consistently improves GPT 4.1's performance. Results on AIME highlight a notable competence gap, where Virtuoso Large performs much worse than GPT 4.1. Nevertheless, MCC with the weak judge can still perform virtually comparable to SC with GPT 4.1 on AIME. Remarkably, as MCC delegates nearly half of the computation to Virtuoso Large, a model that is significantly more affordable and faster than GPT 4.1, MCC achieved this performance at only 58% of the cost of MSC with GPT-4.1 alone: in each run, 16 of the 32 samples are produced by Virtuoso Large, whose output token cost is approximately 10 times lower than GPT 4.1's.

Given the huge capacity gap between GPT 4.1 and Virtuoso Large, their nearly identical SC performances on AQuA dataset suggest the larger LLMs may have become saturated on this benchmark. In such a case, MSC performs similarly to SC with Virtuoso Large, and MCC preserves the performance of the strongest baseline. While MSC with Virtuoso Large exhibits a marginal performance decline compared to SC on MATH 500, MCC in large-scale group continues to exceed the strongest baselines on both MATH 500 and GSM-hard.

3.2 MSC with Reasoning Model

We also evaluate MSC on GPT-OSS-20B, a state-of-the-art reasoning model within its size range. We selected AIME as our benchmark to avoid saturation, as reasoning models typically outperform non-reasoning models on reasoning tasks. As one of the most challenging benchmarks in mathematical reasoning, we observed that AIME often elicits responses from LLMs that are contaminated with

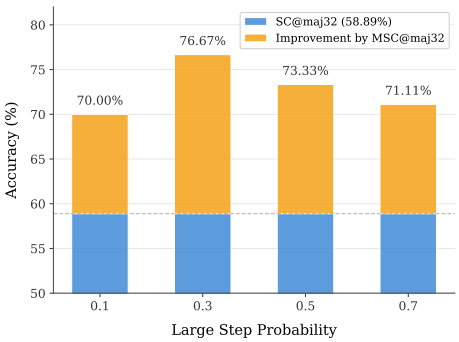

Figure 2: Ablation on large step probability.

MSC@32	Accuracy (%)
E only	70.00 ± 3.85
D only	73.33 ± 2.23
P only	72.59 ± 2.31
E + P	70.74 ± 2.56
E + D	69.26 ± 2.31
P + D	70.00 ± 1.11
E + D + P	73.33 ± 0.00

Table 2: Ablation of correctness function components. E, D, P denote step-level, token-level and path-level confidence respectively.

incorrect answers. This provides an ideal setting for which MSC is precisely designed. As shown on Figure 2, MSC leads SC around 15% in absolute accuracy (58.89% to 73.33%). We ablate large step probability (p_{large}) and observed that MSC's performance retains at $\geq 70\%$. p_{large} is the chance for MSC to conduct global mutation during an iteration: $p_{large} \to 0$ means MSC will dedicate to refining upon previously accepted paths; $p_{large} \to 1$ means MSC will keep looking for new paths.

We also ablate the 3 components of our correctness function and sample size in Table 2 and Figure 3, respectively. We find that using each confidence term alone produces unstable results, as individual components fail to capture certain types of errors in solutions. Pairing effectively enables the relatively stable term to stabilize the more volatile one. When all three terms are combined, standard deviation becomes negligible and performance also peaks. We also observe that variations stabilize as sample size approaches 32. Notably, even with a sample size as small as 4, MSC still delivers statistically significant improvements over SC, demonstrating remarkable sample efficiency.

Note that we set max output length to 4096 tokens, motivated by AIME ground truth solutions averaging approximately 1115 tokens. While many recent works permit outputs spanning tens or hun-

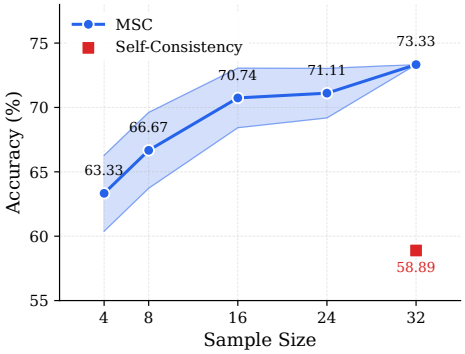

Figure 3: Analysis on sample size efficiency and stability.

