Efficient Structured Reasoning Framework for SLMs via Dynamic Directed Acyclic Graph Construction

Anonymous ACL submission

Abstract

Small Language Models (SLMs) offer substantial cost efficiency but often struggle with complex reasoning tasks due to their limited parameter capacity. While methods such as Chain-of-Thought (CoT) and ReAct have proven effective for Large Language Models (LLMs), their effectiveness remains limited in the context of SLMs. Tree of Thoughts (ToT) enhances reasoning by enabling multi-path exploration, but poses practical limitations for SLMs due to its high computational overhead and limited extensibility for external tool integration. In this paper, we propose an Efficient Structured Reasoning Framework via Dynamic DAG Construction to enhance the reasoning capabilities of SLMs. Inspired by cognitive insights, the framework modularizes human problem-solving into "Atomic Thinking" components and dynamically reassembles them into a task-specific Directed Acyclic Graph (DAG), effectively pruning the reasoning search space. In addition, the framework compensates for the inherent limitations of SLMs by augmenting them with external knowledge through a modular architecture that supports node-level tool integration. Applied to 20B-scale SLMs, our framework achieves GPT-4o-level performance across various benchmarks with an average of seven model calls per problem, and outperforms ToT in both token efficiency and inference speed under equal accuracy. The code is available at : an anonymous repository.

1 Introduction

Recent research has demonstrated human-level language understanding and reasoning capabilities through LLMs like GPT-4 (OpenAI, 2023; Bubeck et al., 2023). However, these massive models are accompanied by practical constraints, including immense computational requirements (FLOPs), high-cost GPU infrastructure, and significant inference latency (Kaplan et al., 2020). Given these constraints, and the increasing demand for on-device AI and real-time inference systems, SLMs are gaining traction as a practical alternative, offering markedly improved computational efficiency compared to LLMs (Liu et al., 2024b; Erdogan et al., 2024; Chen et al., 2025a; Belcak et al., 2025). Nevertheless, SLMs often struggle to achieve sufficient performance on high-level reasoning tasks such as mathematical problem solving or formal logical reasoning. Previous work has characterized these advanced reasoning capabilities as emergent Abilities, which appear only after a model reaches a critical scale. This has led to the conventional understanding that SLMs, which fall below this critical threshold, lack the structural capacity to perform complex reasoning (Wei et al., 2022a; Liu and Low, 2023).

Conversely, recent research trends challenge this conventional view and suggest new possibilities. In particular, it has been shown that increasing inference-time compute, rather than merely scaling up model parameters, can lead to consistent improvements in performance (Snell et al., 2024; Wang et al., 2023; Brown et al., 2024). This paradigm, collectively referred to as Test-time Scaling (TTS) (Zhang et al., 2025), dynamically allocates additional computational resources at inference time in response to the difficulty or reasoning complexity of a given problem, without requiring further model training.

Within this context, the field of LLM research has explored a wide range of approaches for solving complex problems, spanning from prompting strategies to structured search frameworks. Chain-of-Thought (Wei et al., 2022b) explicitly articulates intermediate reasoning steps in natural language to enable step-wise problem solving. ReAct (Yao et al., 2023b) synergizes reasoning and acting to enhance the model's ability to utilize external tools. Tree of Thoughts (Yao et al., 2023a; Long, 2023) enables exploration over multiple reasoning paths by representing alternative hypotheses, and Graph of Thoughts (GoT) (Besta et al., 2024) extends this capability by structuring the reasoning process as a graph, allowing for the aggregation and iterative refinement of intermediate thoughts.

Specifically, ToT is considered to be a robust test-time scaling approach that significantly enhances model capabilities by enabling multi-path exploration and backtracking (Yao et al., 2023a; Long, 2023). However, direct application of ToT to SLMs presents two fundamental challenges: scalability and efficiency. First, ToT primarily relies on the model's internal knowledge, operating within a closed-system reasoning approach. This inherently limits its modular extensibility for active integration of external tools, such as search engines or calculators. Second, ToT performs exploration based on the language model's probabilistic sampling, consuming tens to hundreds of times more tokens than a single inference (Yao et al., 2023a).

To address these issues, this work proposes an efficient structured test-time scaling framework that explicitly avoids indiscriminate reasoning expansion. Rather than relying on probabilistic sampling-based exploration as in ToT, we leverage prior knowledge grounded in cognitive science literature. Specifically, we define the elementary cognitive operations that constitute human problem-solving as units of Atomic Thinking, and reorganize them as a dynamic DAG that flexibly supports both sequential and parallel reasoning. This structured reasoning design guides SLMs to follow a more efficient reasoning path that ensures logical coherence while minimizing unnecessary exploration costs. Furthermore, by modularizing each node as an independent functional unit, the framework facilitates seamless integration with external tools and provides a reasoning system that is extensible across diverse tasks, even under limited parameter capacity.

Our main contributions are as follows:

1. *Reasoning Enhancement for SLMs*: We propose a structured reasoning framework for SLMs that efficiently utilizes limited computational resources, and provides a JSON-based execution trace, ensuring high transparency and interpretability.

2. *Competitive Benchmark Performance*: Extensive evaluations show that a 20B-parameter SLM equipped with our framework achieves performance comparable to or surpassing GPT-4o (OpenAI, 2024), particularly on mathematical and causal reasoning tasks.

3. *Tool Scalability*: A modular node-based design allows flexible integration of external tools, effectively mitigating the inherent knowledge limitations of SLMs.

2 Method

We propose a reasoning framework for SLMs drawing on cognitive insights into human problem-solving, which constructs structured reasoning traces in the form of a DAG.

2.1 Cognitive Prior Knowledge and Atomic Thinking Library $\mathcal{U}$

To achieve optimized test-time scaling in SLMs, it is crucial to formalize core reasoning primitives that guide efficient inference while avoiding the inefficiencies of stochastic exploration. Drawing from cognitive science literature, we identified core reasoning patterns indispensable for human problem-solving and defined these as Cognitive Prior Knowledge. We then implemented this knowledge as Atomic Thinking units, represented as executable functions f.

