Test-Time Scaling for Multistep Reasoning in Small Language Models via A* Search

Anonymous ACL submission

Abstract

Large language models (LLMs) have demonstrated strong abilities across various tasks but are costly in computation and memory. In contrast, small language models (SLMs) offer significant advantages in efficiency and deployability but usually struggle with complex mathematical reasoning tasks. To tackle this issue, we present the Test-time A* Search (TTA*), a test-time scaling framework that casts reasoning as a goal-directed search over a tree of partial solutions, guided by an A*-style cost function. TTA* is training-free and requires no external supervision or multi-model orchestration, making it practical in resource-constrained settings. As a drop-in decoding wrapper for SLMs, TTA* systematically explores, critiques, and refines candidate solution paths via its own self-reflection capability. Extensive experiments[1] on popular mathematical reasoning benchmarks (GSM8K, MATH500, AIME, etc.) and a variety of base models show that TTA* consistently improves accuracy and robustness, indicating broad applicability to general mathematical reasoning tasks.

1 Introduction

Large language models (LLMs), such as GPT-5 (OpenAI, 2025) and Claude Opus 4.6 (Anthropic, 2026), have demonstrated strong performance on a wide range of tasks. However, their high performance comes at the cost of substantial computational and memory requirements, which complicates personalization (e.g., on-device adaptation) and limits deployment in resource-constrained settings. Small language models (SLMs), in contrast, offer an attractive alternative due to their efficiency, lower latency, and ability to run locally (Cheng et al., 2024; Wang et al., 2024; Al-Garadi et al., 2025). Consequently, SLMs are increasingly viewed as a promising foundation for agentic AI systems, especially in remote or resource-limited environments (Belcak et al., 2025). Nevertheless, the development of SLM still faces challenges in complex reasoning tasks, particularly in high-stakes domains such as mathematics and healthcare (Ma et al., 2025; Ahn et al., 2024). In these complex domains, SLMs often underperform on multi-step reasoning and long-horizon decision making which are the critical capabilities for safety and reliability (Huang et al., 2025b; Zhou et al., 2025; Glazer et al., 2024; Lewkowycz et al., 2022).

To improve the reasoning capabilities of SLMs, a growing body of work addresses this. However, many approaches rely on additional training, such as preference optimization from human feedback or distillation from larger teacher models, to encourage more reliable step-by-step reasoning (Ong et al., 2025). Although effective, these methods introduce significant training cost and deployment complexity, making them less suitable for resource-constrained or rapidly evolving settings.

To avoid the cost and complexity of retraining, *test-time scaling* methods have been proposed, in which models allocate additional inference-time computation via structured search in order to improve answer quality. However, many existing test-time scaling approaches rely on ancillary components such as progress reward models (PRMs) or other forms of external guidance, which complicate deployment and increase engineering overhead. A complementary line of work explores self-rewarding or self-reflection strategies, where models critique their own intermediate outputs. While promising, these methods can amplify errors across multiple steps due to the limited evaluative capacity of small models (Sakib et al., 2025).

In this paper, we pursue a practical *test-time scaling* approach with self-reflection for small language models, aiming to increase reliability without the infeasible costs of additional training (Xia et al.,

[1] Our code is available on anonymous repository https://anonymous.4open.science/r/TTA-Star-Search-9042/README.md

2024). We introduce **Test-time A* Search (TTA*)**, a framework that integrates heuristic search with SLMs to improve solution quality in complex reasoning tasks. TTA* formulates reasoning as a goal-directed search over a tree of partial solutions, where nodes represent candidate derivations and expansions are prioritized using an A*-style score (Hart et al., 1968). This score combines a cost-to-come based on the search trajectory with a conservative heuristic estimating future potential from the model's self-evaluation, rather than treating self-assessments as direct rewards. The resulting procedure is *training-free*, requires no external supervision or multi-model orchestration, and supports explicit compute budgets with anytime behavior—making it well suited for deployment in resource-constrained environments. As illustrated in Figure 1, our contributions are as follows:

- We formulate multi-step reasoning as a guided tree search and leverage model self-reflection to define a heuristic that prioritizes high-quality solution paths.
- We design a conservative cost function that balances exploration with skepticism toward self-evaluation. By aggregating multiple self-assessments via median scoring and prompting the model as a harsh critic, our scoring mechanism reduces variance and mitigates overconfidence, enabling more reliable evaluation of both partial and final solution nodes.
- We demonstrate through extensive experiments across multiple benchmarks and SLM backbones that TTA* consistently improves accuracy, often matching the reasoning performance of substantially larger models without additional training, external supervision, or multi-model orchestration.

2 Related Works

2.1 Multi-step reasoning in LLMs

Recent advances in LLM research highlight their ability to perform multi-step reasoning. Chain-of-thought (CoT) prompting (Wei et al., 2023) exposes intermediate steps, improving performance on complex tasks such as mathematical problem solving. Beyond single-path CoT, search-based methods explore multiple reasoning trajectories before producing a final answer. In particular, tree search techniques integrate with LLMs to expand and prune reasoning branches: Monte Carlo Tree Search (MCTS) has been applied to explore intermediate steps, evaluate partial solutions, and backtrack from low-value branches (Xie et al., 2024; Zhang et al., 2024a). These approaches systematically balance exploration and exploitation, allowing models to consider alternative reasoning paths and recover from early errors. However, many of these methods rely on external reward models, curated supervision (e.g., outcome-based or process-based rewards), or fine-tuned evaluators. Such dependencies can limit practicality in constrained deployment settings or when access to large models is restricted.

