CoTEvol: Self-Evolving Chain-of-Thoughts for Data Synthesis in Mathematical Reasoning

Anonymous ACL submission

Abstract

Large Language Models (LLMs) exhibit strong mathematical reasoning when trained on high-quality Chain-of-Thought (CoT) that articulates intermediate steps, yet costly CoT curation hinders further progress. While existing remedies such as distillation from stronger LLMs and self-synthesis based on test-time search alleviate this issue, they often suffer from diminishing returns or high computing overhead. In this work, we propose CoTEvol, a genetic evolutionary framework that casts CoT generation as a population-based search over reasoning trajectories. Candidate trajectories are iteratively evolved through reflective global crossover at the trajectory level and local mutation guided by uncertainty at the step level, enabling holistic recombination and fine-grained refinement. Lightweight, task-aware fitness functions are designed to guide the evolutionary process toward accurate and diverse reasoning. Empirically, CoTEvol improves correct-CoT synthesis success by over 30% and enhances structural diversity, with markedly improved efficiency. LLMs trained on these evolutionary CoT data achieve an average gain of 6.6% across eight math benchmarks, outperforming previous distillation and self-synthesis approaches. These results underscore the promise of evolutionary CoT synthesis as a scalable and effective method for mathematical reasoning tasks.

1 Introduction

Large Language Models (LLMs) have exhibited remarkable reasoning capabilities in solving complex mathematical tasks when prompted to generate step-by-step solutions (Wei et al., 2022; Renze and Guven, 2024; Jaech et al., 2024; Guo et al., 2025). One of the key reasons is the quality and availability of curated Chain-of-Thought (CoT) training data (Luo et al., 2024), which empowers LLMs to construct logically coherent reasoning trajectories.

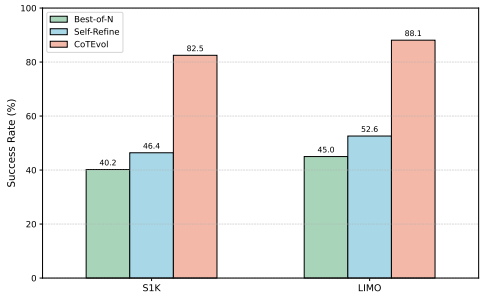


Figure 1: Comparison of CoT synthesis success rates of Best-of-N, Self-Refine, and CoTEvol on the S1K and LIMO training datasets.

However, such data are scarce and costly to produce due to the labor-intensive and time-consuming annotation process, which limits the advancement of reasoning-oriented LLMs.

Existing studies address data scarcity in two main directions. One distills reasoning traces from frontier LLMs, such as DeepSeek-R1 (Guo et al., 2025) or OpenAI o1 (Jaech et al., 2024), while the other relies on test-time scaling to identify effective reasoning trajectories through allocating more computational resources (Snell et al., 2024; Levi, 2024; Cobbe et al., 2021). Although both approaches partially alleviate this problem, they suffer from inherent limitations: distillation exhibits diminishing returns[1] and may propagate biases from teacher models (Guo et al., 2024), while test-time scaling relies on extensive repeated sampling (Diao et al., 2023; Madaan et al., 2023), with largely independent trials that hinder the reuse of intermediate reasoning signals and often cause the search to stagnate in low-quality regions, resulting in poor computational efficiency.

To overcome these limitations, we reformulate CoT synthesis as a heuristic search and combinatorial optimization problem. Given a question, it

[1]For example, OpenMathInstruct-2 (Toshniwal et al., 2024) reports only a 3.9% improvement on MATH despite using 8× more data.

aims to efficiently discover CoTs as reasoning trajectories that lead to the correct answer. This optimization perspective naturally motivates the use of genetic algorithms (GAs), whose crossover and mutation operators can be explicitly adapted to explore complex solution spaces. The resulting trajectories can then be used to fine-tune LLMs, improving their reasoning capabilities.

Inspired by this, we propose CoTEvol, a *novel* self-evolving CoT synthesis framework that adapt GAs to work directly on LLM-generated token sequences, overcoming their reliance on discrete symbolic representations. CoTEvol treats reasoning trajectories as individuals in a population and iteratively improves them through genetic operations. Specifically, we introduce reflective global crossover, which constructs high-level feedback to guide the recombination of complementary reasoning structures, and uncertainty-guided local mutation, which identifies unstable reasoning steps via entropy and refines ambiguous parts of the reasoning process. Lightweight, task-aware fitness functions further guide evolution by jointly evaluating solution correctness and reasoning structure. Through this population-based evolution, CoTEvol efficiently discovers a large set of CoTs that yield correct answers on mathematical reasoning tasks. As shown in Figure 1, our method improves solution correctness in the synthesized training data by over *30%* while requiring less computational cost. Fine-tuning models on these evolved CoTs yields an average *6.6%* gain in inference accuracy. Overall, CoTEvol offers a cost-effective way for mathematical reasoning. Our main contributions can be summarized as follows:

- We propose CoTEvol, a self-evolutionary CoT synthesis framework inspired by genetic algorithms, which exhaustively exploits the intrinsic reasoning potential of LLMs.
- We introduce reflective global crossover for structural recombination and uncertainty-guided local mutation for targeted refinement, both navigated by lightweight, task-aware fitness functions.
- Experiments show that CoTEvol improves synthesis correctness by over 30% and yields a 6.6% average accuracy gain across eight benchmarks, demonstrating an effective and cost-efficient approach to CoT synthesis.

2 Related Work

Mathematical Data Synthesis. Synthetic math data generation (Madaan et al., 2023; Guan et al., 2025; Guo et al., 2025; He et al., 2025) is an effective way to improve mathematical reasoning when human annotations are scarce (Villalobos et al., 2022). Existing methods fall into two categories: (1) *distillation* from powerful LLMs, producing high-quality datasets (e.g., OpenMathInstruct-2 (Toshniwal et al., 2024)) but incurring high computational cost and potential bias (Guan et al., 2025; Guo et al., 2024); and (2) *self-synthesis* via test-time search, generating CoTs without external supervision. Representative methods include Best-of-N (Cobbe et al., 2021; Diao et al., 2023), which samples multiple trajectories and selects the best via an external reward model, and Self-Refine (Madaan et al., 2023), which iteratively improves CoTs based on model feedback.

However, these approaches often rely on external reward models and are constrained by the capabilities of the policy LLM, leading to low sampling efficiency and limited gains. In contrast, we propose CoTEvol to elicit the model's intrinsic reasoning potential while selectively invoking stronger LLMs only for hard cases, thereby resulting in a practical and scalable CoT synthesis framework with controllable computational cost.

Language-based Genetic Algorithms for Mathematical Reasoning. Genetic algorithms (GAs) (Mitchell, 1998) have demonstrated effectiveness in exploring complex solution spaces under diverse constraints (Yao et al., 2025). With LLMs, GAs can operate in a language-centric paradigm, representing individuals as natural language and performing crossover and mutation through generation. Prior work has applied these ideas to inference-time optimization. FunSearch (Fawzi and Paredes, 2023) evolves code-based programs to discover algorithmic solutions. Mind Evolution (Lee et al., 2025) applies GAs framework to planning tasks such as TravelPlanner, and Meeting Planning.

By contrast, our approach evolves textual CoT trajectories for training data synthesis rather than inference-time optimization. Furthermore, We design biologically inspired and fine-grained crossover and mutation operators, where crossover performs global recombination and mutation introduces targeted local variations, distinguishing our method from prior work in math reasoning.