Smaller Model	Method	AQuA
Gemma 2 27B	MSC (Standard)	63.66
	MSC (Normalized)	72.87
	MSC (Symmetric)	**76.81**
Larger Model	**Method**	**AIME**
Virtuoso Large	MSC (Standard)	7.56
	MSC (Normalized)	15.89
	MSC (Symmetric)	**20.44**

Table 3: Performance of MSC with standard / normalized / symmetric transition probabilities.

dreds of thousands of tokens despite reference solutions requiring only ≈ 1000 words, this raises concerns about practical applicability and interpretability. A key contribution of MSC is enabling competitive performance with VRAM requirements accessible to a broader research community.

Regarding token consumption, MSC matches SC with non-reasoning models, as both only output reasoning paths (see Appendix C). With reasoning model, MSC **reduces** token usage by $\approx 12\%$ compared to SC. This reduction mainly occurs because the reasoning tokens produced by refining a solution are often fewer than those produced from generating from scratch every time: local mutation can carry useful reasoning from previous iterations over to the current proposed state.

3.3 MSC with Asymmetric Transitions

An important design decision in translating MH algorithm to LLM workflow is how we treat the transition probabilities $Q(x|y)$ and $Q(y|x)$ in Equation (1). In MSC with $p_{large} = 0.5$, transition probabilities simplify to: $\frac{Q(x|y)}{Q(y|x)} = \frac{Q_{global}(x)+Q_{local}(x|y)}{Q_{global}(y)+Q_{local}(y|x)}$. The probabilities of getting the proposal sequence through a global mutation (Q_{global}) and a local mutation (Q_{local}) can be computed as the sum of log probability of all the tokens in the proposal sequence. Besides this standard probability calculation, we also evaluated: 1) length normalization of transition probabilities (Brown et al., 2020) by dividing $Q(y|x)$ by the number of tokens in y; 2) assuming symmetric mutations by letting $Q(y|x) = Q(x|y)$ so that they cancels out each other and yields equation (2). As shown in Table 3, standard transition probabilities yield the worst performance. This can be due to two failure modes as follows. 1) *Length bias.* When y and x differ substantially in length, MSC degenerates into selecting the longest sequences: in Equation 1, if y is much longer than x, $\frac{Q(x|y)}{Q(y|x)}$ goes to infinity which favors accepting y; if x is much longer than y, $\frac{Q(x|y)}{Q(y|x)}$ goes to 0 which favors retaining x. 2) *Refinement bias.* When x and y have similar lengths, $Q_{local}(y|x)$ is almost always much larger than the other three terms: LLM is substantially more confident under local mutation than under global mutation due to the extra provided context, and thus $Q_{local}(y|x)$ is much larger than $Q_{global}(x)$; $Q_{local}(y|x)$ is also much larger than $Q_{local}(x|y)$ as further refinement rarely produces an unrefined solution. This drives the acceptance rate a toward 0, causing refined proposals to be almost always rejected. Performance improves with length-normalized probabilities. As length-normalized probabilities across paths are similar, comparable or even better performances are observed when using symmetric transitions. These findings lead us to adopt symmetric transition probabilities in our main experiments, which improves effectiveness and efficiency by removing the need to collect log probabilities.

Smaller Model	Method	AQuA
Mistral Small 3 24B	Global-only MSC	52.99
	MSC (Symmetric)	**77.05**
Larger Model	**Method**	**AIME**
GPT 4.1	Global-only MSC	22.56
	MSC (Symmetric)	**48.00**

Table 4: Performance of global-only MSC.

3.4 Global-only MSC

In an earlier stage of our work, we implemented MSC by directly following the MH algorithm, i.e. Equation (1). The implementation relied solely on global mutation, the simplest mutation strategy that satisfies the MH requirements, and did not draw on any ideas from MLT. Since only global

mutation is involved here, Equation (1) becomes $a(y \mid x) = \min\left(1, \frac{\hat{f}(y)Q(x)}{\hat{f}(x)Q(y)}\right)$. We can eliminate the influence of sequence length by defining samples drawn by MSC as solely the final answer, and use SC's sample distribution to obtain a reliable estimate of the transition probabilities: $Q(\cdot) \approx \frac{N_{SC}(\cdot)}{N_{SC}}$ where N_{SC} is the sample size of SC, and $N_{SC}(\cdot)$ is the frequency of the final answer. Here, x and y represent final answers rather than full paths. As multiple paths can map to the same answer, $f(x)$ and $f(y)$ are replaced by realizations of random variables for the correctness of answers x and y, denoted as $\hat{f}(x)$ and $\hat{f}(y)$, respectively. One robust feature of MH algorithms is that, theoretically, the sampling still converges towards the expected values $E(\hat{f}(x))$ and $E(\hat{f}(y))$, which by definition are the correctness of the final answers.