The collection of these functions constitutes the Atomic Thinking Library $\mathcal{U}$, where each Atomic Thinking unit $f \in \mathcal{U}$ modularizes an abstract cognitive strategy into a concrete input–output schema. Such a unit can be implemented either as a system-prompt-based function for specific inferential tasks or as a code module encompassing API calls and computational logic for external tool integration. These units serve as independent nodes within a DAG, ensuring that the model operates within a validated reasoning framework rather than searching an unstructured space. The library $\mathcal{U}$ is categorized into five functional groups based on their cognitive roles in Table 1.

2.2 Reasoning Framework via Dynamic DAG Construction

This section details our framework's mechanism for constructing and executing task-specific DAG from library $\mathcal{U}$ in Figure 1.

2.2.1 Problem Analysis and DAG Construction

The type and complexity of the input X, a natural language query or problem instance, are analyzed to establish strategic planning for problem

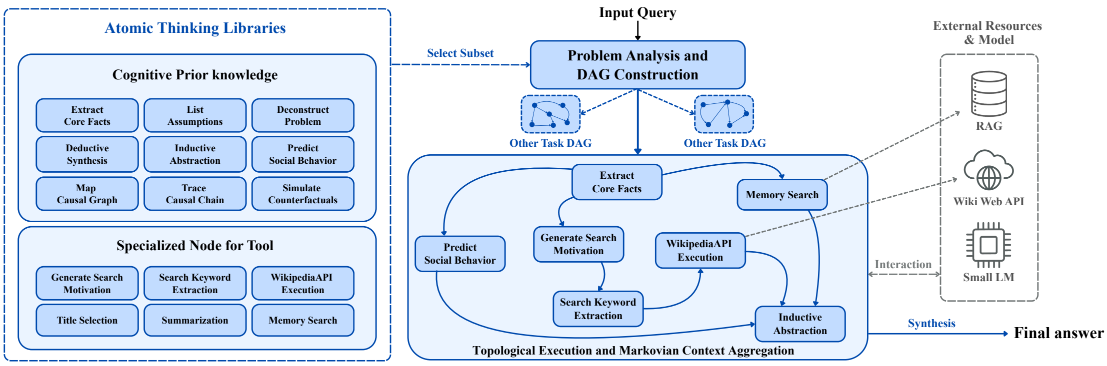

Figure 1: Our framework dynamically assembles task-specific Directed Acyclic Graph from the Atomic Thinking Library $\mathcal{U}$ and executes them through topological sorting. It employs a Markovian context aggregation mechanism to maintain logical consistency while ensuring token efficiency throughout extended inferential processes.

Group	Description
Structural Decomposition (G1)	Implements Pólya's (Pólya, 1945) *Understanding the Problem* and *Devising a Plan* phases as discrete prompts to prevent premature intuitive reasoning.
Directional Reasoning (G2)	Modularizes deduction and induction into discrete prompts based on Johnson-Laird (Johnson-Laird, 2010) to guide logical reasoning.
Contextual Reasoning (G3)	Drawing on Theory of Mind (ToM) from Kosinski (Kosinski, 2023) and Premack & Woodruff (Premack and Woodruff, 1978), it implicitly infers intent and context.
Causal Reasoning (G4)	Performs association, intervention, and counterfactual reasoning based on Pearl's *Hierarchy of Causality* (Pearl, 2009).
External Knowledge Retrieval (G5)	Consists of specialized nodes for external tool use (e.g., Web Search, RAG (Lewis et al., 2020)) to overcome parametric knowledge limitations.

Table 1: Categories of Atomic Thinking Library defined in our framework.

solving. At this stage, from an Atomic Thinking Library $\mathcal{U} = \{f^{(1)}, \ldots, f^{(m)}\}$, a subset $S \subseteq \mathcal{U}$ of functions required to solve the input X is selected. These functions are arranged according to logical dependencies to generate a Directed Acyclic Graph $G = (V, E)$. E is a set of directed edges representing logical dependencies between nodes, where $(v_i, v_j) \in E$ indicates that the execution of the subsequent node v_j conditionally depends on the output of the preceding node v_i. In this context, considering that the same type of thinking may be used multiple times depending on the context, each node $v_i \in V$ is defined as a tuple of Function and Reasoning Intent:

$$v_i = \langle f_i, r_i \rangle. \tag{1}$$

Here, $f_i \in S$ denotes the Atomic Thinking function to be executed, and r_i describes the reasoning intent required at that step. The detailed system prompt is provided in Appendix A.1.

2.2.2 Topological Execution and Markovian Context Aggregation

The generated graph G is converted into a linear execution sequence $T = (v_1, v_2, ..., v_N)$ through topological sorting. To maximize token efficiency, our framework adopts a Markovian approximation (Sutton and Barto, 2018), assuming that the reasoning at the current step depends only on the states of its immediate predecessor nodes rather than the entire history $H_{<t} = \{(v_\tau, o_\tau)\}_{\tau<t}$. Specifically, the Local Context C_t at execution time t is defined as the aggregation of outputs from the predecessor nodes directly connected to the current node v_t within the graph:

$$C_t = \oplus_{v_k \to v_t} o_k \tag{2}$$

where o_k denotes the execution result of node v_k, and $\oplus$ represents an aggregation operator that concatenates or synthesizes prior outputs into a coherent state. This process leverages SLM-based summarization to unify previous reasoning results into a single condensed contextual state. Accordingly, the execution of an Atomic Thinking unit and the probability of its output generation are approximated by assuming conditional independence as follows:

$$P(o_t \mid H_{<t}) \approx P(o_t \mid C_t, f_t, r_t) \qquad (3)$$

In this mechanism, C_t functions as a sufficient statistic that encapsulates the essential information of all preceding steps. By referencing only the localized state C_t instead of the full history, the framework prevents information overload while ensuring that past contextual information is condensed and transmitted. This serves as a core mechanism for maintaining the logical consistency even throughout extended inferential processes. Upon completion of all sequences, the framework synthesizes the accumulated reasoning steps to generate the final answer Y. The resulting full reasoning trace $H = \{(v_t, o_t)\}_{t=1}^{T}$ serves as an explainable reasoning trace that demonstrates the logical basis for the conclusion.

2.3 Modular Design for Scalable Tool Integration

To overcome the limitations of internal parametric knowledge and augment reasoning with external information, our framework integrates Specialized Nodes for Tools. In particular, we ensure the general scalability of the framework by varying the configuration of necessary tool nodes according to the specific requirements of each task.