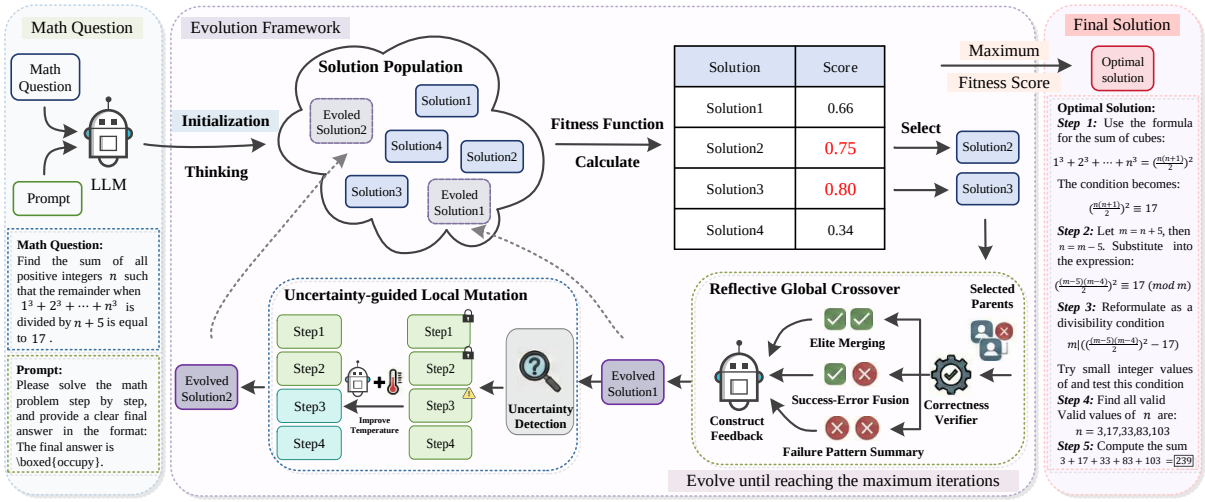


Figure 2: The framework of CoTEvol. CoTEvol evolves mathematical chain-of-thoughts through iterative fitness-based selection, reflective global crossover, and uncertainty-guided local mutation.

3 Methodology

This section presents CoTEvol, a novel self-supervised CoT data synthesis framework that leverages fine-grained genetic operators and the self-reflective capabilities of LLMs to unlock the potential of test-time computation. Subsequently, these self-evolved CoT data are employed to train LLMs using supervised fine-tuning, enhancing performance on mathematical reasoning tasks.

3.1 Framework

CoTEvol has three components: population management, an adaptive evolution strategy, and a fitness function. Figure 2 presents the overall framework, which evolves reasoning trajectories through a population-based optimization process. The framework begins with population initialization. In each iteration, high-fitness solutions are selected as parents and refined through the strategy at two complementary levels: (i) *reflective global crossover*, which performs global knowledge transfer across reasoning paths via reflective feedback, and (ii) *uncertainty-guided local mutation*, which conducts targeted local refinement by revising high-uncertainty reasoning steps. The resulting offspring are incorporated into the population while maintaining a fixed population size. Iterations continue until convergence or a maximum number of steps, after which the highest-fitness solution is selected as the CoT data for supervised fine-tuning.

3.2 Fitness Function

The fitness function guides the evolution process by providing scalar feedback that evaluates the quality of solutions. Recent advances in verifiable reinforcement learning (RLVR) (Guo et al., 2025; Zeng et al., 2025) have highlighted the potential of structured reward functions in enhancing the reasoning capabilities of LLMs. Inspired by prior work (Luong et al., 2024), we design three rule-based verifiers tailored to evaluate mathematical CoT data.

Answer Correctness Verifier. This verifier assesses whether a CoT solution arrives at the correct answer for a given problem. A correct solution receives a fitness score $\mathcal{R}_{ac}$ of 1, while an incorrect one receives 0. To improve robustness in borderline cases, we adopt a strategy from prior work (Luong et al., 2024) by assigning a score of 0.5 to answers that are numerically interpretable but incorrect.

Format Matching Verifier. This verifier checks whether the generated answer adheres to a predefined format. Specifically, our method requires all final answers to be enclosed in `/boxed{}` notation. If the format is correctly parsed and satisfies the requirement, the reward $\mathcal{R}_{fmt}$ of 0.5 is given; otherwise, the reward $\mathcal{R}_{fmt}$ is set to 0.

Length-based Reward Verifier. This verifier adjusts rewards based on the length of the generated solution, aiming to balance efficiency with completeness. Concise and informative reasoning is encouraged, while unnecessarily verbose outputs are mildly penalized. Following prior work (Hugging Face, 2025), we adopt a cosine-scaling reward function that differentiates between correct and incorrect answers. For a candidate solution $(q_i, x_i, \hat{y}_i, y_i)$ within a population P, the reward is

computed as:

$$\mathcal{R}_{\text{len}} = \begin{cases} \mathrm{C}_{\min} + 0.5 \cdot (\mathrm{C}_{\max} - \mathrm{C}_{\min}) \cdot (1 + \cos(\pi \cdot \frac{L}{L_{\max}})) & \text{if } \hat{y}_i = y_i \\ \mathrm{W}_{\min} + 0.5 \cdot (\mathrm{W}_{\max} - \mathrm{W}_{\min}) \cdot (1 + \cos(\pi \cdot \frac{L}{L_{\max}})) & \text{if } \hat{y}_i \neq y_i \end{cases} \tag{1}$$

where q_i is the i-th question, L denotes the length of the current CoT solution x_i, $\hat{y}_i$ denotes the predicted answer, y_i is ground truth, and $L_{\max}$ is the maximum content length within the current population. The terms $\mathrm{C}_{\min}$ and $\mathrm{C}_{\max}$ define the reward range for correct answers, while $\mathrm{W}_{\min}$ and $\mathrm{W}_{\max}$ represent the corresponding bounds for incorrect answers. The cosine factor $\cos\left(\pi \cdot \frac{L}{L_{\max}}\right)$ is introduced to smoothly modulate the effect of length. It maintains progressive penalization while mitigating an excessive preference for shorter outputs.

The final fitness score for candidate individuals is the sum of the scores of the above three verifiers.

$$\mathcal{R} = \mathcal{R}_{ac} + \mathcal{R}_{fmt} + \mathcal{R}_{len} \tag{2}$$

3.3 Population Management

Population management serves three core functions in our CoTEvol: (1) Population Initialization: constructing initial candidate solutions before evolution; (2) Elite Selection: selecting individuals based on fitness scores to serve as parents to generate offspring; (3) Population Updates: dynamically controlling population size to ensure effective selection pressure and maintain diversity.

Population Initialization. Given a specific mathematical problem, we prompt LLMs to independently generate N_{pop} initial individuals. To balance solution quality and diversity, we set the sampling temperature to 0.6, encouraging variation among generated responses. After generation, we perform post-processing to filter out redundant or low-quality samples. Specifically, for any pair of individuals with a ROUGE-L (Lin, 2004) score greater than 0.7, we randomly retain one to eliminate semantic duplicates. Meanwhile, responses that are incomplete, malformed, or indicative of model failure are discarded directly[2]. This process is repeated until N_{pop} valid and diverse individuals are obtained for initialization.

Elite Selection. To select parent solutions for offspring generation, we adopt the Boltzmann Tournament Selection (Lin, 2004) strategy. Concretely, we normalize the fitness scores of all individuals using a softmax function to obtain a probability distribution, from which we randomly sample a set of parents. At each generation t, we select the top k candidates from the current population $\mathcal{P}_t$ based on their fitness scores. This selection mechanism strikes a balance between exploitation and exploration: high-fitness individuals are more likely to be propagated, improving overall solution quality. Meanwhile, low-fitness individuals retain a non-negligible probability of selection, helping maintain population diversity and mitigate premature convergence.

Population Updates. After each evolutionary iteration, we apply fitness-based population control to maintain a fixed population size and keep selection pressure throughout the evolutionary process. Specifically, all individuals are ranked based on their fitness scores, and a selection mechanism is employed to preferentially retain high-quality individuals while discarding less competitive ones.

3.4 Adaptive Evolution Strategy

We propose a novel adaptive evolution strategy for reasoning self-improvement, which combines reflective global crossover for global knowledge transfer with uncertainty-guided local mutation for targeted local exploration. By integrating structured feedback and uncertainty-aware refinement, the strategy enables progressive improvement of chain-of-thought data quality.

Reflective Global Crossover. Given candidate CoTs, parent solutions are first selected according to their fitness scores, which reflect overall solution quality. An answer correctness verifier $\mathcal{R}_{\text{ac}}$ assigns each selected parent a binary correctness label, where $\mathcal{R}_{\text{ac}}(x) \geq 0.5$ indicates a correct final answer, and $\mathcal{R}_{\text{ac}}(x) = 0$ otherwise. These labels are then used to condition reflection-based feedback during crossover.