As Table 4 shows, global-only MSC performs significantly worse, confirming that mixing global mutations and local perturbations for balancing ergodicity with local exploration is critical for effective sampling. This aligns with well-established results in MLT, where mutation strategy relying solely on random picking a new light path globally (called *new path mutation*) often do not work well without combining with other mutation strategies.

4 Related Work

Self Refinement (SR). SR can be done by tuning a model using its own generated data as contrastive example (Chen et al., 2024b). Structured search algorithms such as Monte-Carlo Tree Search have been used to collect more thoroughly explored data for self-supervised enhancement of LLM performance (Guan et al., 2025; Zhang et al., 2024). Beyond training-based methods, SR can be achieved through iterative self-exploration guided by self-diagnosed feedback during inference (Madaan et al., 2023; Miao et al., 2024; Hao et al., 2023). However, recent studies have shown that LLMs often struggle to distinguish their own responses from right or wrong (Huang et al., 2024a), limiting self-rewarding mechanisms' reliability. While being training-free, MCC integrates cross-consistency checks between distinct LLMs, effectively mitigating internal bias from SR.

Order Bias in LLM Judgments. Recent studies indicate that the ordering of information significantly influences LLM-generated judgments (Chen et al., 2024a). When using LLM-as-a-judge, a simple yet effective mitigation strategy is calculating win rate in pairwise comparisons to counteract this positional bias (Jiang et al., 2024; Lu et al., 2024). We adopt similar technique in MCC to ensure robust evaluations.

Best-of-N Sampling. Standard Best-of-N (BoN) sampling first generates N independent samples in parallel, then evaluates them post-hoc using a reward model to select the highest-scoring one via an argmax operation. In our context, this would correspond to scoring N independent SC samples with our correctness function and greedily selecting the best. Our framework differs structurally: correctness evaluation is integrated on-the-fly into the sampling process via the accept-reject mechanism, iteratively guiding subsequent samples rather than evaluating all samples after the fact. In particular, the argmax operation demands a reward model with sufficiently high resolution to strictly rank all candidates, whereas our approach imposes a weaker requirement: rather than relying on the correctness function to perfectly order all samples, it estimates the distribution of correctness over possible solutions, so that it moves the majority of samples to the bright regions without requiring the brightest one to be absolutely correct. Conversely, any effective reward model developed for BoN can be plugged into our framework as a drop-in correctness function, as our design is modular with respect to the choice of f.

Please see Appendix A for more related work.

5 Conclusion

In this work, we present MSC and MCC, a series of self- and cross-consistency framework that autonomously guides sampling toward a comprehensive and easily customizable notion of correctness. MSC/MCC allow flexible combination of mutation strategies to enhance LLM reasoning. Experiments show that MSC and MCC efficiently enhance reasoning performance of LLMs at various sizes and types on highly challenging mathematical tasks, demonstrating the potential of bridging multi-step LLM reasoning with PBR algorithms, and offering a novel perspective for error analysis in LLMs.

6 Limitations

Time Cost.

Our approach requires sequential processing, as each iteration depends on the previous one. Thus,

samples are not generated fully in parallel, and the wall-clock time grows linearly with sample size. That said, this linear scaling is predictable and well controlled, and far simpler than in tree- or graph-based reasoning frameworks, such as Tree of Thoughts (Yao et al., 2023), Graph of Thoughts (Besta et al., 2024), or MCTS-based methods, where intermediate states can grow exponentially with search depth. By using a chain-based structure, our method avoids this combinatorial blow-up while still supporting Exploration-and-Exploitation via global and local mutations. Meanwhile, our implementation fully parallelizes chains within multi-LLM architectures like MCC, partially mitigating the sequential overhead.

Correctness Representation and Hyperparameter Generalizability.