Earlier work also explored distilling reasoning capabilities from large LLMs into smaller models (Zhu et al., 2024; Zhang et al., 2025). While distillation can transfer some reasoning skills, it is computationally intensive, often requires teacher models with high accuracy ceilings, and provides limited gains when task distributions shift. Other studies have leveraged stronger models to generate synthetic datasets for fine-tuning smaller models (Liu et al., 2023), or used larger models as test-time supervisors (Yang et al., 2025b), but both approaches introduce additional compute burdens and dependency on external resources, restricting applicability in low-resource or test-time-only scenarios.

2.2 Small Language Models

Small Language Models (SLMs) are particularly attractive for deployment in resource-constrained environments, including edge devices and mobile platforms, due to their low memory footprint and computational requirements (Belcak et al., 2025). Enhancing their reasoning capabilities allows these lightweight models to handle complex tasks efficiently, without relying on large-scale LLMs, enabling real-time inference and practical applications in low-power or embedded settings.

Improving mathematical reasoning in small language models (SLMs) has been explored through knowledge distillation, fine-tuning, and inference-time assistance from larger models. Knowledge distillation (Zhu et al., 2024; Zhang et al., 2025) transfers reasoning capabilities from a high-performing teacher model to an SLM, but is computationally expensive, time-consuming, and ultimately bounded by the teacher's performance. Inference-time supervision methods, such as *Speculative Thinking*, instead pair a small model with a stronger

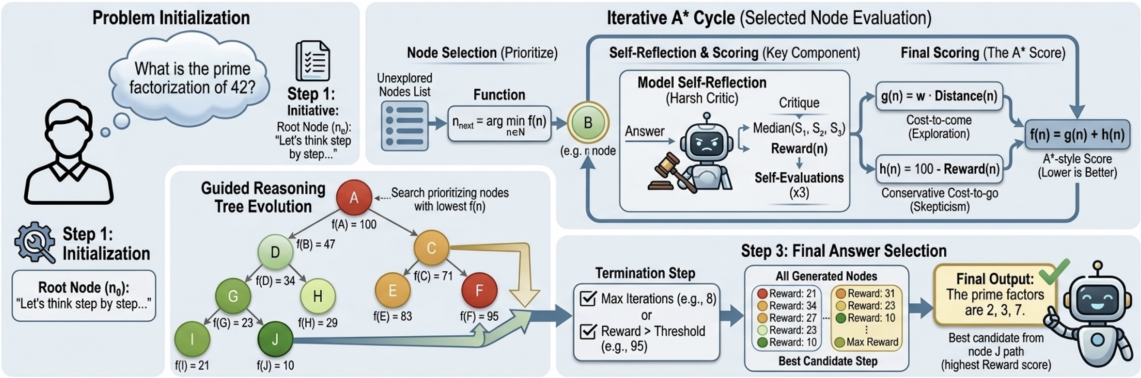

Figure 1: TTA* methodology overview. TTA* is a single-model, training-free test-time strategy that frames reasoning as a heuristic tree search problem. The search prioritizes nodes based on an A*-style cost function, balancing path distance (g) and a conservative self-reflection reward (h). During each iteration, a harsh critic prompt (shown in panel 2) is used, with the median of multiple self-rewards (e.g., 3 evaluations) calculated to reduce variance and mitigate overconfidence common in SLMs, as described in sections 3.2-3.4.

"guide" to assist with challenging reflective steps during decoding (Yang et al., 2025b). While effective, these approaches introduce a dependency on real-time access to large models, limiting their practicality and scalability.

Other methods, like *rStar-Math*, combine SLMs with internal reward models to drive search and iterative self-evaluation (Guan et al., 2025). These frameworks are effective, but require multi-round data generation, reward model training, and careful orchestration between models, increasing engineering complexity and compute costs. Direct fine-tuning methods, including Solution Guidance (Bi et al., 2025) or reinforcement learning-based approaches (Dang and Ngo, 2025), similarly incur high costs, rely on additional supervision, and limit accessibility for low-resource settings.

2.3 Self-reflection and self-rewarding

An alternative line of work equips LLMs with *self-evaluation* signals to guide reasoning. Here, the model generates candidate derivations and attaches lightweight diagnostics, such as confidence estimates, critiques, or partial checks, which inform revisions, backtracking, or reordering. Such mechanisms reduce ungrounded derivations and help prioritize promising reasoning branches without relying on ground-truth labels. Recent tree-search systems incorporate internal assessments as *progress heuristics* or *value estimates* to steer exploration, sometimes augmented with verifier-like signals or learned progress models (Xie et al., 2024; Zhang et al., 2024a). Notably, the Tree of Thoughts (ToT) framework (Yao et al., 2023) formalizes this idea by structuring reasoning as an explicit tree search over candidate reasoning steps, demonstrating that self-evaluation can effectively guide multi-step reasoning in large language models.