2.3.1 Web Search Node for Open-Domain Reasoning

When up-to-date information or external fact-checking is required, the task is processed by constructing a Sequential DAG consisting of five specialized nodes to maximize search quality and relevance: Generate Search Motivation, Search Keyword Extraction, Search Execution (Wikipedia API), Title Selection, and Summarization. As all intermediate outputs are fully recorded in the reasoning trace, the search process is embedded within the reasoning trace, rather than treated as an independent pipeline, with context shared across preceding and subsequent nodes. The detailed functionalities of these nodes are described in Appendix A.2.

2.3.2 Memory Retrieval Node for Long-Context and Recall

For memory recall tasks that require retrieving specific historical information, the Atomic Thinking Library $\mathcal{U}$ is configured to include a Memory Search Node. This node adopts a Retrieval-Augmented Generation approach (Lewis et al., 2020) to overcome the limitations of the model's finite context window. Upon node execution, the input query is transformed into a high-dimensional vector, and the top-k most semantically relevant knowledge chunks are retrieved from a pre-indexed embedding-based vector database. The retrieved information is immediately integrated into the generation input context, enabling the stored knowledge to be effectively utilized in the ongoing reasoning process.

3 Experiments

3.1 Experiment setup

Models. The proposed framework requires reliable instruction-following capability and structured JSON-based outputs. Based on prior findings that models below 10B parameters lack such capabilities (Liu et al., 2024a), we adopt gemma-3-27b-it-qat-q4_0-gguf (Gemma Team, 2025) and Mistral-Small-3.2-24B-Instruct (Mistral AI, 2025) as backbone SLMs. Mistral-Small-3.2-24B-Instruct is executed under a 4-bit quantization setting.

Benchmarks. To comprehensively evaluate the reasoning capabilities of the models, we utilize six diverse benchmarks. Mathematical reasoning is assessed using the test set of MATHQA (2,985 questions) (Amini et al., 2019) and AIME (Committee, 2025), which consists of 30 problems from a high-difficulty U.S. mathematics competition. Causal reasoning is evaluated with CounterBench (Chen et al., 2025b), which consists of 1,000 questions covering four types of counterfactual reasoning. For an in-depth assessment of commonsense and open-domain reasoning, we use StrategyQA dataset (Geva et al., 2021), which contains 2,290 questions, and the QA subset of CoLoTA (Toroghi et al., 2025), consisting of 1,739 questions. Finally, we evaluate long-term memory recall and long-context reasoning capabilities using LoCoMo (Maharana et al., 2024), a long-conversation benchmark, with the QA subset (ID 1; 199 questions).

Baseline. To validate our framework, we categorized the baselines into three groups. First, Standard Reasoning Strategies include Zero-shot (question-only input) and Chain-of-Thought Prompting (Wei et al., 2022b; Kojima et al., 2022). Second, we selected advanced frameworks such as ReAct (Yao et al., 2023b) and Tree of Thoughts (Yao et al., 2023a). All hyperparameters followed

Model	Method	MATHQA	AIME2025	CounterBench	StrategyQA	CoLoTA
Gemma	Zero-shot	0.3776	0.1777	0.5131	0.5642	0.4030
	CoT Prompting	0.3696	0.1444	0.5190	0.5577	0.3565
	Tree of Thoughts	**0.8526**	0.2502	**0.7590**	0.6520	0.5950
	ReAct	0.6191	0.0667	0.6637	0.6278	0.6130
	Ours	0.8411	**0.2670**	0.7380	0.6664	0.6222
	Ours (+Web)	0.8010	0.2333	0.7380	**0.7576**	**0.6504**
Mistral	Zero-shot	0.2178	0.0333	0.5440	0.5483	0.4687
	CoT Prompting	0.2916	0.0667	0.6400	0.6289	0.4903
	Tree of Thoughts	0.8110	0.0834	0.7020	0.6750	0.5555
	ReAct	0.3031	0.0333	0.6430	0.6149	0.5265
	Ours	0.8111	**0.1167**	**0.7520**	0.7485	0.5934
	Ours (+Web)	**0.8139**	**0.1167**	0.7450	**0.7524**	**0.6527**

Table 2: Main Results: Performance Comparison. The best scores are in **bold**, second-best scores are underlined.

	MATHQA	AIME2025	CounterBench	StrategyQA	CoLoTA
GPT-4o(Nov 24)	0.3853	0.1000	0.6243	**0.7581**	**0.6600**
Gemma + Ours	**0.8411**	**0.2670**	0.7380	0.6664	0.6222
Gemma + Ours (+Web)	0.8010	0.2333	0.7380	0.7576	0.6504
Mistral + Ours	0.8111	0.1167	**0.7520**	0.7485	0.5934
Mistral + Ours (+Web)	0.8139	0.1167	0.7450	0.7524	0.6527

Table 3: Performance comparison with GPT-4o(Nov 24).

validated, competitive settings from existing literature to focus on structural comparison. Finally, GPT-4o(Nov,24) (OpenAI, 2024) was included to assess the performance gap against frontier models. Detailed hyperparameters and configurations for each baseline are provided in Appendix B.1.

Evaluation Metric. Model performance is reported as the mean accuracy (ACC) over two runs, and all experiments are performed on an RTX A6000 GPU.

3.2 Main Results

3.2.1 Comparison with Existing Reasoning Enhancement Methods

We evaluated our framework against four major methodologies—Zero-shot, CoT Prompting, ToT, and ReAct—using gemma-3-27b-it-qat-q4_0-gguf and Mistral-Small-24B-Instruct backbones in Table 2. Across all benchmarks, our framework consistently outperformed Zero-shot, CoT prompting, and ReAct. Notably, on MATHQA, Gemma-Ours (0.8411) and Mistral-Ours (0.8111) more than doubled their respective Zero-shot scores of 0.3776 and 0.2178.

Compared to ToT, our framework performs comparably on mathematical and causal benchmarks, while achieving an absolute improvement of over 10% on commonsense reasoning benchmarks. Unlike ToT, which is confined to the SLM's internal deliberation and self-evaluation, our framework explicitly interacts with and integrates external knowledge and contextual information into the reasoning process. Consequently, our approach demonstrates superior generalization not only in internal logical reasoning but also in tasks that require implicit world knowledge.