The correctness function f is defined solely by the model's internal confidence reflected in token probabilities. This choice preserves our framework's training-free design, but whether LLMs are capable of revealing the true correctness of their own outputs remains an open question. Even if a confidence-to-correctness mapping exists, it is likely to differ across models because of differences in training data and training architectures. Consequently, each model may require its own hyperparameter calibration for f to achieve optimal representation of correctness. By using a single set of hyperparameters for all models (e.g., proposal tolerance T and the weights on E, D, and P), our experiments may not fully unlock the algorithms' potential. This could partly explain why our algorithms produced more significant accuracy gains for some models while yielding only marginal improvements for others.

Minimal Mixing of Mutation Strategies.

Our implementation adopts a simple mixing strategy, combining only sampling fresh solutions and prompting the LLM to refine a previously accepted solution. In comparison, the original MLT framework (Veach and Guibas, 1997) proposes a richer set of mutation rules, each targeting specific families of light paths. By analogy, it may be beneficial to develop task-tailored mutation strategies for different types of reasoning paths. Nevertheless, our work provides a flexible framework that can accommodate other mutation strategy mixtures, and we leave the exploration of such extensions to future work.

References

AI-MO. 2024. AIMO Validation AIME Dataset. https://huggingface.co/datasets/AI-MO/aimo-validation-aime. Accessed: 2025-12-22.

Arcee AI. 2025. Releasing five open-weights models: Supernova 70b, virtuoso-large 72b, caller 32b, glm-4-32b-base-32k, and homunculus 12b. https://www.arcee.ai/blog/releasing-five-new-open-weights-models. [Online; accessed 2025-08-01].

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michał Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: solving elaborate problems with large language models. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Junying Chen, Zhenyang Cai, Ke Ji, Xidong Wang, Wanlong Liu, Rongsheng Wang, and Benyou Wang. 2025a. Towards medical complex reasoning with LLMs through medical verifiable problems. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 14552–14573, Vienna, Austria. Association for Computational Linguistics.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025b. Do NOT think that much for 2+3=? on the overthinking of long reasoning models. In *Forty-second International Conference on Machine Learning*.

Xinyun Chen, Ryan A. Chi, Xuezhi Wang, and Denny Zhou. 2024a. Premise order matters in reasoning with large language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024b. Self-play fine-tuning convertsweak language models to strong language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 81 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Olga Golovneva, Sean O'Brien, Ramakanth Pasunuru, Tianlu Wang, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. Pathfinder: Guided search over multi-step reasoning paths. *arXiv preprint arXiv:2312.05180*.

Google. 2024. Gemini 2.0. https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/?utm_source=deepmind.google&utm_medium=referral&utm_campaign=gdm&utm_content=.

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. In *Forty-second International Conference on Machine Learning*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173, Singapore. Association for Computational Linguistics.

W. K. Hastings. 1970. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024a. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*.

Zhen Huang, Haoyang Zou, Xuefeng Li, Yixiu Liu, Yuxiang Zheng, Ethan Chern, Shijie Xia, Yiwei Qin, Weizhe Yuan, and Pengfei Liu. 2024b. O1 replication journey - part 2: Surpassing o1-preview through simple distillation, big progress or bitter lesson? *CoRR*, abs/2411.16489.

Zhongzhen Huang, Gui Geng, Shengyi Hua, Zhen Huang, Haoyang Zou, Shaoting Zhang, Pengfei Liu, and Xiaofan Zhang. 2025. O1 replication journey - part 3: Inference-time scaling for medical reasoning. *CoRR*, abs/2501.06458.

Lingjie Jiang, Shaohan Huang, Xun Wu, and Furu Wei. 2024. Textual aesthetics in large language models. *arXiv preprint arXiv:2411.02930*.

Daniel Kahneman. 2011. *Thinking, fast and slow*. macmillan.

Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. 2002. A simple and robust mutation strategy for the metropolis light transport algorithm. In *Computer Graphics Forum*, volume 21, pages 531–540. Wiley Online Library.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, and 1 others. 2025. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904, Miami, Florida, USA. Association for Computational Linguistics.

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*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Zichen Liu, Changyu Chen, Wenjun Li, Tianyu Pang, Chao Du, and Min Lin. 2025. There may not be aha moment in r1-zero-like training — a pilot study. https://oatllm.notion.site/oat-zero. Notion Blog.