Although self-guided evaluations remove dependence on external supervision, they are prone to error amplification when the evaluator is weak, which is common in small models (Zhu et al., 2024). Consequently, small models' self-evaluation can be noisy, leading to overconfidence in incorrect answers or excessive critique in accurate responses (Zhang et al., 2024b). RLHF-style self-critique (Lee et al., 2024) has been studied to address this issue, but typically requires additional training, external supervision, or complex multi-step orchestration, substantially increasing the engineering burden—a demand often impractical for SLM deployment.

Our work builds on this direction by integrating self-evaluation directly into an A*-style cost function that balances cost-to-come and cost-to-go. During test-time, the model performs a self-reward step by querying itself three times per candidate solution and taking the median score to reduce variance and mitigate overconfidence. To further guide the evaluation, the prompt is designed to act as a harsh critic, penalizing errors and unlikely reasoning steps. This approach limits error propagation during search and functions entirely within the A*-style framework, without requiring additional training, external supervision, or complex orchestration, making it practical for deployment in real-world SLMs.

2.4 Training-free and inference-time methods for SLMs

In contrast to prior approaches, our method is a single-model, training-free, test-time strategy that frames multi-step reasoning as a heuristic search problem. By deriving the heuristic entirely from the model's own critiques and self-evaluations, we remove the need for external rewards, teacher models, or fine-tuning. At the same time, structured exploration and self-correction are preserved. This yields a practical, low-resource framework for improving SLM reasoning that combines tree-based search with model-internal feedback, scales naturally with model size and complexity, and provides a deployable alternative for enhancing small models' reasoning capabilities in real-world scenarios.

3 Methodology

3.1 Motivation

LLM reasoning frequently spans long, multi-step derivations (Shen et al., 2025). When treated as a single forward pass, early mistakes propagate and compound, degrading reliability in tasks such as mathematical problem solving. This is only exacerbated in SLMs, which lack the same depth as LLMs and are often observed to hallucinate or repeat mistakes (Ma et al., 2025).

To improve trustworthiness, reasoning should be decomposed into structured, incremental tasks that build upon one another, allowing the model to maintain coherence and reduce the risk of compounding errors (Wu and Tsioutsiouliklis, 2024). Consequently, long and complex problems—where LLMs are more prone to hallucination—should be broken down into smaller, manageable steps that can be sequentially solved and then combined to reach a final solution.

To effectively manage the complexity above, we propose representing the LLM reasoning process as a tree. This structure naturally supports the exploration of multiple reasoning trajectories, enabling branching, backtracking, and iterative refinement.

Prior research has demonstrated that such approaches can significantly boost accuracy through iterative refinement and the identification of high-quality leaf nodes (Zhang et al., 2024a; Ahn et al., 2024; Xie et al., 2024). However, many of these tree search approaches for LLMs rely on auxiliary components—such as reward/value models, progress reward models, or additional fine-tuning—to evaluate partial solutions and guide exploration, which undermines deployability in resource-constrained environments. Our goal is to build on this foundation, leveraging tree-based reasoning to improve reasoning accuracy, specifically in limited-capacity models.

3.2 Tree-Based Reasoning

We begin with the formulation of the tree-based reasoning, as illustrated in Figure 1. In particular, each node n represents a partial, intermediate, or imperfect reasoning attempt and edge denoted by $e(n_1, n_2)$ describes the self-reflection and refinement between node n_1 to node n_2. By treating each node as an independent candidate solution, we can assign evaluation scores using techniques such as self-consistency (Wang et al., 2023), model-generated critiques (Yu et al., 2025), or correctness-based metrics. These numerical scores enable direct quantitative comparison between nodes, allowing us to frame multi-step reasoning as a search problem whose objective is to efficiently expand nodes that are both promising and near a complete solution.

As previously mentioned, the self-reflection capabilities of small language models (SLMs) are generally less reliable than those of larger models. Consequently, relying on a single self-assessment as a search signal can be noisy. To address this, we design our node evaluation as a composite heuristic, in which self-reflection is strengthened and combined with metrics of tree-depth. In particular, each candidate node is assigned a heuristic score derived from the median of multiple independent self-evaluations, reducing variance. As overconfidence is common among small model self-evaluation (Tian et al., 2025), our proposed self-evaluation prompt is further framed as a harsh critic, encouraging conservative scoring and penalizing poor reasoning steps. Overall, this design yields a more reliable heuristic estimate for node evaluation, allowing the search to prioritize promising reasoning paths while suppressing the expansion of nodes that are incorrectly favored due to overconfident self-assessments.

3.3 A* Search

The A* search is a best-first algorithm that balances the cost to reach a node with a heuristic estimate of the cost to the goal (Hart et al., 1968), defined as:

$$n_{\text{next}} = \arg\min_{n \in \mathcal{N}_{\text{visited}}} f(n) = g(n) + h(n), \tag{1}$$

where $\mathcal{N}_{\text{visited}}$ is the collection of *visited* nodes in a tree search task, $g(n)$ is the cost of reaching node n, and $h(n)$ is the heuristic function defined by the self-critic / self-reflection of the language models. As suggested in (Hart et al., 1968), the path of the A* search is guaranteed to be optimal when the heuristic function $h(\cdot)$ is *admissible* (i.e., $h(\cdot)$ never overestimates the cost of reaching the goal).

In our setting, $g(n)$ represents the accumulated "cost" of reasoning steps taken to reach the current node. Intuitively, nodes that require many corrections or refinements from previous steps have higher $g(n)$, while nodes closer to a coherent and promising partial solution have lower $g(n)$. This prevents the search from blindly expanding nodes that appear promising according to the heuristic $h(n)$ but are built on long, error-prone chains of reasoning.