3.2.2 Comparison with Large-Scale LLM

We further analyze whether the proposed framework can achieve competitive performance against a frontier model in Table 3. Notably, our framework outperforms GPT-4o across several reasoning domains. On MATHQA, Gemma-Ours (0.8411) achieves more than a 2x improvement over GPT-4o (0.3853), and on the causal reasoning benchmark (CounterBench), it surpasses GPT-4o (0.7380 vs. 0.6243). On StrategyQA, Gemma-Ours (+Web) (0.7576) attains performance comparable to GPT-4o (0.7581). These results indicate that structured test-time scaling can effectively mitigate performance gaps arising from model scale differences, enabling smaller models to achieve competitive reasoning performance.

3.2.3 Efficiency Analysis

We evaluate the efficiency of our framework against ToT (Yao et al., 2023a) in terms of the average per-problem token usage, execution time, and number of LLM calls. In mathematical reasoning tasks, dif-

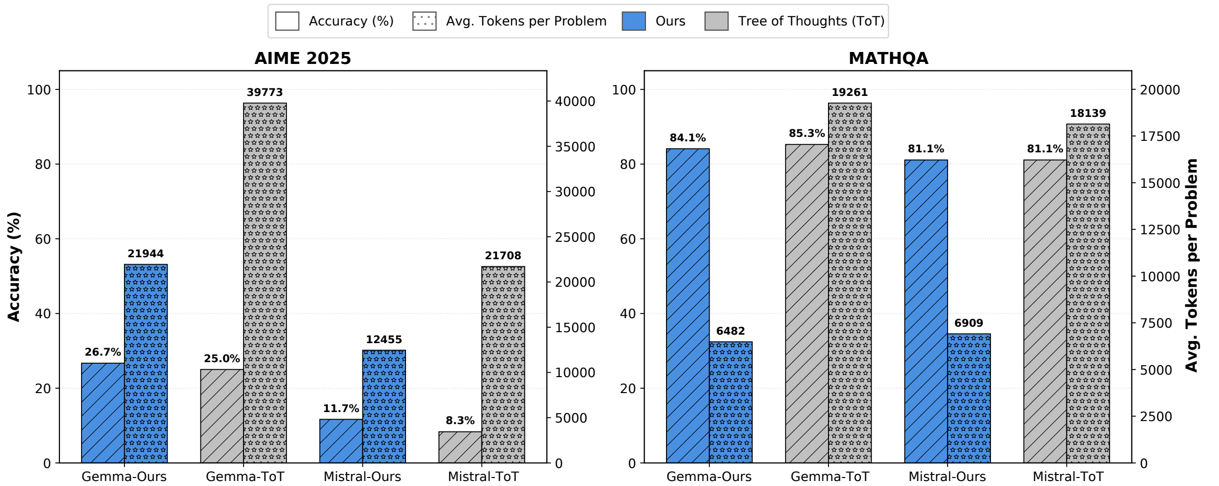

Figure 2: Accuracy and token efficiency comparison on mathematical benchmarks (AIME 2025 and MATHQA). Results are shown for Gemma and Mistral backbones using our method (Ours) and Tree of Thoughts. The left y-axis reports accuracy, and the right y-axis reports the average number of tokens per problem.

ferences in runtime and token consumption across reasoning frameworks tend to be particularly pronounced, as even problems with short final answers often require multi-step computations and explicit intermediate reasoning (Wei et al., 2022b). Indeed, ToT has been reported to incur substantially higher computational costs on mathematical tasks (Yao et al., 2023a).

Token efficiency. Across both Gemma and Mistral backbones, our framework consistently uses fewer average tokens per problem than ToT while achieving comparable accuracy levels in Figure 2.

Execution time. At comparable accuracy levels ($\pm 1\%$), on the AIME benchmark with the Gemma backbone, our framework requires an average of 239 seconds per problem, compared to 939 seconds for ToT, resulting in a 3.9× speedup. On the MATHQA benchmark with the Mistral backbone, our framework solves each problem in an average of 74 seconds, compared to 180 seconds for ToT, corresponding to a 2.4× speedup.

LLM calls. Our framework requires an average of 7–8 LLM calls per problem, compared to 25–26 calls for ToT.

Overall, these results show that our framework significantly reduces computational overhead while preserving strong reasoning performance. Detailed efficiency analyses and additional benchmark results are reported in Appendix B.3.

3.2.4 Scalability Analysis

We conduct an analysis on the long-term memory benchmark LoCoMo (Maharana et al., 2024).

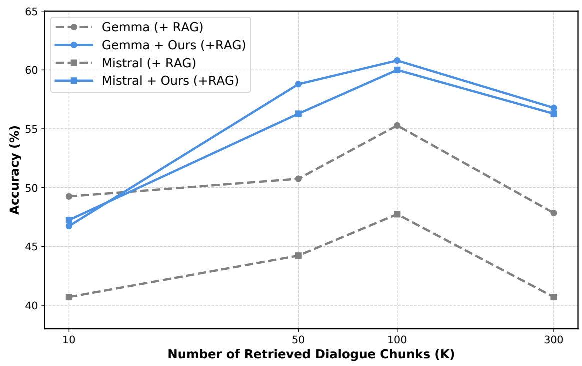

Figure 3: Scalability of memory retrieval under long-context conditions. Accuracy as a function of the number of retrieved dialogue chunks K.

As a preprocessing step, dialogue histories are first encoded into semantic embeddings using Sentence-BERT (all-MiniLM-L6-v2) (Reimers and Gurevych, 2019), based on which a high-dimensional vector database is constructed (Lewis et al., 2020). We then progressively increase the retrieval size $K \in 10, 50, 100, 300$ to analyze memory retrieval and reasoning performance under long-context conditions.