Jianqiao Lu, Zhiyang Dou, Hongru WANG, Zeyu Cao, Jianbo Dai, Yunlong Feng, and Zhijiang Guo. 2024. AutoPSV: Automated process-supervised verifier. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. 1953. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092.

Ning Miao, Yee Whye Teh, and Tom Rainforth. 2024. Selfcheck: Using LLMs to zero-shot check their own step-by-step reasoning. In *The Twelfth International Conference on Learning Representations*.

Mistral AI. 2025. Mistral-Small-24B-Instruct-2501. https://huggingface.co/mistralai/Mistral-Small-24B-Instruct-2501. Accessed: 2025-12-22.

Mistral team. 2025. Mistral small 3. https://mistral.ai/news/mistral-small-3. [Online; accessed 2025-08-01].

OpenAI. 2024. Gpt-4o. https://openai.com/index/hello-gpt-4o/.

OpenAI. 2024. Gpt-4o mini: Advancing cost-efficient intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/. Accessed: 2025-08-01.

OpenAI. 2024a. Learning to reason with LLMs.

OpenAI. 2024b. Reinforcement fine-tuning.

OpenAI. 2025a. Gpt-4.1. https://openai.com/index/gpt-4-1/. Accessed: 2025-08-01.

OpenAI. 2025b. GPT-OSS-120B & GPT-OSS-20B model card. Model card.

OpenAI. 2025. OpenAI o3-mini.

Zhenting Qi, Mingyuan MA, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2025. Mutual reasoning makes smaller LLMs stronger problem-solver. In *The Thirteenth International Conference on Learning Representations*.

Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, and Pengfei Liu. 2024. O1 replication journey: A strategic progress report - part 1. *CoRR*, abs/2410.18982.

Zeyu Tang, Zhenhao Chen, Xiangchen Song, Loka Li, Yunlong Deng, Yifan Shen, Guangyi Chen, Peter Spirtes, and Kun Zhang. 2025. Reflection-window decoding: Text generation with selective refinement. In *Forty-second International Conference on Machine Learning*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Fengwei Teng, Zhaoyang Yu, Quan Shi, Jiayi Zhang, Chenglin Wu, and Yuyu Luo. 2025. Atom of thoughts for markov llm test-time scaling. *CoRR*, abs/2502.12018.

Together AI. 2025. Together AI Playground. `https://api.together.ai/playground/chat`. Accessed: 2025-12-22.

Robert Vacareanu, Anurag Pratik, Evangelia Spiliopoulou, Zheng Qi, Giovanni Paolini, Neha Anna John, Jie Ma, Yassine Benajiba, and Miguel Ballesteros. 2024. General purpose verification for chain of thought prompting. *arXiv preprint arXiv:2405.00204*.

Eric Veach and Leonidas J. Guibas. 1997. Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, page 65–76, USA. ACM Press/Addison-Wesley Publishing Co.

Junlin Wang, Siddhartha Jain, Dejiao Zhang, Baishakhi Ray, Varun Kumar, and Ben Athiwaratkun. 2024. Reasoning in token economies: Budget-aware evaluation of LLM reasoning strategies. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19916–19939, Miami, Florida, USA. Association for Computational Linguistics.

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*.

P. C. Wason and J. S. T. B.. T. Evans. 1974. Dual processes in reasoning? *Cognition*, 3(2):141–154.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

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

Violet Xiang, Charlie Snell, Kanishk Gandhi, Alon Albalak, Anikait Singh, Chase Blagden, Duy Phung, Rafael Rafailov, Nathan Lile, Dakota Mahan, and 1 others. 2025. Towards system 2 reasoning in llms: Learning how to think with meta chain-of-thought. *arXiv preprint arXiv:2501.04682*.