We use A* instead of purely greedy search because relying solely on the self-reflection heuristic $h(n)$ can be misleading: small language models (SLMs) may provide noisy or inconsistent evaluations. A* naturally balances the heuristic with the path cost $g(n)$, allowing the algorithm to favor nodes that are both promising and reachable with minimal accumulated error.

3.4 Adapting A* Search to Tree-based Reasoning

We reinterpret the A* search algorithm in the context of LLM-based reasoning by modeling the reasoning process as a search through a structured tree space of potential answers.

To navigate this space efficiently, we define a modified A* cost function that balances exploration and exploitation. We define g and h as:

$$g(n) = w \cdot \text{Distance}(n) \tag{2}$$

$$h(n) = 100 - \text{Reward}(n) \tag{3}$$

$$f(n) = g(n) + h(n) \tag{4}$$

where Distance(n) is the depth of the node from the root, encouraging broad exploration, and w controls the exploration-exploitation tradeoff. A larger w penalizes responses further down the tree, effectively promoting exploration over exploitation. A small w value has the inverse effect. Reward(n) is derived from correctness, self-consistency, or model-generated critiques using the same LLM.

As self-reflection is less reliable in SLMs, we sample and aggregate multiple self-evaluations to help stabilize the possibly noisy evaluation. This choice naturally reduces the variance caused by flawed single-step self-reflection, allowing for Reward(n), and in turn the heuristic $h(n)$, to be more reliable guides in the A* search.

This formulation promotes efficient traversal of the reasoning tree by balancing exploration and exploitation: nodes near the root encourage broader search, while high-reward nodes reflect promising or correct reasoning. As a result, search decisions are not driven solely by the model's potentially unreliable self-reflection but by a combination of aggregated reward signals and structural heuristics that together provide more robust guidance.

This setup enables the LLM to iteratively refine its response and solution by generating and evaluating child nodes, then selecting the next node to expand based on the lowest estimated cost $f(n)$.

3.5 Tree-Based Reasoning Procedure

We model SLM reasoning as a tree search problem, where nodes represent candidate answers and edges correspond to refinements. The search begins at a root node and iteratively generates new candidates through critique and revision, exploring the most promising paths in the reasoning space. All prompts used are provided in Appendix A.2.

Node Structure: Each node stores the question, candidate answer, critique, and reward. This reward is used in cost heuristic $h(n)$ to guide node selection, balancing exploration and exploitation.

Initialization: The root node is initialized by prompting the model to solve the problem using the phrase *"Let's think step by step."* We set the current node as root note, initialize a list to store all visited answers with their corresponding rewards, and maintain a separate list of nodes yet to be explored.

The the following process is then iterated until a pre-defined number of times.

Critique: The current node is critiqued by the same model, which provides detailed, constructive feedback on how to improve the answer while also highlighting what has been done correctly.

Self-Evaluation: After the critique, each node is assigned a reward score estimating the quality and completeness of the candidate answer, which is then converted into the heuristic cost $h(n)$ in the A* cost function. Specifically, the LLM is prompted with the original question, the candidate answer, and the preceding critique, and is asked to assign a numerical grade from 0 to 100 based on correctness, coherence, and completeness. We define this score

as $\text{Reward}(n)$ and use $h(n) = 100 - \text{Reward}(n)$ as the residual heuristic cost, so that candidates judged to be closer to a correct solution receive lower heuristic cost. This design is motivated by the admissibility principle in classical A* search: while we do not claim that the LLM-generated heuristic is formally admissible, we aim to make it conservative in the practical sense that incomplete or erroneous reasoning should not be assigned an artificially low residual cost. To this end, the model is first asked to critique the candidate before scoring it, encouraging the evaluator to identify local mistakes, missing steps, or unsupported conclusions before assigning a grade. To further reduce variance and mitigate overconfident single-shot evaluations, we query the LLM multiple times, e.g., three independent evaluations, and take the median of the resulting scores as $\text{Reward}(n)$. This critique-first and median-aggregated reward design enables more reliable comparisons between candidate reasoning nodes without requiring an external reward model, an additional verifier, or any training-time supervision for node evaluation. Importantly, neither the critique nor the self-evaluation step requires access to the ground-truth answer.

Tree Expansion: After incorporating the critique and reward scores, we generate two child nodes by prompting the model to revise the parent node's answer. Each child node represents an attempted refinement or improvement over its parent, explicitly addressing the feedback provided in the parent's critique. Distinct children can be obtained by sampling multiple outputs from the LLM (using temperature sampling) to introduce variation between the refinements. For each child node, we then generate a new critique and self-evaluation score. Both child nodes are added to the list of nodes yet to be explored. The next node to expand is selected using the A* cost function, choosing the node with the lowest $f(n)$. This node becomes the current node and is appended, along with its score, to the list of all generated answers. Two child nodes are generated per iteration for simplicity and to maintain a tractable search. This strikes a balance between exploring multiple candidate solutions and limiting computational cost.

Final Answer Selection: The search iterates up to a fixed number of steps (`max_iterations`) However, if any candidate achieves a self-evaluation score above a pre-defined threshold (e.g., 95/100), the search terminates early. The final answer is selected as the candidate with the highest self-evaluation score among all generated nodes, ensuring both efficiency and high-quality solutions. A formal write-up is provided in Algorithm 1.