Figure 3 compares performance with and without the proposed framework as the amount of recalled context increases. When the context length exceeds a critical threshold ($K > 100$), models without the proposed framework exhibit substantial performance degradation, indicating difficulty in effectively integrating large volumes of retrieved information. In contrast, models equipped with our framework consistently maintain higher accuracy

Setting	MATHQA	CounterBench	StrategyQA
Zero-shot	0.3776	0.5131	0.5672
w/o G1	0.6693	0.7080	0.6279
w/o G2	0.8265	0.7305	0.6461
w/o G3	0.8294	0.7280	0.6397
w/o G4	0.8221	0.7230	0.6308
Full Framework	**0.8318**	**0.7380**	**0.6664**

Table 4: Impact of cognitive prior knowledge groups (G1–G4) on reasoning performance. The Full Framework achieves the best performance, while removing Structural Decomposition (G1) results in the largest performance drop across benchmarks.

under long-context settings. These results suggest that the proposed method goes beyond simply appending retrieved chunks; instead, it enables robust long-context reasoning and memory retrieval by structurally connecting salient information across extended dialogue contexts while progressively filtering out irrelevant noise. Detailed experimental settings and evaluation protocols are provided in Appendix B.4.

3.3 Ablation Study

We conduct ablation studies of the cognitive prior knowledge groups and the web search node.

3.3.1 Impact of Cognitive Prior Knowledge

To evaluate the contribution of cognitive prior knowledge, we use the Gemma backbone on three benchmarks—MATHQA, CounterBench, and StrategyQA—and systematically remove each of the four Atomic Thinking groups (G1–G4), measuring the resulting performance changes shown in Table 4. The results show that performance remains above the zero-shot baseline even when individual groups are removed. However, removing G1 (Structural Decomposition) leads to the largest performance drop across all benchmarks, suggesting that SLM failures are primarily driven by incorrect initial problem interpretation and strategy formulation. These results highlight that accurate problem definition and early-stage plan formulation are key determinants of model performance.

3.3.2 Impact of Web Search Node

We analyze the effect of incorporating the Web Search node using the results in Table 2. For mathematical and causal reasoning tasks, which are primarily driven by internal logical reasoning, the inclusion of the Web Search node has a limited impact or even leads to slight performance degradation, likely because the retrieved information introduces additional noise. In contrast, for commonsense reasoning tasks that require external knowledge, the Web Search node serves as a decisive factor for performance improvement. Gemma-Ours achieves a 9.1%p improvement on StrategyQA, and Mistral-Ours gains 5.9%p on CoLoTA. These results indicate that integrating web search into the reasoning process effectively compensates for the limited internal knowledge of SLMs.

3.4 Interpretability

We compare the summarized reasoning processes H generated by our framework with those produced by CoT and ToT on the same AIME 2025 problem in Figure 4. CoT (Wei et al., 2022b) represents reasoning as a single sequential text, which often leads to error propagation, where mistakes in intermediate steps carry over to later stages. As a result, it becomes difficult to trace individual step contributions or identify the root cause of errors. ToT (Yao et al., 2023a) partially alleviates this limitation by exploring multiple reasoning paths, but it does not explicitly structure inter-branch dependencies or global reasoning objectives. As a result, redundant computations over symmetric cases may occur, and inefficient backtracking is often required after entering incorrect paths.

In contrast, our framework explicitly defines reasoning objectives and inter-step dependencies through prior problem analysis and dynamic DAG construction. The reasoning process is composed of goal-oriented Atomic Thinking units, enabling direct monitoring of step-level success or failure and their downstream effects. This structured design allows the model's decision-making process to be explained in a human-interpretable manner. More detailed qualitative analysis of the summarized reasoning processes is provided in Appendix C.1.

3.5 Additional Analysis

In this section, we analyze the experimental results of existing reasoning enhancement methods on SLMs to identify which reasoning strategies are most effective. As shown in Table 2, CoT and ReAct yield limited improvements over Zero-shot prompting and underperform compared to ToT, indicating that methods effective for LLMs do not necessarily transfer reliably to SLMs.

Concretely, in CoT prompting, minor logical in-

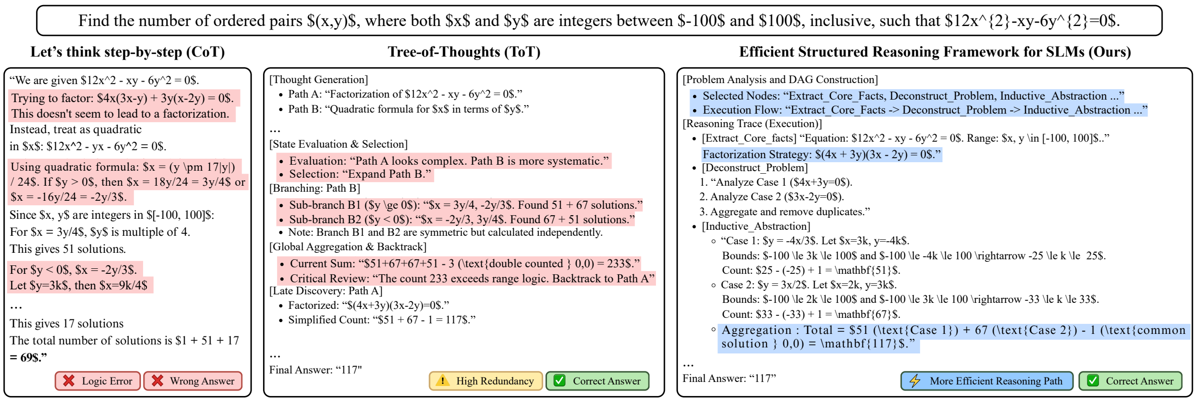

Figure 4: Comparison of summarized reasoning processes for the same AIME 2025 problem, highlighting structural differences between CoT, ToT, and our framework.

consistencies introduced in intermediate reasoning steps can be amplified throughout the reasoning chain (Wei et al., 2022b), leading to error propagation issues that are particularly pronounced in SLMs. ReAct also exhibits limitations when the model's intrinsic planning capability is insufficient, causing it to fall into inefficient action loops that prevent the development of deeper reasoning (Yao et al., 2023b). In contrast, ToT consistently achieves the strongest baseline performance across all benchmarks by decomposing problems into a tree, exploring multiple reasoning paths, and pruning unpromising branches (Yao et al., 2023a). These findings highlight structured reasoning as a key factor for effective reasoning in SLMs.