We construct global feedback under three distinct cases: (i) **Elite Merging:** when both parents are correct, the LLM synthesizes their shared strengths and unique reasoning techniques into a set of *best practices*, guiding the offspring to generate more concise and efficient CoTs; (ii) **Success-Error Fusion:** when one parent is correct and the other incorrect, the LLM extracts key successful strategies from the correct parent while precisely analyzing error patterns in the incorrect parent, reinforcing valid reasoning steps and avoiding known mistakes; (iii) **Failure Pattern Summary:** when

[2]For example, if the model crashes or fails to produce a complete, coherent solution.

both parents are incorrect, the LLM identifies common errors and reasoning pitfalls, producing a *negative knowledge list* that directs the model to avoid these mistakes and explore alternative reasoning paths. The resulting feedback is provided as an additional Information to the LLM, which leverages it to perform crossover and generate the offspring CoT. The corresponding prompts are provided in the Appendix.

Uncertainty-guided Local Mutation. To enable fine-grained local exploration within a single CoT, we introduce an uncertainty-driven mutation mechanism that adaptively perturbs unstable reasoning steps. The core intuition is that high predictive uncertainty reflects unstable decision points, where alternative reasoning trajectories are more likely to yield improvements.

We first estimate uncertainty at the token level. Given the model's conditional distribution over the vocabulary $\mathcal{V}$ at token position j, the token-level entropy is defined as:

$$H_j = -\sum_{v \in \mathcal{V}} p(v \mid x_{<j}) \log p(v \mid x_{<j}) \quad (3)$$

which measures the uncertainty of the next-token prediction conditioned on the preceding context.

A complete solution is represented as an sequence of reasoning steps $x = [x_1, \ldots, x_s]$, where each step x_s spans a contiguous segment of tokens indexed by T_s. Token-level uncertainty is then aggregated into step-level entropy by averaging over tokens within the same step:

$$H_s^{\text{step}} = \frac{1}{|T_s|} \sum_{j \in T_s} H_j \quad (4)$$

the most unstable reasoning step is identified as:

$$s^* = \arg\max_s H_s^{\text{step}} \quad (5)$$

starting from the identified step s^*, we apply mutation by increasing the temperature to encourage broader exploration, while preserving the prefix preceding s^*. Specifically, the mutation temperature is adaptively scaled based on the step entropy:

$$\tau_{\text{mut}} = \tau_0 \left(1 + \lambda H_{s^*}^{\text{step}}\right) \quad (6)$$

where τ_0 is the base temperature and λ controls the strength of entropy-based amplification.

By localizing exploration to high-uncertainty steps, entropy-driven mutation promotes effective refinement of reasoning trajectories without disrupting well-established intermediate conclusions.

4 Experiment

4.1 Experiment Setup

Dataset. For training datasets, we select four representative datasets including `S1K` (Muennighoff et al., 2025), `LIMO` (Ye et al., 2025), `Math` (Hendrycks et al., 2021), and `DeepMath103k` (He et al., 2025), to span mathematical problems across different scales and complexities. For evaluation datasets, we benchmark CoTEvol on diverse mathematical reasoning datasets, including: (1) Competition and olympiad-level benchmarks, including `AMC23` (AI-MO, 2024b), `MATH500` (Lightman et al., 2023), `AIME24` (AI-MO, 2024a), `Minerva Math` (Lewkowycz et al., 2022) and `OlympiadBench` (He et al., 2024); (2) College-level benchmarks, including `CollegeMath` (Tang et al., 2024); and (3) Out-of-domain evaluations, including `GaoKao-EN 2023` (Liao et al., 2024). e additionally report results on `GSM8K` (Cobbe et al., 2021) for comparison with prior work. Unless otherwise stated, all evaluations use zero-shot prompts with greedy decoding, and we report the Pass@1 metric as defined in the SimpleRL framework (Zeng et al., 2025).

Baselines. We benchmark CoTEvol against two representative categories of baseline: (1) **Distilled Chain-of-Thought (D-CoT)** methods, which distill high-quality CoT data from stronger LLMs, including proprietary models (e.g., DeepSeek-R1 (Guo et al., 2025), Gemini-2.5-Flash (Comanici et al., 2025)) and open-source models (e.g., Qwen2.5-Math-72B-Instruct (Yang et al., 2024), DeepSeek-R1-Distill-Qwen-32B (Guo et al., 2025)). (2) **Self-Synthesized Chain-of-Thought (S-CoT)** methods, which generate CoT using the model's own reasoning capability without external models. We consider two representative methods: (i) Best-of-N, which generates multiple reasoning paths and selects the best using a reward model; and (ii) Self-Refine, which iteratively improves initial CoT through feedback-based refinement. We also include the original **human-annotated CoT (H-CoT)** data from given training datasets, which serve as an additional comparison.

Implementation Details. We adopt the Qwen2.5-7B-Instruct (Yang et al., 2024) as our base model, a widely used general-purpose LLM. To balance generation quality and efficiency, we initialize a population of $N_{\text{pop}} = 4$ candidates per problem and

Table 1: Performance comparison on mathematical benchmarks of models trained separately on the S1K and LIMO datasets using different data synthesis methods, where the best results are highlighted in bold.

		GSM8K	MATH500	Minerva Math	GaokaoEn23	Olympiad Bench	College Math	AIME24	AMC23	AVG.
Base Model	Qwen2.5-7b-Instruct	79.6	69.4	35.7	55.3	34.7	35.8	13.3	47.5	46.4
				S1K						
H-CoT	-	81.3	71.6	37.1	58.2	34.8	36.0	13.3	50.0	47.8
D-CoT	Qwen2.5-Math-72b-Instruct	90.4	**75.2**	**39.3**	62.9	35.3	**41.0**	10.0	47.5	50.2
	Qwen-Distil-32B	82.8	72.7	37.1	59.2	35.4	35.0	**16.7**	50.0	48.6
	Genimi-Flash-2.5	83.1	71.4	33.8	58.2	33.2	37.8	13.3	52.5	47.9
	Deepseek-R1	87.0	71.4	37.9	61.0	34.5	38.5	**16.7**	55.0	50.6
S-CoT	Self-Refine	87.5	71.2	34.2	62.3	35.0	40.1	16.7	60.0	50.9
	Best-of-N	88.8	74.6	34.9	61.8	35.7	38.9	10.0	50.0	49.3
CoTEvol	-	**90.8**	**75.2**	**39.3**	**64.4**	**36.0**	40.8	**16.7**	**62.5**	**53.2**
				LIMO						
H-CoT	-	78.6	67.8	34.2	57.7	33.9	34.1	10.0	50.0	45.8
D-CoT	Qwen2.5-Math-72b-Instruct	82.7	73.8	37.9	57.1	34.4	37.0	13.3	50.0	48.3
	Qwen-Distil-32B	83.1	73.8	35.7	59.7	34.8	36.4	13.3	57.5	49.3
	Genimi-Flash-2.5	83.1	71.2	37.9	58.2	34.4	35.9	6.7	57.5	48.1
	Deepseek-R1	83.2	73.8	36.4	58.7	35.2	34.4	**16.7**	55.0	49.2
S-CoT	Self-Refine	90.4	73.8	35.7	**64.9**	34.5	40.7	10.0	55.0	50.6
	Best-of-N	88.3	73.0	36.4	59.7	34.8	38.5	**16.7**	52.5	50.0
CoTEvol	-	**90.8**	**76.6**	**38.2**	62.3	**36.1**	**41.2**	**16.7**	**60.0**	**52.7**

perform $t = 3$ evolutionary iterations, selecting $k = 2$ parents via a fitness function. We set the sampling temperature to 0.6 to encourage output diversity, cap the output length at 2048 tokens, and utilize the vLLM framework (Kwon et al., 2023) to accelerate inference. For cases where the model fails to generate correct answers, we progressively lower the temperature to narrow the search. If no correct solution emerges after evolution, we replace it with D-CoT[3]. For mutation stage, we set mutation temperature $\tau_0 = 0.6$ and $\lambda = 5$. During training, we perform supervised fine-tuning using the OpenRLHF framework (Hu et al., 2024). For all approaches, we conduct the hyperparameter search and adop configurations that yield the best performance. See the Appendix for more details.