Algorithm 1 Test-time A* Search

Require: Problem prompt p (e.g., "What is 4 times 3?")
Ensure: Answer with maximum LLM evaluation score

$root \leftarrow$ LLM(p with "Let's think step by step")
$AS \leftarrow \{\}$ ▷ Stores $(answer, critique, score)$
Generate critique for $root$ and evaluate score
Add $(root, critique, score)$ to AS
for $i = 1$ **to** $max_iterations$ **do**
  Compute $f(n) = w \cdot \text{Distance}(n) + h(n)$ for unvisited leaves
  $n \leftarrow$ leaf with minimum $f(n)$
  Generate children of n
  for each child c **do**
    Prompt LLM with parent answer + critique
    Generate critique and score for c
    Add $(c, critique, score)$ to AS
  end for
end for
return answer in AS with maximum score

4 Experiments

4.1 Models

Our work focuses on SLMs due to their reduced memory footprint and low inference cost, which make mathematical reasoning available without prohibitive GPU requirements. Therefore, we prioritize models in the 1–8B parameter range that typically require only 2–16 GB of VRAM, allowing deployment on consumer-grade GPUs or even embedded devices(Subramanian et al., 2025). By prioritizing applicability over scale, we aim to highlight the broad usage of LLMs in realistic, resource-constrained environments rather than maximizing performance on specialized hardware.

To align with our focus on wide deployment, we conduct all experiments using open-source models from two major families: Llama and Qwen. Specifically, we use Llama-3.2-1B (Meta AI, 2024b), Qwen3-4B (Base) (Yang et al., 2025a), Llama-3-8B (Grattafiori et al., 2024), and Llama-3.1-8B (Meta AI, 2024a) to represent widely adopted general-purpose SLMs. In addition, we include

Name	GSM8K	MATH500	AIME24	MATH401
Llama Models				
Llama-3.2-1B	39.7	29.8	3.3	58.5
w/ TTA*	51.2 (+29.0%)	39.4 (+32.2%)	6.7 (+103%)	68.1 (+16.4%)
Llama-3-8B	75.4	26.2	3.3	62.8
w/ TTA*	87.1 (+15.5%)	44.2 (+68.7%)	6.7 (+103%)	78.3 (+24.7%)
Llama-3.1-8B	77.2	52.2	3.3	68.8
w/ TTA*	90.2 (+16.8%)	66.6 (+27.6%)	10.0 (+203%)	80.1 (+16.4%)
Qwen Models				
Qwen3-4B	84.8	53.9	23.3	62.1
w/ TTA*	95.9 (+9.5%)	73.4 (+36.2%)	30.0 (+28.8%)	78.3 (+26.1%)
Qwen2.5-Math-7B	73.3	46.8	6.7	65.5
w/ TTA*	87.1 (+18.9%)	74.2 (+58.5%)	10.0 (+49.3%)	78.9 (+20.5%)

Table 1: Reasoning accuracy for TTA*. Relative improvements over zero-shot CoT (Wei et al., 2023) are reported in parenthesis.

Qwen2.5-Math-7B (Yang et al., 2024), a math-specialized model. We choose these models for their open-source availability and ease of deployment, making them representative examples of off-the-shelf SLMs. Together, these models offer a diverse but practical testbed for evaluating TTA*.

For all models, we set the decoding temperature to 0.3 to encourage mostly deterministic outputs while still allowing limited diversity, following prior work on reasoning evaluations.

4.2 Datasets

We evaluate on four mathematics benchmarks spanning a range of difficulty:

GSM8K (Cobbe et al., 2021): Grade-school word problems requiring multi-step arithmetic and commonsense reasoning.

MATH500 (Lightman et al., 2023): Competition-style problems that require high school-level reasoning and complex logic.

AIME (2024) (Mathematical Association of America, 2024): Problems from the American Invitational Mathematics Examination, known for their difficulty and requirement of creative algebraic problem-solving.

MATH401 (Yuan et al., 2023): A collection of 401 problems covering arithmetic and elementary functions (e.g., trigonometry, logarithms) designed to test reliability on fundamental operations.

This collection of datasets allows us to evaluate models on both basic reasoning and advanced problem-solving, while maintaining focus within the mathematical reasoning domain.

4.3 Evaluation

Table 1 and Figure 2 summarize the performance of our experiments across four benchmarks: GSM8K, MATH500, AIME (2024), and MATH401. Overall, TTA* consistently improves reasoning accuracy across both Llama and Qwen model families.

On the **AIME (2024)** benchmark, which is the most challenging, relative improvements are particularly notable. For example, Llama-3.1-8B improves from 3.3% to 10.0% (+6.7 points, +203% relative), while Qwen2.5-Math-7B improves from 6.7% to 10.0% (+3.3 points, +49.3%). These results indicate that TTA* effectively enhances structured reasoning even on competitive problems.

On **MATH500**, TTA* delivers substantial gains for both model families, with relative improvements ranging from +27.6% to +71.0%. This demonstrates that structured search benefits scale with problem complexity, allowing small and mid-sized LLMs to achieve competitive performance.