4 Related Work

LLM Reasoning Strategies. A variety of reasoning strategies have been proposed to enhance the reasoning performance of language models, including Chain-of-Thought (Wei et al., 2022b; Kojima et al., 2022), Tree of Thoughts (Yao et al., 2023a; Long, 2023), and Graph of Thoughts (Besta et al., 2024). While these approaches attempt to guide intermediate reasoning processes through explicit prompting or search-based exploration, they generally lack mechanisms to actively incorporate external tools or interactions with the environment into the reasoning loop (Yao et al., 2023a; Long, 2023). As a result, their applicability is limited in real-world scenarios where external information acquisition, such as search or API calls, is essential. To address these limitations, we propose a practical SLM reasoning framework that flexibly integrates tool usage into the core reasoning process.

Agentic AI Frameworks. Agentic AI research has increasingly focused on enabling LLMs to solve complex tasks through tool use. ReAct (Yao et al., 2023b) showed that coupling reasoning with action allows models to leverage external information and reduce hallucinations, while more recent work extends this line by enabling autonomous knowledge management and research behaviors (Wu et al., 2025). At the same time, the rise of on-device AI has driven active research on SLM-based agents (Erdogan et al., 2024; Chen et al., 2025a; Khiabani et al., 2025). However, these approaches mainly emphasize stable tool usage and accurate function calling under resource constraints, and remain limited in supporting high-level reasoning that fully exploits parametric knowledge. Building on their strengths in execution efficiency and tool robustness, we propose a framework that enhances the reasoning capability of SLMs for complex problem solving.

5 Conclusion

In this work, we propose an Efficient Structured Reasoning Framework for SLMs via Dynamic DAG Construction, inspired by cognitive insights. Experimental results show that, despite using only a 20B-scale model, our approach outperforms GPT-4o on mathematical and causal reasoning tasks, while achieving higher efficiency than Tree of Thoughts, thereby enabling practical test-time scaling. These results indicate that performance improvements in SLMs are driven primarily by structured reasoning, and further suggest that such reasoning frameworks can serve as a key foundation for future on-device AI systems aiming to achieve high-level intelligence under strict resource constraints.

6 Limitations

The proposed Efficient Structured Reasoning Framework achieves substantial reasoning improvements for SLMs. However, several limitations remain and warrant further investigation. One challenge is that the reasoning scope of the framework remains bounded by a predefined Atomic Thinking Library. Although these primitives cover broad reasoning patterns, they may lack the flexibility required in scenarios that demand highly fine-grained, domain-specific reasoning or previously unseen cognitive strategies. This constraint could potentially be mitigated by extending the framework toward the dynamic generation of Atomic Thinking patterns conditioned on the problem context. The framework also has limitations stemming from its plan-then-execute paradigm, which may restrict real-time adaptation or plan revision during the execution stage. This static nature presents an opportunity for incorporating meta-cognitive feedback mechanisms in future iterations. Such an extension would allow the framework to dynamically calibrate its reasoning path based on intermediate execution outcomes, mirroring the iterative self-correction characteristic of human problem-solving. We believe that transitioning toward such an adaptive architecture will be a crucial step in further enhancing the robustness of structured reasoning in SLMs.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 2357–2367. Association for Computational Linguistics.

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

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michał Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of Thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38(16), pages 17682–17690.

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

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of Artificial General Intelligence: Early experiments with GPT-4. *arxiv preprint*, arXiv:2303.12712.

Wei Chen, Zhiyuan Li, and Mingyuan Ma. 2025a. Octopus: On-device language model for function calling of software apis. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 3: Industry Track, pages 329–339. Association for Computational Linguistics.

Yuefei Chen, K.Singh Vivek, Jing Ma, and Ruxiang Tang. 2025b. CounterBench: A benchmark for counterfactuals reasoning in large language models. *arxiv preprint*, arXiv:2502.11008.

AIME 2025 Organizing Committee. 2025. AIME 2025 shared task dataset. 23rd International Conference on Artificial Intelligence in Medicine. Pavia, Italy. Accessed: 2025-12-26.

Lutfi Eren Erdogan, Nicholas Lee, Siddharth Jha, Sehoon Kim, Ryan Tabrizi, Suhong Moon, Coleman Richard Charles Hooper, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. 2024. TinyAgent: Function calling at the edge. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 80–88. Association for Computational Linguistics.

Gemma Team. 2025. Gemma 3: Technical report. *arxiv preprint*, arXiv:2503.19786.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Philip N. Johnson-Laird. 2010. Mental models and human reasoning. *Proceedings of the National Academy of Sciences of the United States of America*, 107(43):18243–18250.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arxiv preprint*, arXiv:2001.08361.

Yahya Sowti Khiabani, Farris Atif, Chieh Hsu, Sven Stahlmann, Tobias Michels, Sebastian Kramer, Benedikt Heidrich, M. Saquib Sarfraz, Julian Merten, and Faezeh Tafazzoli. 2025. Optimizing small language models for in-vehicle function-calling. *arxiv preprint*, arXiv:2501.02342.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2022)*, pages 22199–22213.

Michal Kosinski. 2023. Evaluating large language models in theory of mind tasks. *arxiv preprint*, arXiv:2302.02083.

Patrick Lewis, Ethan Perez, Aleksandr Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 2020)*, pages 9459–9474.

Tiedong Liu and Bryan Kian Hsiang Low. 2023. Goat: Fine-tuned llama outperforms gpt-4 on arithmetic tasks. *arxiv preprint*, arXiv:2305.14201.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, and 3 others. 2024a. AgentBench: Evaluating llms as agents. In *Proceedings of the International Conference on Learning Representations*.

Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, Liangzhen Lai, and Vikas Chandra. 2024b. MobileLLM: Optimizing sub-billion parameter language models for on-device use cases. In *Proceedings of the 41st International Conference on Machine Learning*, pages 32431–32454.

Jieyi Long. 2023. Large language model guided tree-of-thought. *arxiv preprint*, arXiv:2305.08291.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, volume 1: Long Papers, pages 13851–13870, Bangkok, Thailand. Association for Computational Linguistics.

Mistral AI. 2025. Mistral 3. Accessed: 2025-12-27.

OpenAI. 2023. GPT-4 technical report. *arxiv preprint*, arXiv:2303.08774.