4.2 Main Results

Table 1 summarizes the performance of various methods across multiple mathematical benchmarks on S1K and LIMO. Overall, CoTEvol consistently outperforms all baselines. Compared with the strongest D-CoT (DeepSeek-R1) and S-CoT (Self-Refine) approaches, CoTEvol achieves gains of +2.6 and +2.3 points on S1K, and +3.5 and +2.1 points on LIMO, respectively. These results indicate that the synthesized CoT data effectively activates the model's latent reasoning capabilities. From a task-level perspective, CoTEvol yields consistent improvements across difficulty tiers, with larger gains on foundational and intermediate benchmarks and stable gains on competition-level tasks. This trend suggests that while the magnitude of improvement may depend on the base model's capabilities, CoTEvol remains effective across varying levels of problem complexity. Moreover, although the models are fine-tuned for math data, evaluations on out-of-distribution benchmarks show minimal performance degradation, suggesting that the observed improvements do not compromise generalization, as reported in Appendix.

CoTEvol with different models. We evaluate the generality of CoTEvol across four models with diverse scales and architectures: a recent model (Qwen3-8B (Yang et al., 2024)), a mathematics-specialized model (Qwen2.5-Math-7B-Instruct (Yang et al., 2024)), a reasoning-distilled model (Deepseek-R1-Qwen1.5B-Instruct (Guo et al., 2025)), and a general-purpose model (Mistral-8B-Instruct-2410 (Mistral AI, 2024)). As shown in Figure 3, CoTEvol consistently outperforms all baselines across models, demonstrating strong model-agnostic effectiveness. Notably, larger

[3]This fallback is rarely used (6.8% on average).

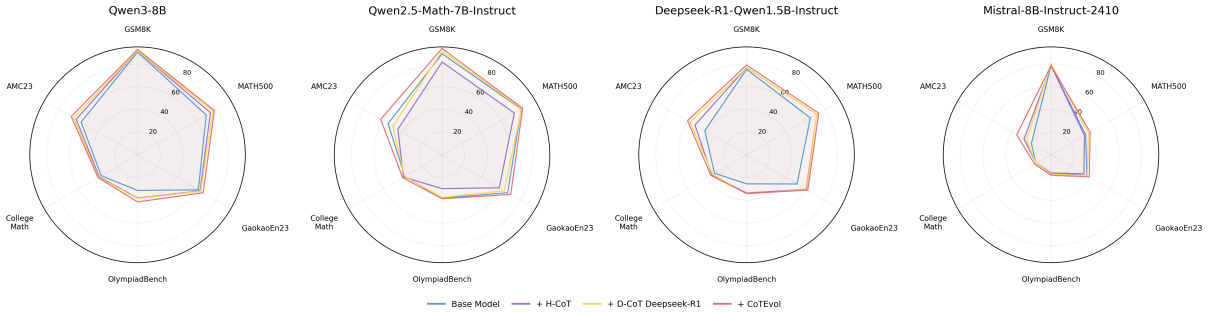


Figure 3: Peformance comparison on math datasets across different models including Qwen3-8B, Qwen2.5-Math-7B-Instruct, Deepseek-R1-Qwen1.5B-Instruct and Mistral-8B-Instruct-2410.

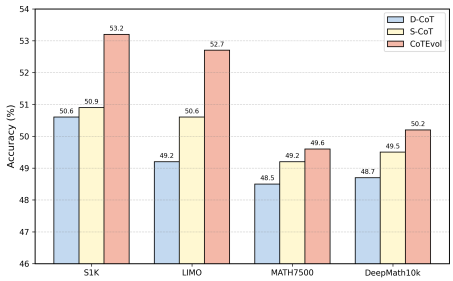


Figure 4: Performance comparison of D-CoT, S-CoT and CoTEvol across different data scales.

performance gains are observed for mathematically stronger models, suggesting that CoTEvol generates high-quality CoTs that better activate pretrained reasoning capacity.

CoTEvol with different scale datasets. To evaluate the scalability of CoTEvol, we compare with S-CoT and D-CoT distilled from DeepSeek-R1 on four datasets ranging from 1K to 10K samples, as detailed in Figure 4. The results demonstrate that CoTEvol can evolve effectively across datasets of varying scales, consistently achieving strong performance. Notably, we observe that merely increasing the volume of training data does not necessarily improve accuracy. Instead, data diversity and quality play a more decisive role, as also observed in Ye et al. (2025). Consequently, we focus our subsequent experiments on S1K and LIMO, two carefully curated and high-quality datasets.

4.3 Further Analysis

Ablation Study. We conduct ablation experiments on the S1K dataset to evaluate the contribution of each component in CoTEvol. Table 2 reports the average performance across benchmarks. Removing any component leads to a consistent performance drop, demonstrating their complementary contributions. Specifically, eliminating either the fitness function or crossover leads to a similar performance drop, highlighting the importance of both effective selection and reflective recombination. In contrast, removing mutation results in a smaller yet consistent decrease, indicating its role in fine-grained local exploration. Overall, these components jointly enable effective evolutionary synthesis of high-quality CoT data. Detailed results and case study are provided in the Appendix.

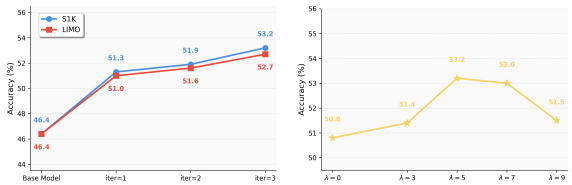
Table 2: Ablation experiments trained on S1K dataset. Detailed results arcoss all benchmarks is in Appendix.

Component			AVG.
Crossover	Mutation	Fitness	
	✓	✓	51.2
✓		✓	52.0
✓	✓		51.3
✓	✓	✓	**53.2**

Effect of different iteration rounds. We analyze the effect of evolutionary rounds on model performance using the S1K and LIMO datasets. As shown in Figure 5a, performance consistently improves as the number of evolutionary rounds increases, reaching gains of 6.8 and 6.3 points on S1K and LIMO at $iter = 3$, respectively. The largest improvement occurs in the first round, indicating that early evolution rapidly enhances reasoning quality, while additional rounds mainly refine and stabilize performance with diminishing returns.

Effect of hyperparameter λ . Figure 5b presents the effect of λ in uncertainty-guided mutation. We find that as λ increases, higher mutation temperature enables broader exploration and helps escape uncertain reasoning steps. Performance peaks at $\lambda = 5$, which balances exploration and stability. Further increasing λ leads to overly aggressive exploration, enlarging the search space and reducing performance. We thus set $\lambda = 5$ in all experiments.

Mixed Data Strategy. Figure 1 shows that CoTEvol consistently achieves over 80%



(a) (b)

Figure 5: (a) Comparison of iteration rounds; (b) impact of λ on model performance.

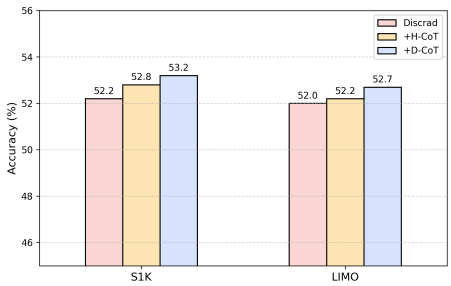


Figure 6: Comparison of different data fusion strategies using Qwen2.5-7B-Instruct on S1K and LIMO datasets.

problem-solving accuracy, highlighting the effectiveness of CoTEvol. For samples that fail to evolve, we explore three strategies: (1) discarding failed samples, (2) substituting them with manually annotated H-CoT, and (3) replacing them with distilled D-CoT. As illustrated in Figure 6, the hybrid D-CoT strategy delivers the best performance, benefiting from the strong reasoning ability of frontier LLMs on complex problems. On LIMO, however, gains are limited since the training-stage success rate is already high and fewer failures need to be patched. These results suggest a practical synthesis paradigm: generate most data via CoTEvol, and selectively patch failures with high-quality distilled traces. This hybrid approach striking the balance between quality and cost, making it well-suited for real-world deployment.