On more standard benchmarks like **GSM8K** and **MATH401**, performance gains are consistent and reliable. For GSM8K, relative improvements range from +9.5 to +18.9, while on MATH401 (arithmetic and elementary functions), relative gains range from +16.2 to +26.1. This suggests that TTA* not only improves advanced reasoning but also reinforces fundamental problem-solving skills.

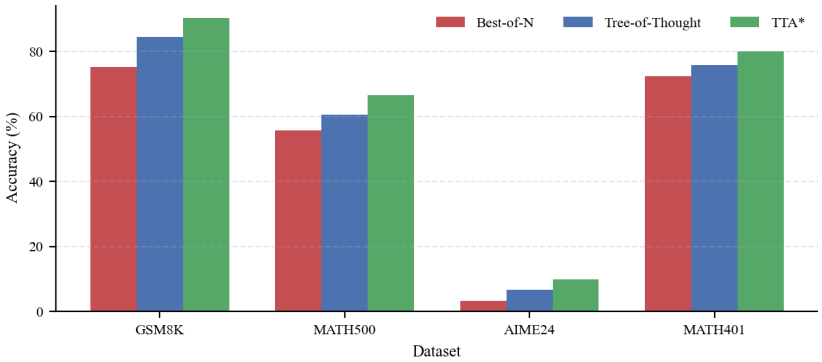

Figure 2: Performance comparison of TTA* against different test-time scaling methods on mathematical reasoning benchmarks for Llama-3.1-8B.

Overall, these results demonstrate that TTA* consistently improves reasoning performance across a wide range of model scales and benchmarks, without requiring additional model training. The improvements suggest that TTA* mitigates general reasoning limitations in SLMs rather than exploiting model-specific idiosyncrasies. Gains are observed across both general-purpose and specialized SLMs, small and mid-sized architectures, and tasks of varying difficulty. All baselines are evaluated against zero-shot chain-of-thought (Wei et al., 2023). Full prompts and an ablation study are provided in the Appendix.

4.4 Comparing with Test-Time Scaling Methods

We compare TTA* against two widely used test-time scaling strategies: Best-of-N (BoN) (Huang et al., 2025a) sampling and Tree-of-Thoughts (ToT) (Yao et al., 2023).

Dataset	Best-of-N	Tree-of-Thought	TTA*
GSM8K	75.3	84.4	**90.2**
MATH500	55.8	60.5	**66.6**
AIME24	3.33	6.67	**10.0**
MATH401	72.4	75.9	**80.1**

Table 2: Performance comparison of test-time scaling methods on four datasets using LLaMA-8B. Bold indicates the best method per dataset.

Best-of-N sampling improves accuracy by generating multiple independent solutions and selecting the highest-scoring output (Huang et al., 2025a). Tree-of-Thoughts (ToT) structures multi-step reasoning as an explicit tree search over intermediate reasoning states, using self-evaluation to guide expansion and pruning of candidate thought paths.

Leveraging the Llama-3.1-8B base model, we compare TTA* against Best-of-N and Tree-of-Thoughts, and find that our method consistently outperforms these test-time scaling baselines. To ensure a fair comparison, we match the maximum number of inference steps `max_iterations` across all methods. Our results are shown in Table 2

5 Conclusion

In this paper, we introduced Test-Time A* Search (TTA*), a novel framework that equips language models with structured, iterative reasoning through tree-based search.

Our work is motivated by the critical issue of providing smaller, yet more accessible models with advanced reasoning abilities. This emphasis is crucial for promoting the eventual usage of SLMs for users lacking access to state-of-the-art hardware. Domains of interest include medical diagnosis in low-resource hospitals and offline decision support in remote environments, where deploying massive LLMs is infeasible, making it essential to support advanced reasoning on limited compute.

Our experiments show that TTA* consistently outperforms zero-shot chain-of-thought on multiple mathematical reasoning tasks, highlighting the potential of combining classical search with modern LLMs. This promising results also implie the broader applicability of TTA* to areas such as commonsense reasoning, scientific discovery, or other complex tasks remains unexplored.

Future work should expand evaluation beyond mathematics and integrate TTA* into more practical environments to test real-world reasoning capabilities. Our current version of TTA* is a step toward making reasoning accessible not only to researchers with large compute budgets, but also to users operating under constrained conditions.

Limitations

While TTA* demonstrates consistent improvements across mathematical reasoning benchmarks, several limitations should be acknowledged.

First, we currently focus on mathematical reasoning tasks. It would be possible to examine whether TTA* generalizes to other domains such as common sense reasoning, scientific QA, or code generation is still an open question, and we make no claims about the performance of TTA* outside the mathematical setting.

Second, the inference time complexity may be affected by the design of better heuristics in TTA* and wall-clock inference time scales with the number of iterations and children per expansion.

Third, the self-evaluation component relies on the model's own critiques, which are known to be noisy and overconfident in smaller models. Although our median aggregation and harsh critic prompt partially mitigate this, the reliability of the heuristic is still limited by the base model's evaluative capacity.

-

References

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. *Preprint*, arXiv:2402.00157.

Mohammed Al-Garadi, Tushar Mungle, Abdulaziz Ahmed, Abeed Sarker, Zhuqi Miao, and Michael E. Matheny. 2025. Large language models in healthcare. *Preprint*, arXiv:2503.04748.