Ling Yang, Zhaochen Yu, Bin Cui, and Mengdi Wang. 2025. Reasonflux: Hierarchical llm reasoning via scaling thought templates. *arXiv preprint arXiv:2502.06772*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yuxuan YAO, Han Wu, Zhijiang Guo, Zhou Biyan, Jiahui Gao, Sichun Luo, Hanxu Hou, Xiaojin Fu, and Linqi Song. 2024. Learning from correctness without prompting makes LLM efficient reasoner. In *First Conference on Language Modeling*.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Zhiyuan Zeng, Xiaonan Li, Junqi Dai, Qinyuan Cheng, Xuanjing Huang, and Xipeng Qiu. 2024. Reasoning in flux: Enhancing large language models reasoning through uncertainty-aware adaptive guidance. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2401–2416, Bangkok, Thailand. Association for Computational Linguistics.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. ReST-MCTS*: LLM self-training via process reward guided tree search. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Zhi Zhou, Tan Yuhao, Zenan Li, Yuan Yao, Lan-Zhe Guo, Xiaoxing Ma, and Yu-Feng Li. 2025. Bridging internal probability and self-consistency for effective and efficient llm reasoning. *arXiv preprint arXiv:2502.00511*.

Appendix

A More Related Work

Reinforcement Fine-Tuning (RFT). Recently, RFT has leveraged reward mechanisms to tune the reasoning behaviors of pretrained LLMs, significantly improving reasoning performance in specific tasks (OpenAI, 2024b,a, 2025; DeepSeek-AI et al., 2025). However, the mechanisms driving these improvements remain underexplored, and it is unclear whether foundational models inherently possess the ability of emerging Long-CoT without RFT (Liu et al., 2025). Moreover, generating extensive reasoning tokens not only introduces a significant VRAM challenge as discussed in Section 3.2, but models frequently struggle with long contexts, resulting in overthinking, topic drift, and context overflow (Xiang et al., 2025; Qin et al., 2024; Huang et al., 2024b, 2025; Tang et al., 2025; Teng et al., 2025). A major contributor to these inefficient behaviors, as mentioned in Section 2.2, is that existing reasoning models often fail to strike a balance between System 1 (fast) and System 2 (slow) thinking (Wason and Evans, 1974; Kahneman, 2011; Li et al., 2025; Chen et al., 2025b).

Structured Search. Structured approaches, including tree-based, graph-based, and macro-action-integrated methods, organize sampling systematically, allowing broader and detailed exploration of reasoning trajectories (Yao et al., 2023; Besta et al., 2024; Qi et al., 2025; Wu et al., 2024; Yang et al., 2025; Chen et al., 2025a). Elaborated multi-sampling pipelines tried to use token probability to identify and prune away low-confidence cluster in sample distribution (Zhou et al., 2025). However, these methods suffer from excessive context dependencies, high computational overhead with high intermediate samples redundancy, and sensitivity to manually-defined actions, limiting their generalizability and effectiveness. In contrast, our MSC method defines actions without manual intervention (global/local mutation). Inheriting from Metropolis-Hastings, MSC retains minimal historical information from only the previous iteration, and each accepted sample remains informative and contributes directly to the final answer selection. Unlike previous approaches that construct reasoning step by step or at the token level, MSC always sample at full-trajectory level. This global perspective helps avoid getting trapped in inconsistent states caused by undetected errors at early stages of the reasoning trajectory.

Evaluation using Token Probability. Recent research has leveraged token probabilities from LLM logits to design efficient, training-free statistical metrics for error detection in multi-step reasoning tasks, either at the step-level (YAO et al., 2024; Vacareanu et al., 2024; Golovneva et al., 2023) or at the token-level (Yin et al., 2024). Building upon these insights, our approach integrates error-detection metrics at multiple granularities, collectively offering a robust and comprehensive evaluation of solution correctness.

B Plot of Token Probabilities within Steps

Figure 4 shows the probabilities assigned to tokens in 10 randomly selected steps.

As we can see, the initial $\approx$10 tokens in erroneous steps often have lower probabilities. Although initial tokens in correct steps tend to have higher probabilities, the probabilities of later tokens in many correct steps can still fluctuate greatly.

C Prompt Details

The prompt templates we used for global mutations, local mutations and pairwise judgments are illustrated in Figure 5, Figure 6 and Figure 7, respectively.