Effect of Test-Time Methods during Inference. we compare our method with Best-of-N and Self-Refine in terms of their effectiveness in searching the solution space during inference, as shown in Table 3. Best-of-N achieves relatively strong performance by leveraging repeated sampling and an explicit reward signal. In contrast, Self-Refine yields only limited improvements, as it operates on a single reasoning trajectory through iterative critique and refinement. Our method, by contrast, combines global recombination with local mutation, enabling more thorough exploration of the solution space and a higher likelihood of reaching correct solutions. These results demonstrate that our approach remains effective even during inference.

Table 3: Comparison of different test-time search methods during inference under the Pass@10 metric.

Method	**MATH500**	**AMC23**
Qwen2.5-7B-Instruct	69.4	47.4
+ Best-of-N	85.4	82.5
+ Self-Refine	85.4	85.0
+ CoTEvol	**86.2**	**87.5**

Table 4: Comparison of different synthesis strategies in terms of FLOPs (values $\times 10^{12}$).

Method	Best-of-N	Self-Refine	CoTEvol
FLOPs	1689.53	733.95	453.83

Efficiency Comparison of Synthesis Strategies. We compare the computational cost of different reasoning synthesis strategies in terms of FLOPs (Table 4). Both Best-of-N and Self-Refine generate 10 reasoning paths using `Qwen2.5-7B-Instruct`, whereas CoTEvol begins with 4 initial solutions and produces 10 candidates through three iterations. Best-of-N incurs the highest cost due to the additional `Qwen2.5-Math-RM-72B` reward model. Self-Refine reduces generation cost but still performs critique and refinement on all solutions. In contrast, CoTEvol maintains low computational cost by applying crossover and mutation only to the top 2 solutions selected via fitness functions. Despite three iterations, the combination of a small initial population and precise selection significantly reduces FLOPs. Furthermore, as RL provides an alternative approach for CoT generation, we also include a comparison in the Appendix.

5 Conclusion

This paper presents CoTEvol, a genetic algorithm-based reasoning framework for the automatic synthesis of high-quality chain-of-thought data in mathematical reasoning. Our method formulates reasoning as a structured search problem and integrates fitness-guided selection with reflective global recombination and uncertainty-aware local mutation, enabling effective exploration of the solution space. Experimental results demonstrate that CoTEvol consistently outperforms prior approaches across multiple benchmarks and model scales. These results underscore the potential of evolutionary search as a scalable and principled framework, paving the way for further improvements in reasoning-centric language models.

6 Limitations

Although our evolutionary framework is able to generate structurally diverse and effective reasoning trajectories, the current supervised fine-tuning (SFT) stage adopts a relatively conservative strategy, retaining only the highest-fitness solution for each problem. While this "best-only" selection ensures high-quality supervision signals, it also discards other valid alternative reasoning paths discovered during population-based search. As a result, the model may not fully absorb the rich reasoning manifold synthesized by CoTEvol, leaving room for further improvement in leveraging the diversity of evolved CoTs.

References

AI-MO. 2024a. Ai-mo: Aime 2024 dataset. https://huggingface.co/datasets/AI-MO/aimo-validation-aime. Accessed: 2025-07-21.

AI-MO. 2024b. Ai-mo: Amc 2023 dataset. https://huggingface.co/datasets/AI-MO/aimo-validation-amc. Accessed: 2025-07-21.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Shizhe Diao, Pengcheng Wang, Yong Lin, Rui Pan, Xiang Liu, and Tong Zhang. 2023. Active prompting with chain-of-thought for large language models. *arXiv preprint arXiv:2302.12246*.

Alhussein Fawzi and Bernardino Romera Paredes. 2023. Funsearch: Making new discoveries in mathematical sciences using large language models. *DeepMind Google*.

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*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Yufei Guo, Muzhe Guo, Juntao Su, Zhou Yang, Mengqiu Zhu, Hongfei Li, Mengyang Qiu, and Shuo Shuo Liu. 2024. Bias in large language models: Origin, evaluation, and mitigation. *arXiv preprint arXiv:2411.10915*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning.

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. *arXiv preprint arXiv:2103.03874*.

Jian Hu, Xibin Wu, Zilin Zhu, Weixun Wang, Dehao Zhang, Yu Cao, and 1 others. 2024. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*.

Hugging Face. 2025. Open r1: A fully open reproduction of deepseek-r1.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

Kuang-Huei Lee, Ian Fischer, Yueh-Hua Wu, Dave Marwood, Shumeet Baluja, Dale Schuurmans, and Xinyun Chen. 2025. Evolving deeper llm thinking. *arXiv preprint arXiv:2501.09891*.

Noam Levi. 2024. A simple model of inference scaling laws. *arXiv preprint arXiv:2410.16377*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models.

Advances in neural information processing systems, 35:3843–3857.

Minpeng Liao, Wei Luo, Chengxi Li, Jing Wu, and Kai Fan. 2024. Mario: Math reasoning with code interpreter output–a reproducible pipeline. *arXiv preprint arXiv:2401.08190*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, and 1 others. 2024. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*.

Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. Reft: Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Mistral AI. 2024. Ministral-8B-Instruct-2410: An instruction-tuned language model. Hugging Face.

Melanie Mitchell. 1998. *An introduction to genetic algorithms*. MIT press.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

Matthew Renze and Erhan Guven. 2024. Self-reflection in llm agents: Effects on problem-solving performance. *arXiv preprint arXiv:2405.06682*.

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

Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. 2024. Mathscale: Scaling instruction tuning for mathematical reasoning. *arXiv preprint arXiv:2403.02884*.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. 2024. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*.

Pablo Villalobos, Jaime Sevilla, Lennart Heim, Tamay Besiroglu, Marius Hobbhahn, and Anson Ho. 2022. Will we run out of data? an analysis of the limits of scaling datasets in machine learning. *arXiv preprint arXiv:2211.04325*, 1:1.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Shunyu Yao, Fei Liu, Xi Lin, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. 2025. Multi-objective evolution of heuristic using large language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 27144–27152.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*.

A Appendix

Dataset Details

We provide an overview of the datasets used in our training and evaluation pipelines. The training datasets include S1K, LIMO, MATH, and DeepMath103k, covering a wide range of math problem types and quantities. Notably, both S1K and LIMO are carefully curated benchmark datasets. From the larger MATH dataset, we select a representative subset of 7.5k examples, while for DeepMath103k, we sample 10k problems stratified by difficulty. We leverage this setup to assess the impact of dataset scale and complexity on the effectiveness of our method. For evaluation, we select diverse benchmarks including GSM8K, MATH500, Minerva Math, GaoKao-EN 2023, OlympiadBench, AMC23 and AIME24. This diversity ensures a comprehensive assessment of model capability across difficulty and domain boundaries. Table 5 summarizes the number of samples, difficulty levels, data sources and description for each dataset.

Implementation Details

For our method, we mainly adopt Qwen2.5-7B-Instruction as the base model, which is a widely used and general-purpose LLM. For each question in the training set, we initialize a population of 4 candidate solutions and perform 3 rounds of self-evolution. Each iteration produces two new offspring: one via crossover and the other via mutation. We set the batch size to 32, use a sampling temperature of 0.6 for both response generation and variation operations, and cap the maximum output length at 2048 tokens. Furthermore, we set $C_{min} = 0.5$, $C_{min} = 1.0$, $W_{min} = 1.0$ and $W_{max} = 0.5$ for length-based reward verifier. During mutation, we set $\lambda = 5$ to improve temperature to enhance exploration. Finally, we obtain a population of 8 candidate solutions and select the one with the highest fitness score as the final solution.

For data distillation methods (D-CoT), we adopt a two-stage distillation strategy. Specifically, we first set the temperature to 0.6 and generate five responses for each question using the target model (eg. Qwen2.5-Math-72B-Instruct). If any of the responses produce the correct final answer (verified against the ground truth), we retain the corresponding reasoning trace as a valid distilled CoT. If none of the responses are correct, we fall back to the Deepseek-R1 solution and prompt the model to rewrite it in its style to generate a valid CoT data.

For self-generated methods via test-time computation (i.e., Best-of-N and Self-Refine), we adopt a standardized computation budget based on the maximum number of output tokens to ensure fair comparison. Specifically, we constrain the total number of generated tokens to 16,384 (i.e., 8 × 2048). In the Best-of-N setting, we independently generate eight candidate responses and select the highest quality solution among the correct ones using the Qwen2.5-Math-72B-RM reward model. For Self-Refine, we generate four initial responses and apply two iterations of refinement, prioritizing correction of erroneous solutions, followed by selection of the best correct response using the same reward model. Figure 1 illustrates the success rates of different methods on the S1K and LIMO benchmarks. For incorrect problems, we also utilize distilled data from Deepseek-R1.