Anthropic. 2026. Introducing Claude Opus 4.6. https://www.anthropic.com/news/claude-opus-4-6. Accessed: 2026-03-24.

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. 2025. Small language models are the future of agentic ai. *Preprint*, arXiv:2506.02153.

Jing Bi, Yuting Wu, Weiwei Xing, and Zhenjie Wei. 2025. Enhancing the reasoning capabilities of small language models via solution guidance fine-tuning. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9074–9084, Abu Dhabi, UAE. Association for Computational Linguistics.

Yihang Cheng, Lan Zhang, Junyang Wang, Mu Yuan, and Yunhao Yao. 2024. Remoterag: A privacy-preserving llm cloud rag service. *Preprint*, arXiv:2412.12775.

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

Quy-Anh Dang and Chris Ngo. 2025. Reinforcement learning for reasoning in small llms: What works and what doesn't. *Preprint*, arXiv:2503.16219.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, Olli Järviniemi, Matthew Barnett, Robert Sandler, Matej Vrzala, Jaime Sevilla, Qiuyu Ren, Elizabeth Pratt, Lionel Levine, Grant Barkley, and 5 others. 2024. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *Preprint*, arXiv:2411.04872.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

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. *Preprint*, arXiv:2501.04519.

Peter Hart, Nils Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

Audrey Huang, Adam Block, Qinghua Liu, Nan Jiang, Akshay Krishnamurthy, and Dylan J. Foster. 2025a. Is best-of-n the best of them? coverage, scaling, and optimality in inference-time alignment. *Preprint*, arXiv:2503.21878.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025b. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Kyungjae Lee, Dasol Hwang, Sunghyun Park, Youngsoo Jang, and Moontae Lee. 2024. Reinforcement learning from reflective feedback (rlrf): Aligning and improving llms via fine-grained self-reflection. *Preprint*, arXiv:2403.14238.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy

Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. *Preprint*, arXiv:2206.14858.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Bingbin Liu, Sebastien Bubeck, Ronen Eldan, Janardhan Kulkarni, Yuanzhi Li, Anh Nguyen, Rachel Ward, and Yi Zhang. 2023. Tinygsm: achieving >80% on gsm8k with small language models. *Preprint*, arXiv:2312.09241.

Jingyuan Ma, Damai Dai, Zihang Yuan, Rui li, Weilin Luo, Bin Wang, Qun Liu, Lei Sha, and Zhifang Sui. 2025. Large language models struggle with unreasonability in math problems. *Preprint*, arXiv:2403.19346.

Mathematical Association of America. 2024. 2024 aime i and ii problems.

Meta AI. 2024a. Introducing llama 3.1: Our most capable models to date. Accessed: 2025-07-16.

Meta AI. 2024b. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models. Published September 25, 2024.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2025. Routellm: Learning to route llms with preference data. *Preprint*, arXiv:2406.18665.

OpenAI. 2025. Introducing gpt-5. Accessed on 2025-08-11.

Tanjil Hasan Sakib, Md. Tanzib Hosain, and Md. Kishor Morol. 2025. Small language models: Architectures, techniques, evaluation, problems and future adaptation. *Preprint*, arXiv:2505.19529.

Si Shen, Fei Huang, Zhixiao Zhao, Chang Liu, Tiansheng Zheng, and Danhao Zhu. 2025. Long is more important than difficult for training reasoning models. *Preprint*, arXiv:2503.18069.

Shreyas Subramanian, Vikram Elango, and Mecit Gungor. 2025. Small language models (slms) can still pack a punch: A survey. *Preprint*, arXiv:2501.05465.

Zailong Tian, Zhuoheng Han, Yanzhe Chen, Haozhe Xu, Xi Yang, Richeng Xuan, Houfeng Wang, and Lizi Liao. 2025. Overconfidence in llm-as-a-judge: Diagnosis and confidence-driven solution. *Preprint*, arXiv:2508.06225.

Shen Wang, Tianlong Xu, Hang Li, Chaoli Zhang, Joleen Liang, Jiliang Tang, Philip S. Yu, and Qingsong Wen. 2024. Large language models for education: A survey and outlook. *Preprint*, arXiv:2403.18105.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Xue Wu and Kostas Tsioutsiouliklis. 2024. Thinking with knowledge graphs: Enhancing llm reasoning through structured data. *Preprint*, arXiv:2412.10654.

Yuchen Xia, Jiho Kim, Yuhan Chen, Haojie Ye, Souvik Kundu, Cong Hao, and Nishil Talati. 2024. Understanding the performance and estimating the cost of llm fine-tuning. *Preprint*, arXiv:2408.04693.

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P. Lillicrap, Kenji Kawaguchi, and Michael Shieh. 2024. Monte carlo tree search boosts reasoning via iterative preference learning. *Preprint*, arXiv:2405.00451.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

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. *Preprint*, arXiv:2409.12122.

Wang Yang, Xiang Yue, Vipin Chaudhary, and Xiaotian Han. 2025b. Speculative thinking: Enhancing small-model reasoning with large model guidance at inference time. *Preprint*, arXiv:2504.12329.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

Yue Yu, Zhengxing Chen, Aston Zhang, Liang Tan, Chenguang Zhu, Richard Yuanzhe Pang, Yundi Qian, Xuewei Wang, Suchin Gururangan, Chao Zhang, Melanie Kambadur, Dhruv Mahajan, and Rui Hou. 2025. Self-generated critiques boost reward modeling for language models. *Preprint*, arXiv:2411.16646.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. 2023. How well do large language models perform in arithmetic tasks? *Preprint*, arXiv:2304.02015.