OpenAI. 2024. Hello gpt-4o.

OpenAI. 2025. Introducing GPT-5. Accessed: 2025-12-27.

Judea Pearl. 2009. *Causality: Models, Reasoning, and Inference*, 2 edition. Cambridge University Press, Cambridge, UK.

David Premack and Guy Woodruff. 1978. Does the chimpanzee have a theory of mind? *Behavioral and Brain Sciences*, 1(4):515–526.

George Pólya. 1945. *How to Solve It*. Princeton University Press, Princeton, NJ, USA.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

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.

Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*, 2 edition. MIT Press, Cambridge, MA, USA.

Armin Toroghi, Willis Guo, and Scott Sanner. 2025. CoLoTa: A dataset for entity-based commonsense reasoning over long-tail knowledge. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3444–3454. Association for Computing Machinery.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of the International Conference on Learning Representations*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022)*.

Junde Wu, Jiayuan Zhu, Yuyuan Liu, Min Xu, and Yueming Jin. 2025. Agentic reasoning: A streamlined framework for enhancing llm reasoning with agentic tools. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, volume 1: Long Papers, pages 28489–28503. Association for Computational Linguistics.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. ReAct: Synergizing reasoning and acting in language models. In *Proceedings of the International Conference on Learning Representations*.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, Irwin King, Xue Liu, and Chen Ma. 2025. A Survey on Test-Time Scaling in Large Language Models: What, how, where, and how well? *arxiv preprint*, arXiv:2503.24235.

A Implementation Details

This appendix describes the implementation details of the proposed reasoning framework, including the system prompt for problem analysis and DAG construction and the design of specialized nodes for external tool integration.

A.1 System Prompt for Problem Analysis and DAG Construction

System Prompt

Task Description:
You are given the following problem: {{**input_query**}}
You have access to reasoning **Actions ONLY from the following Atomic Thinking Library**.

Atomic Thinking Library:

- **Extract_core_facts** – Identify only the essential facts and conditions of the problem.
- **List_assumptions** – Explicitly list all assumptions you're making.
- **Deconstruct_problem** – Solve the problem by breaking it into concrete, executable steps and carrying out each step in order.
- **Deductive_synthesis** – Start from the big picture and reason down to the details.
- **Inductive_abstraction** – Use small details to infer the overall structure or meaning.
- **Trace_causal_chain** – Lay out events in time order with cause-effect links.
- **Map_causal_graph** – Visualize causal links among key elements in a diagram.
- **Simulate_counterfactuals** – Imagine how outcomes change if conditions were different.
- **Predict_social_behavior** – Predict what a typical person would do in this situation.

Rules:

1. Only use actions from the list.
2. Select the actions that are most relevant for solving the problem.
3. Organize them as a **directed acyclic graph (DAG)** representing the reasoning flow.
4. Do NOT output any explanation or commentary. Only return pure, valid JSON.
5. Before finalizing, re-check that all rules are followed.

Output Format:

```
{
  "nodes": [{"name": "...", "reason": "..."}],
  "edges": [["node_a", "node_b"]]
}
```

In our framework, based on the analysis of the input problem, the required Atomic Thinking units are selected from the Atomic Thinking Library $\mathcal{U}$ and reassembled into a DAG that explicitly represents both the reasoning intent of each node and their logical dependencies. The Atomic Thinking Library $\mathcal{U}$ is a predefined set but can be flexibly extended by adding specialized nodes, such as web search nodes or memory retrieval nodes, to the library. This allows external tools to be integrated into the reasoning process without modifying the DAG construction logic. In addition, the JSON-based input–output interface enables stable and controllable execution throughout the reasoning pipeline while maintaining an interpretable structure.

A.2 Implementation Details of Web Search Nodes

When external information or fact-checking is required, the framework constructs a sequential DAG composed of multiple connected nodes rather than issuing a single search call. Each node corresponds to a specific step in the search process—such as search motivation generation, keyword extraction, search execution, document title selection and result summarization—forming a coherent reasoning flow in Tabel 5. In our implementation, the node responsible for search execution uses the Wikipedia API; however, this choice is interchangeable, and the search backend can be replaced by alternative search engines or knowledge sources without altering the overall reasoning structure. Except for the search execution node, all other nodes are implemented as prompt-based reasoning modules executed by the language model.

Importantly, the framework follows a design in which the search procedure is decomposed into multiple reasoning nodes, with intermediate outputs recorded in the reasoning trace and shared with subsequent nodes. Under this design, external knowledge access is integrated as a structural component of the overall reasoning process, rather than as a single tool invocation, thereby enhancing the model's reasoning capability.

Node Type	Description
Generate Search Motivation	Generates explicit motivation to align the search direction with the reasoning objective.
Search Keyword Extraction	Extracts core entities and queries based on the generated motivation.
Search Execution	Calls the Wikipedia API to acquire candidate documents.
Title Selection	Filters noise by selecting titles matching the search motivation.
Summarization	Refines selected text into essential context for the reasoning path.

Table 5: Specification and functional roles of specialized nodes for web search.

B Experimental Details

B.1 Experimental Setup

To ensure fair and reproducible comparison, all methods were evaluated under a unified inference setting with *max_new_tokens=3000* and *temperature=0.1* per call.

- **CoT Prompting.** CoT is a representative and effective prompting technique for enhancing the reasoning capabilities of large language models. However, since few-shot CoT (Wei et al., 2022b) requires intensive manual effort to craft task-specific exemplars, we adopt zero-shot CoT (Kojima et al., 2022) instead. By simply appending the prompt *"Let's think step by step."* to the input question, this approach improves reasoning performance without the need for explicit, human-authored demonstrations.

- **ReAct.** ReAct (Yao et al., 2023b) follows the implementation and prompt configuration described in the original paper. The use of the Wikipedia search engine is also consistent with the original setup. The maximum number of reasoning steps is set to 7, and if a final answer is not produced within the allowed steps, the most recently generated reasoning output was used to determine the final answer.

- **Tree of Thoughts.** For ToT (Yao et al., 2023a), hyperparameters were calibrated through preliminary experiments with the goal of preserving its original reasoning capability while maintaining a search scale comparable to that of the proposed framework. Breadth-First Search (BFS) is used as the exploration strategy. We set $N_{\text{gen}} = 3$, $N_{\text{eval}} = 1$, and $N_{\text{sel}} = 3$, with the tree depth fixed to 3. Prompts are adapted from those proposed in the original ToT work and tailored to the specific requirements of each benchmark.