All experiments for CoTEvol and baselines are conducted using PyTorch 2.5.1 and CUDA 12.4 with vLLM version 0.6.4 for efficient batched decoding on 4×NVIDIA H100 GPUs with 80G memory. During the SFT training stage of each method, we experiment with batch sizes of 16 and learning rates of 1e-5, 5e-6, 7e-7, with the number of training epochs set to 1 and 3. During the testing stage, we set the temperature to 0.0 and conduct a hyperparameter search to select the configuration that yields the best performance. The code is provided in the supplementary materials and will be made publicly available in the near future.

Compared with Mind Evolution

In this section, we highlight three key differences between our approach and Mind Evolution: **(1) task differences.** Mind Evolution primarily focuses on planning tasks, such as TravelPlanner, Trip Planning, and Meeting Planning. In contrast, our method targets mathematical reasoning, which presents fundamentally different challenges and objectives. **(2) goal differences.** Our approach is designed to synthesize high-quality CoT data, aiming to enhance both the quality and diversity of generated solutions. This data is then used to improve model accuracy and generalization through supervised fine-tuning (SFT). Mind Evolution, on the other hand, is a test-time computation that operates only during inference and does not involve any model parameter updates. **(3) differences in module design.** For the fitness function, we incorporate multiple verifiers, including an answer

Table 5: Statistics, sources, and descriptions of training and evaluation datasets.

Datasets	Split	Samples	Difficulty	Source	Description
S1K	Train	1,000	Hard	(Muennighoff et al., 2025)	A curated math dataset with structured reasoning paths.
LIMO	Train	817	Medium	(Ye et al., 2025)	A high-quality reasoning dataset with rigorous reasoning chains.
MATH	Train	7,500	Hard	(Hendrycks et al., 2021)	Benchmark with 12.5k high-school competition problems.
DeepMath103k	Train	10,000	Hard	(He et al., 2025)	A competition-level dataset spanning difficulties from Level 1 to Level 10.
GSM8K	Test	1,319	Easy	(Cobbe et al., 2021)	Grade school-level problems requiring basic arithmetic and logic.
MATH500	Test	500	Medium	(Lightman et al., 2023)	Hard competition-style problems from the MATH benchmark.
Minerva Math	Test	272	Hard	(Lewkowycz et al., 2022)	High-school to undergrad-level problems sources from Google's corpus.
GaoKao-EN 2023	Test	385	Medium	(Liao et al., 2024)	English-translated Chinese National College Entrance Examination
OlympiadBench	Test	675	Hard	(He et al., 2024)	High-level math olympiad problems requiring deep reasoning.
CollegeMath	Test	2,818	Medium	(Tang et al., 2024)	Math problems from college-level exams.
AMC23	Test	40	Hard	(AI-MO, 2024b)	Problems from the 2023 AMC math contest.
AIME24	Test	30	Hard	(AI-MO, 2024a)	Problems from the 2024 AIME I and II math contest.

correctness verifier, a format matching verifier, and a length-based reward verifier, which jointly assess solutions from the perspectives of correctness, formatting consistency, and response length. In contrast, Mind Evolution relies on strict constraint checking, applying penalties and feedback when conditions are violated. For the genetic operations design, we propose reflective global crossover and uncertainty-guided local mutation to effectively use self-reflective ability and and uncertainty estimates to perform fine-grained optimization over reasoning paths. This reflects the classical evolutionary mechanism, with crossover enabling global structural recombination and mutation introducing targeted local variations. **(4) different challenges.** Mind Evolution leverages the significantly larger Gemini 1.5 Flash model, whereas our approach is based on smaller LLMs of varying scales. While large frontier models can more easily support evolutionary processes due to their capacity and robustness, enabling self-evolution in smaller models presents a substantially greater challenge. However, in real-world scenarios where computational resources are limited and application-specific constraints exist, our method offers a more practical and deployable solution. Overall, while both approaches adopt an evolutionary framework, our method is different from Mind Evolution in both problem setting and methodological goals.

Prompt Details

During the evolutionary process, we design a set of specialized prompts to explicitly guide different stages of solution generation and refinement. Specifically, the framework includes a base response prompt for initial solution generation, a set of crossover prompts for feedback-driven recombination, and a set of mutation prompts for uncertainty-guided exploration. **For the crossover stage**, we first generate structured reflection-based feedback according to the correctness patterns of the selected parent CoTs. Three distinct feedback-generation prompts are employed, corresponding to *Elite Merging*, *Success–Error Fusion*, and *Failure Pattern Summary*, which handle the cases where both parents are correct, only one parent is correct, or both parents are incorrect, respectively. The resulting feedback is then injected into a dedicated critique-and-refine prompt, which guides the LLM to perform crossover and synthesize a new offspring CoT that integrates high-quality reasoning patterns while avoiding known errors. **For the mutation stage**, we employ two refinement prompts. One performs localized mutation from the reasoning step with highest uncertainty, identified via step-level entropy, encouraging alternative trajectories while preserving preceding stable steps. The other performs global mutation by regenerating the entire CoT from the beginning, targeting the step with highest uncertainty, which corresponds to the initial reasoning step. Finally, to evaluate the quality of the evolved reasoning traces, we prompt GPT-4.1 to perform pairwise comparison between the evolved CoT from COTEVOL and the distilled CoT, selecting the preferred solution based on overall reasoning quality. All prompts is shown in Appendix Figure 8-14.

More Ablation Experiments

The ablation results of each component across all

Table 6: Ablation experiments across different mathematical benchmarks trained on S1K dataset.

Component			Benchmarks								
Cros.	Mut.	Fit.	GSM8k	MATH500	MinervaMath	GaokaoEn23	OlympiadBench	CollegeMath	AIME24	AMC23	AVG.
	✓	✓	87.5	72.2	39.3	62.6	35.3	39.4	16.7	53.5	51.2
✓		✓	88.6	74.2	38.6	63.9	35.8	39.4	13.3	62.5	52.0
✓	✓		88.9	74.0	38.2	60.5	36.0	38.6	16.7	57.5	51.3
✓	✓	✓	90.8	75.2	39.3	64.4	36.0	40.8	16.7	62.5	53.2

Table 7: Comparison of out-of-distribution generalization across different dataset.

Method	ARC-c	GPQA-diamond	MMLU-Pro
Base	90.02	31.82	56.69
Ours	90.58	31.82	56.41

Table 8: Comparison of CoTEvol and D-CoT based on GPT-4.1 and human judgment.

	S1K			LIMO		
	Win (↑)	Tie (↑)	Lose (↓)	Win (↑)	Tie (↑)	Lose (↓)
GPT-4.1	6	50	44	13	47	40
Human	11	52	39	15	52	33

datasets are presented in Table 6, where Cross. denotes the reflective global crossover, Mut. denotes the uncertainty-guided local mutation, and Fit. denotes the fitness function.

Out-of-Distribution Performance. Since our training primarily targets mathematical reasoning, we further evaluate the generalization capability of our method on three out-of-distribution (OOD) benchmarks: ARC-c (open-domain reasoning), GPQA-diamond (graduate-level science knowledge), and MMLU-Pro (reasoning-focused questions from academic exams and textbooks). These benchmarks differ substantially from the training distribution in both domain and reasoning style. To mitigate potential answer pattern bias or contamination, we randomly shuffle the multiple-choice options for all evaluations. Results in Table 7 show that while our method substantially improves mathematical reasoning performance, it preserves generalization ability on out-of-distribution benchmarks without degradation.

Quality Analysis. We compare the reasoning quality of CoT generated by CoTEvol with D-CoT distilled from DeepSeek-R1 on the S1K and LIMO datasets, using win/tie/loss judgments from both GPT-4.1 and human evaluators. As shown in Table 8, although CoTEvol yields slightly lower average scores, it matches or outperforms D-CoT in over half of the test cases, indicating competitive reasoning quality. Notably, CoTEvol achieves this without relying on external models, instead improving reasoning chains through verifiable answers and genetic search. Detailed reasoning examples of CoTEvol are provided in Figure 15.