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

Wenqi Zhang, Yongliang Shen, Linjuan Wu, Qiuying Peng, Jun Wang, Yueting Zhuang, and Weiming Lu. 2024b. Self-contrast: Better reflection through inconsistent solving perspectives. *Preprint*, arXiv:2401.02009.

Yong Zhang, Bingyuan Zhang, Zhitao Li, Ming Li, Ning Cheng, Minchuan Chen, Tao Wei, Jun Ma, Shaojun Wang, and Jing Xiao. 2025. Self-enhanced reasoning training: Activating latent reasoning in small models for enhanced reasoning distillation. *Preprint*, arXiv:2502.12744.

Shuang Zhou, Zidu Xu, Mian Zhang, Chunpu Xu, Yawen Guo, Zaifu Zhan, Yi Fang, Sirui Ding, Jiashuo Wang, Kaishuai Xu, Liqiao Xia, Jeremy Yeung, Daochen Zha, Dongming Cai, Genevieve B. Melton, Mingquan Lin, and Rui Zhang. 2025. Large language models for disease diagnosis: A scoping review. *Preprint*, arXiv:2409.00097.

Xunyu Zhu, Jian Li, Can Ma, and Weiping Wang. 2024. Improving mathematical reasoning capabilities of small language models via feedback-driven distillation. *Preprint*, arXiv:2411.14698.

A Appendix

A.1 Experimental Configurations

Datasets and models. Our experiments used publicly available benchmarks: GSM8K, MATH500, MATH401, and the AIME (2024) problems. Models are used off-the-shelf; the exact model families and identifiers used in the paper are listed in the repository. Our code loaded models from the Hugging Face Hub.

Hyperparameters

- Decoding temperature: 0.3 (used in all reported runs).
- Children per expansion: 2 (two child nodes generated for each expanded node).
- Self-evaluation samples per node: 3 independent evaluations (scores averaged to compute the node reward).
- A* path-cost weight w: default value used is 3.
- Maximum iterations: defaulted value used is 8.

Evaluation and scoring. To maximize the reliability of our experimental results, we used GPT-4 as our "grader." After each experiment, GPT-4 computed accuracy on each benchmark using exact-match scoring rules.

Licensing and model access. Code in the repository is released under the MIT license.

A.2 Prompts

Prompt for Critique

Question: question
Answer: answer
Please provide detailed constructive criticism, yet highlight what is already correct. Point the student in the right direction. Do not solve the problem.
Provide a grade (out of 100) in the format 'Grade: xx'.

Prompt for Generating Child Nodes

Question: question
Previous Answer: previous answer
Critique: critique
Given the feedback above, please try to solve the original problem again, step by step.

A.3 Ablation Studies

To understand the contribution of each design choice in TTA*, we conduct ablation studies on GSM8K using Llama-3.1-8B. Unless otherwise stated, we use the default configuration: 8 iterations, 2 children per expansion, depth penalty weight $w = 3$, and temperature $T = 0.3$.

Number of Iterations. Increasing the number of search iterations from 2 to 8 yields a steady improvement from 80.1% to 90.2%, confirming that additional search budget directly translates to better reasoning quality over the range tested. The gains are roughly linear, suggesting that the A* heuristic continues to find productive refinements rather than saturating early.

Children per Expansion. Expanding more children per node (from 2 to 6) provides a modest improvement from 90.2% to 91.1%. While additional children increase diversity, the marginal gain diminishes quickly, indicating that two children already capture most of the useful variation. We retain two children as the default to balance accuracy

Hyperparameter	Value	GSM8K Accuracy
Number of Iterations		
`max_iterations`	2	80.1
	4	84.5
	6	88.2
	8	**90.2**
Children per Expansion		
Children	2	90.2
	4	90.7
	6	**91.1**
Depth Penalty Weight w		
w	0	86.7
	3	**90.2**
	5	89.5
	10	83.9
Decoding Temperature		
T	0.1	86.4
	0.3	**90.2**
	0.7	90.1

Table 3: Ablation studies on GSM8K using Llama-3.1-8B. Bold indicates the best result per group. Default configuration: `max_iterations`$= 8$, children $= 2$, $w = 3$, $T = 0.3$.

with computational cost.

Depth Penalty Weight. The depth penalty weight w controls the exploration–exploitation tradeoff. Setting $w = 0$ removes the depth penalty entirely, reducing accuracy to 86.7%, which shows that the search becomes too greedy and over-relies on potentially noisy self-evaluation scores. Conversely, an overly aggressive penalty ($w = 10$) discourages deep refinement and degrades performance to 83.9%. The best balance is achieved at $w = 3$, confirming that moderate depth penalization is critical for guiding the search effectively.

Decoding Temperature. A temperature of $T = 0.3$ achieves the best accuracy (90.2%), closely followed by $T = 0.7$ (90.1%). A lower temperature of $T = 0.1$ reduces diversity among candidate refinements and leads to noticeably lower accuracy (86.4%). This suggests that some degree of stochasticity is important for generating meaningfully distinct child nodes, while excessive randomness ($T = 0.7$) provides no additional benefit.