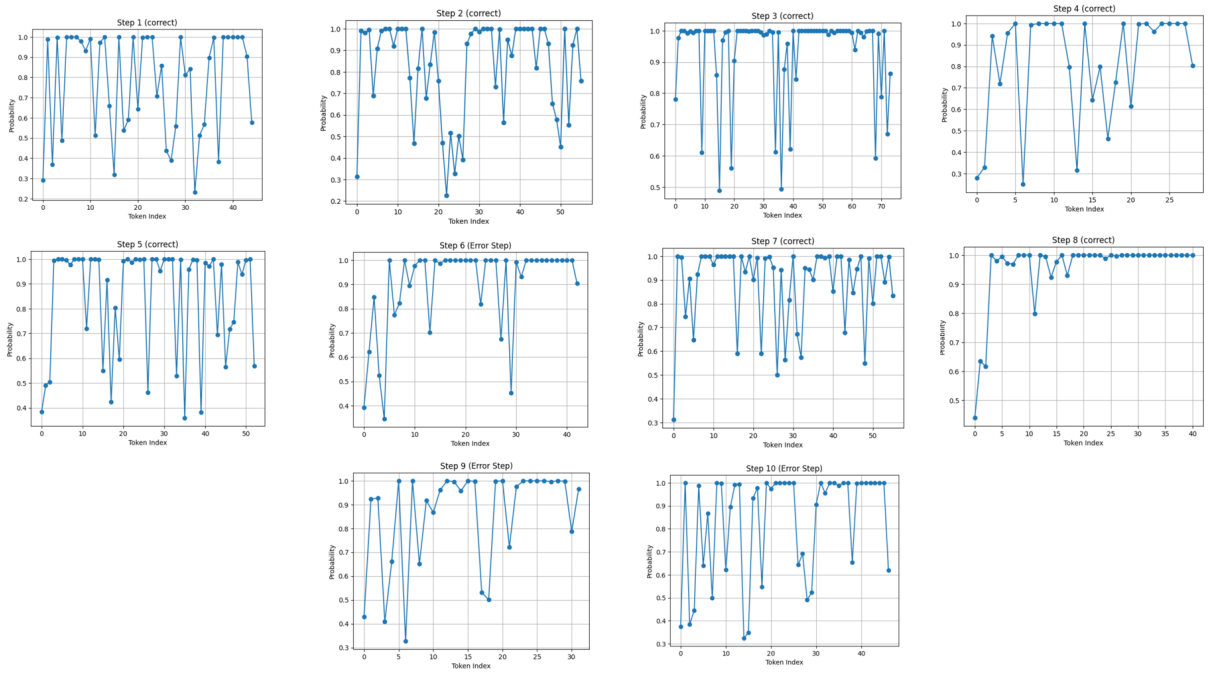

Figure 4: Plot of the token probabilities at 10 randomly selected steps.

Prompt Template (Global Mutation)

System Prompt:
You are a helpful assistant.
The user will provide a `{problem_type}`. Please solve the problem step by step.

```
<example_user_input>
Question:
{example_task_description}
</example_user_input>

<example_output>
{example_muti_step_solution}
</example_output>
```

User Prompt:
Question:
`{task_description}`

Figure 5

Prompt Template (Local Mutation)

System Prompt:
You are a helpful assistant.
The user will provide a {problem_type}, and an existing multi-step solution to the problem.
Your task is to generate a refined solution to the problem based on the existing solution, aiming to correct the mistakes (if any) in the existing solution.

<example_user_input>
Question:
{example_task_description}
Existing Solution:
{example_muti_step_solution}
</example_user_input>

<example_output>
{example_refined_muti_step_solution}
</example_output>

User Prompt:
Question:
{task_description}

Existing Solution:
{accepted_solution_from_previous_iteration}

Figure 6

Prompt Template (Pairwise Judgments)

User Prompt:
You are an expert in mathematics and logical reasoning.
You will be presented with a {problem_type}.
You will then be given two multi-step solutions (Solution A and Solution B) and asked to determine which solution better solves the problem.
Here is the competition math question:

<question>
question
</question>

Here are the two multi-step solutions to evaluate:

<solution_a>
{solution_a}
</solution_a>

<solution_b>
{solution_b}
</solution_b>

Please note that your task is NEITHER to correct errors (if any) in the provided solutions NOR to come up with your own solution; rather, your task is to choose the better one of the two. You must output **only** either SOLUTION A or SOLUTION B, without any additional words.

Figure 7: Note that our pairwise judgments do not use a system prompt. For each iteration, this user prompt is run twice in parallel, with the two proposals swapped between the solution_A and solution_B positions.