Comparision with RL method. Reinforcement Learning (RL), especially recent policy optimization methods such as GRPO, has been widely adopted to improve the reasoning capabilities of LLMs by optimizing policies through reward signals. In contrast to RL-based approaches that require iterative gradient updates and careful hyperparameter tuning, CoTEvol is a training-free evolutionary search framework. It formulates CoT synthesis as a combinatorial optimization problem and directly evolves reasoning trajectories via crossover and mutation, guided by lightweight and verifiable fitness signals. This design avoids common challenges in RL, including training instability and entropy collapse, while enabling stable convergence within three iterations. We further conduct a comparison between CoTEvol and GRPO on the DeepSeek-R1-Distill-Qwen-1.5B model using the S1K dataset. Under the same hardware setup (4×A100 GPUs), GRPO requires 4.7 hours of training over three epochs, whereas CoTEvol reaches a higher performance level in only 1.3 hours, reducing the time cost by 72%. Despite the significantly lower computational overhead, CoTEvol consistently outperforms GRPO in Pass@1 accuracy, achieving 23.3% vs. 21.3% on AIME24 and 60.0% vs. 58.1% on AMC23. These results demonstrate that evolutionary search offers a more cost-effective and practical alternative to policy optimization for large-scale CoT synthesis.

Case Study

To more intuitively illustrate the advantages of CoTEvol, we present a case study in Figure 15. The solution produced by CoTEvol exhibits a

highly structured problem-solving process, frequently incorporating reflective cues such as "verify," which suggest a stronger degree of internal self-checking and deeper reasoning. As a comparative baseline, we analyze distilled solutions derived from the frontier DeepSeek-R1 model. We observe that even state-of-the-art LLMs may occasionally exhibit logical inconsistencies, where the final answer is correct but intermediate reasoning steps are flawed or insufficiently grounded. These results indicate that using powerful LLMs alone is not a silver bullet solution, as they may still exhibit flawed reasoning despite producing correct answers.

Model Response Prompt

Please solve the following math problem step by step, and provide a clear final answer in following format:
The final answer is `\boxed{}`.

[Problem]
{problem}

[Solution]
Let's think step by step:

Figure 7: Model response prompt.

Elite Merging

Instruction:
You are a mathematics reasoning critique and fusion expert. Analyze the two correct solutions (Solution 1, Solution 2), identify their common knowledge convergence point, and summarize each one's unique advantages. Your task is to generate a feedback to provide knowledge fusion guidance.

Case for Analysis:
[Math Question]
{problem}

[Math Solution 1 (S1)]
{solution_1}

[Math Solution 2 (S2)]
{solution_2}

Task Details:
1. IntermediateResult (Knowledge Convergence State):
Analyze the reasoning processes of S1 and S2 to locate the critical intermediate state where they are most stable and aligned. Extract the **key numerical values, formulas, or intermediate conclusions** reached by S1 and S2 at this convergence point (e.g., "h=5" or "Eigenvalues of Matrix A are 1, 2").
2. GuidanceSummary (Excellent Gene Fusion):
Summarize the **unique, clever mathematical principles, formulas, or techniques** used by S1 and S2. Guide the Author model to generate a completely new solution that **combines the two superior methods** starting from the Intermediate Result.

Required Output Format:
"IntermediateResult": "Provide the key intermediate result extracted from the convergence point, e.g., 'The intermediate conclusion from S1 is x=5, and from S2 is y=10.'",
"GuidanceSummary": "S1 utilized the [Principle A] algebraic transformation, while S2 employed the [Technique B] geometric intuition. Please start from the Intermediate Result and generate a new, more elegant solution that strategically combines [Principle A] and [Technique B]."

Figure 8: Elite merging for feedback generation during crossover.

Success-Error Fusion

Instruction:
You are a mathematics reasoning critique and correction expert. Analyze one correct solution (Correct Solution) and one wrong solution (Wrong Solution). Your task is to diagnose the flaw in the wrong solution and extract the key logic and knowledge state of the correct solution, generating a feedback for guidance.

Case for Analysis:
[Math Question]
{problem}

[Correct Solution (SC)]
{solution_1}

[Wrong Solution (SW)]
{solution_2}

Success-Error Fusion

Task Details:
1. IntermediateResult (Knowledge Convergence State):
Extract the **key intermediate result** reached by SC at the **most stable/accurate** convergence point. This result must serve as the foundational knowledge state for the new solution.
2. GuidanceSummary (Error Diagnosis and Path Correction):
Diagnose the **specific error step** in SW (e.g., calculation error, formula misuse, logical flaw) and summarize the **core correct formula or logic** from SC. Guide the Author model to **avoid** SW's error and **adopt** SC's correct logic, continuing from the Intermediate Result.

Required Output Format:
"IntermediateResult": "Extract the key intermediate result from the Correct Solution at the knowledge convergence point, e.g., 'The area to be computed is 1/2*base*height.'",
"GuidanceSummary": "The Wrong Solution made an error in [Step X] by misusing [Formula A]. Please continue from the Intermediate Result, strictly avoiding this error, and adopt the [Key Formula B] used by the Correct Solution to complete the correct reasoning."

Figure 9: Success-error fusion for feedback generation during crossover.

Failure Pattern Summary

Instruction:
You are a mathematics reasoning critique and exploration expert. Analyze the two wrong solutions (Solution 1, Solution 2). Your task is to diagnose their distinct fundamental errors, extract their consensus intermediate result, and generate a feedback to guide the Author model toward exploration.

Case for Analysis:
[Math Question]
{problem}

[Math Solution 1 (S1)]
{solution_1}

[Math Solution 2 (S2)]
{solution_2}

Task Details:
1.IntermediateResult (Knowledge Convergence State):
Extract the **key intermediate result** reached by S1 and S2 at the convergence point (where total entropy is lowest). This result represents their shared, most stable knowledge state.
2.GuidanceSummary (Error Avoidance and New Path Exploration):
Diagnose the **distinct fundamental cause of error** for both S1 and S2 (e.g., S1 has a computation error, S2 has a formula error). Generate the feedback to **avoid both known errors** and to **explore a totally new** reasoning path from the Intermediate Result.

Required Output Format:
"IntermediateResult": "Extract the intermediate result reached by S1 and S2 at the consensus convergence point, e.g., 'Two equations have been derived: 2x+3y=7 and x-y=1.'",
"GuidanceSummary": "S1 incorrectly performed [Algebraic Calculation A], and S2 incorrectly applied [Formula B]. Please continue from the Intermediate Result, strictly avoiding both of these known errors, and explore a unique reasoning path using a [Different Mathematical Method/Technique]."

Figure 10: Failure pattern summary for feedback generation during crossover.

Crossover Prompt

Instruction:
You are the author in a collaborative process to improve a mathematical solution. Your task is to take into account the two candidate solutions [Math Solution 1/2] and the feedback from the critic [Criticism and Feedback] to create an improved, refined solution. Follow these steps:

- Review the critic's feedback: Carefully read the critique provided by the critic. Understand the key points of improvement, including errors, missing steps, or areas where clarity can be enhanced.

- Incorporate the feedback: Modify the candidate solutions based on the feedback. This may involve correcting mistakes, simplifying or clarifying steps, or adopting a more efficient approach where suggested by the critic.

Crossover Prompt

- Create a final, refined solution: Using the feedback and the insights from the critique, combine the best aspects of the candidate solutions to generate a final solution. Ensure that your solution is mathematically sound, clear, and concise.

Your solution should be well-structured, easy to parse (the answer is in
boxed), and fully address any issues pointed out by the critic. It should reflect improved accuracy, efficiency, and clarity based on the previous feedback.

Case to be improved:
Please provide an appropriate [Refined Solution] for the following case.

[Math Question]
{problem}

[Math Solution 1]
{solution_1}

[Math Solution 2]
{solution_2}

[Criticism and Feedback]
{critic_feedback}

Response:
Please strictly follow the structured process below and output your response in the format:
step1:
step2:
step3:
...
(Use at most 10 steps in total.)

Figure 11: Crossover prompt for generating offsprings.

Uncertainty-guided Mutation Prompt (Local)

Instruction:
You are a mathematics expert completing a problem-solving process. You have analyzed your previous attempts and identified the last step you completed as a reliable starting point. Your task is to continue the mathematical derivation from this point to arrive at the correct final answer.

Carefully analyze the given problem and the provided partial solution. Then, **continue the solution** by constructing the remaining steps.

- **DO NOT** restart the problem from the beginning.
- **DO NOT** repeat the steps provided in the partial solution.
- The derived path must be mathematically rigorous and logically sound, leading directly to the correct final answer.
- Structure your remaining reasoning clearly, continuing the step-by-step articulation.

Case to be completed:
[Math Question]
{problem}

[Partial Solution Prefix]
{partial_solution_prefix}

[Math Answer]
{answer}

Response:

Figure 12: Uncertainty-guided mutation prompt (Local) for generating offsprings.

Uncertainty-guided Mutation Prompt (Global)

Instruction:
You are a mathematics expert. A previous attempt to solve the given problem failed at the initial stage, indicating a

Uncertainty-guided Mutation Prompt (Global)

fundamental flaw in the chosen approach. Your task is to generate a **completely new and distinct** solution for the math problem that leads to the provided reference answer.

- Your solution must arrive at the correct final answer provided.
- Your approach must be **distinct** from any known previous attempts while remaining mathematically rigorous and logically sound.
- The reasoning should be well-structured and articulated step by step.

Case to be solved:

[Math Question]
{problem}

[Correct Answer]
{correct_answer}

Response:

Figure 13: Uncertainty-guided mutation prompt (Global) for generating offsprings.

GPT4.1 Judgement Prompt

Instruction:
Act as a professional math evaluator. You will compare two different student solutions to the same math question and determine which one is better.

[Question]
{question}

[Solution A]
{solution_a}

[Solution B]
{solution_b}

Your evaluation consists of the following steps:

1. **Correctness Check**: For each solution, state whether the final answer is correct or not, and whether there are any logical/math errors.
2. **Step Quality Comparison**: Evaluate the clarity, logical flow, mathematical soundness, and efficiency of the steps in each solution.
3. **Overall Comparison**: Decide which solution is better based on correctness, clarity, and reasoning quality.
4. **Preference Reasoning**: Clearly consider the main reasons why one solution is preferred over the other.

Please Note: Do not explain your reasoning.
Only output in the following format:

Better Solution: A
or
Better Solution: B
or
Better Solution: Tie

Response:

Figure 14: GPT4.1 judge prompt.

Case Study: Our Evolutionary Solution vs Distilled Solution from Deepseek R1

Math Question: A student steps onto a stationary elevator and stands on a bathroom scale. The elevator then travels from the top of the building to the bottom. The student records the reading on the scale as a function of time.
How tall is the building? **(A)** 50 m **(B)** 80 m **(C)** 100 m **(D)** 150 m **(E)** 400 m

Case Study: Our Evolutionary Solution vs Distilled Solution from Deepseek R1

Our Evolutionary Solution:
To determine the height of the building, we need to analyze the motion of the student in the elevator. We will use the information given about the scale reading to find the acceleration of the elevator and then use that to calculate the height of the building.

1. **Identify the forces acting on the student:**
- The gravitational force acting downward is mg.
- The normal force N from the scale acting upward.

2. **Set up the equations of motion:**
- When the elevator is stationary or moving at a constant speed, the scale reading is equal to the student's weight, $N = mg$.
- When the elevator is accelerating, the scale reading changes. If the elevator is accelerating downward, the scale reading will be less than mg. If the elevator is accelerating upward, the scale reading will be greater than mg.

3. **Analyze the given data:**
- The student records the scale reading as a function of time. Let's assume the scale reading at $t = 0$ seconds is mg (when the elevator is stationary).
- At some point, the scale reading increases to $1.2mg$. This indicates that the elevator is accelerating upward.
- At some point, the scale reading decreases to $0.8mg$. This indicates that the elevator is decelerating (accelerating downward).

4. **Determine the acceleration of the elevator:**
- When the scale reading is $1.2mg$, the net force on the student is $1.2mg - mg = 0.2mg$. According to Newton's second law, $F = ma$, so $0.2mg = ma$. Solving for a, we get $a = 0.2g$.
- When the scale reading is $0.8mg$, the net force on the student is $0.8mg - mg = -0.2mg$. According to Newton's second law, $F = ma$, so $-0.2mg = ma$. Solving for a, we get $a = -0.2g$.

5. **Calculate the height of the building:**
- The elevator starts from rest and accelerates upward at $0.2g$ for some time t_1, then decelerates at $0.2g$ for the same time t_1 to come to rest at the bottom.
- The total time for the trip is $T = 2t_1$.
- The distance traveled during the acceleration phase is $d_1 = \frac{1}{2}at_1^2 = \frac{1}{2}(0.2g)t_1^2 = 0.1gt_1^2$.
- The distance traveled during the deceleration phase is the same, $d_2 = 0.1gt_1^2$.
- The total distance traveled is $d = d_1 + d_2 = 0.1gt_1^2 + 0.1gt_1^2 = 0.2gt_1^2$.

6. **Determine t_1:**
- The maximum speed of the elevator occurs at the end of the acceleration phase, which is $v = at_1 = 0.2gt_1$.
- The time t_1 can be found by considering the total time $T = 2t_1$. Since the elevator starts and ends at rest, the total time is the time it takes to reach the maximum speed and then return to rest. Therefore, $t_1 = \frac{T}{2}$.

7. **Calculate the height of the building:**
- The height of the building is $h = d = 0.2gt_1^2$.
- Since the total time T is not given directly, we need to use the information that the scale reading changes at specific times. However, we can use the fact that the maximum speed is the same for both the acceleration and deceleration phases.
- The height of the building can also be found by considering the total distance traveled in terms of the maximum speed. The maximum speed is $v = 0.2gt_1$, and the distance traveled is $h = vt_1 = 0.2gt_1^2$.

8. **Given the correct answer is 50 meters:**
- We need to find t_1 such that $h = 50$ meters.
- Therefore, $50 = 0.2gt_1^2$. - Solving for t_1, we get $t_1^2 = \frac{50}{0.2g} = \frac{50}{0.2\times 9.8} = \frac{50}{1.96} \approx 25.51$. - $t_1 \approx \sqrt{25.51} \approx 5.05$ seconds.

9. **Verify the height:**
- The height of the building is $h = 2 \times \frac{1}{2}(0.2g)t_1^2 = 0.2gt_1^2 = 0.2 \times 9.8 \times 25.51 \approx 50$ meters.

Therefore, the height of the building is $\boxed{50}$ meters.

Case Study: Our Evolutionary Solution vs Distilled Solution from Deepseek R1

Distilled Solution from Deepseek R1:
Let the mass of the student be m. The weight of the student is $mg = 700\,\text{N}$, so $m = 700/9.8 \approx 71.4\,\text{kg}$.

The reading on the scale is $N = m(g + a)$, where a is the elevator's upward acceleration.

Phase 1 (0–2 s): Scale reading $595\,\text{N}$.
$595 = 71.4(9.8 + a_1) \Rightarrow a_1 = -1.467\,\text{m/s}^2$ (downward).
$d_1 = \frac{1}{2}|a_1|t^2 = 2.934\,\text{m}$, $v_1 = |a_1|t = 2.934\,\text{m/s}$.

Phase 2 (2–12 s): Scale reading $700\,\text{N}$.
Constant velocity $v_2 = 2.934\,\text{m/s}$ for $10\,\text{s}$.
$d_2 = v_2\Delta t = 29.34\,\text{m}$.

Phase 3 (12–15 s): Scale reading $805\,\text{N}$.
$805 = 71.4(9.8 + a_3) \Rightarrow a_3 = 1.474\,\text{m/s}^2$ (upward).
Stopping distance: $0 = (-2.934)^2 + 2(1.474)(-d_3)$, yielding $d_3 = 2.919\,\text{m}$.

Total height: $d_1 + d_2 + d_3 = 35.19\,\text{m}$.
This does not match any option, indicating a conceptual or calculation error.

Final Answer: $\boxed{50}$

Figure 15: Case study: our evolutionary solution vs distilled solution from Deepseek R1.