- **Frontier Models.** For comparison with large-scale frontier models, GPT-4o(2024-11-20) (OpenAI, 2024) is used, and GPT-5 (Medium) (OpenAI, 2025) is additionally evaluated in Table 6.

B.2 Additional Comparison with GPT-5 (Medium)

As shown in Table 6, GPT-5 (Medium) achieves the highest overall performance across most benchmarks, reflecting the strong reasoning capabilities associated with large-scale parameterized models.

	MATHQA	AIME2025	CounterBench	StrategyQA	CoLoTA
GPT-5 (Medium)	**0.8885**	**0.3670**	0.6860	**0.7721**	**0.6768**
Gemma + Ours	<u>0.8411</u>	<u>0.2670</u>	0.7380	0.6664	0.6222
Gemma + Ours (+Web)	0.8010	0.2333	0.7380	<u>0.7576</u>	0.6504
Mistral + Ours	0.8111	0.1167	**0.7520**	0.7485	0.5934
Mistral + Ours (+Web)	0.8139	0.1167	<u>0.7450</u>	0.7524	<u>0.6527</u>

Table 6: Performance comparison with GPT-5. Despite the model size difference, our method surpasses GPT-5 on the CounterBench task. Best scores are shown in **bold**, and second-best scores are <u>underlined</u>.

Nevertheless, when the proposed framework is applied to a 20B-scale SLM, the resulting model attains performance comparable to GPT-5 (Medium) on multiple tasks, and notably surpasses it on the causal reasoning (CounterBench) benchmark.

These results demonstrate that structured reasoning design can effectively enhance the reasoning performance of SLMs by efficiently leveraging test-time computational resources, and suggest that the performance gap arising from differences in parameter scale can be partially mitigated without additional model scaling.

B.3 Detailed Efficiency Analysis

B.3.1 Comparison of Average Execution Time per Problem on Mathematical Benchmarks

This analysis focuses on the AIME and MATHQA benchmarks, where differences in computational cost across reasoning frameworks tend to be most pronounced in Table 7. On MATHQA, our framework is consistently faster than ToT by a margin of several minutes across both the Gemma and Mistral backbones. On AIME, a substantial execution time gap is observed with the Gemma backbone, whereas the difference is relatively small with the Mistral backbone. This reduced gap can be attributed to the fact that, under the Mistral setting, both methods exhibit low accuracy, which leads to early termination of the ToT search process before a valid solution is found.

Overall, these results indicate that the proposed framework can solve mathematical reasoning tasks with significantly lower computational cost than ToT when sufficient reasoning is required.

B.3.2 Comparison of Average LLM Calls per Problem across All Benchmarks

The average number of LLM calls per problem is compared in Figure 5. Across all benchmarks and both Gemma and Mistral backbones, the proposed framework consistently requires fewer LLM calls than ToT. In particular, our framework completes each problem with an average of 7–8 LLM calls, whereas ToT typically requires 25–26 calls.

Model	Benchmark	Ours (sec)	ToT (sec)
Gemma	AIME 2025	**239**	939
	MATHQA	**64**	474
Mistral	AIME 2025	**147**	154
	MATHQA	**74**	180

Table 7: Average execution time (seconds per problem) comparison between our framework and ToT on mathematical reasoning benchmarks. Lower is better; **bold** indicates the fastest method.

This reduction can be attributed to our structured DAG-based reasoning design, which minimizes unnecessary exploratory branching and enables more efficient use of inference-time computation. The consistency of this reduction across diverse benchmarks further indicates that the proposed framework exhibits robust efficiency across different tasks.

B.3.3 Comparison of Average Tokens per Problem across All Benchmarks

Figure 6 illustrates the trade-off between reasoning accuracy and computational cost across all benchmarks. The x-axis denotes task accuracy (%), while the y-axis represents the average number of tokens consumed per problem. Each marker corresponds to a benchmark result, where stars indicate the proposed framework and circles denote ToT. Lines connecting paired markers highlight the relative token reduction achieved by our method at comparable accuracy levels.

Across both Gemma and Mistral backbones, our method consistently attains comparable or higher accuracy while consuming fewer tokens, indicating improved reasoning efficiency.

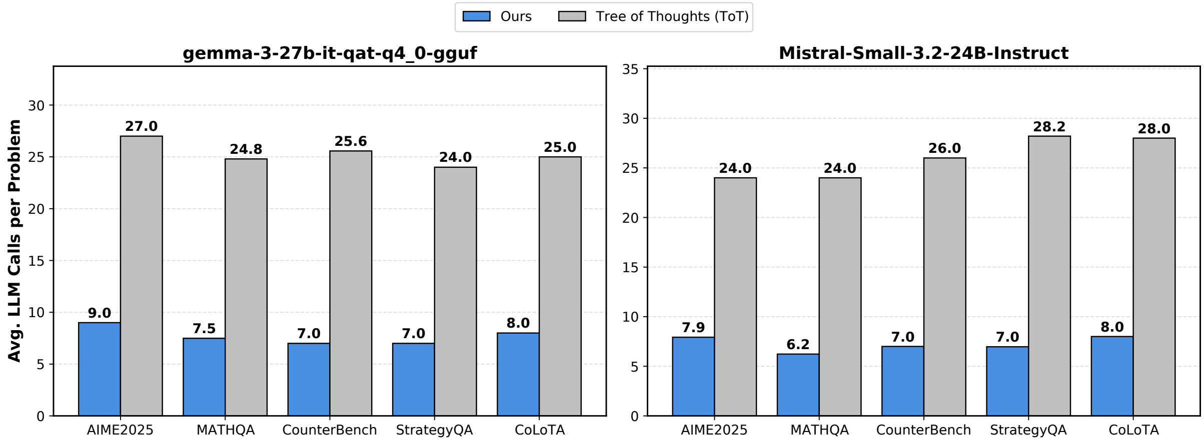

Figure 5: Average number of LLM calls per problem comparison between our framework and ToT across all benchmarks.

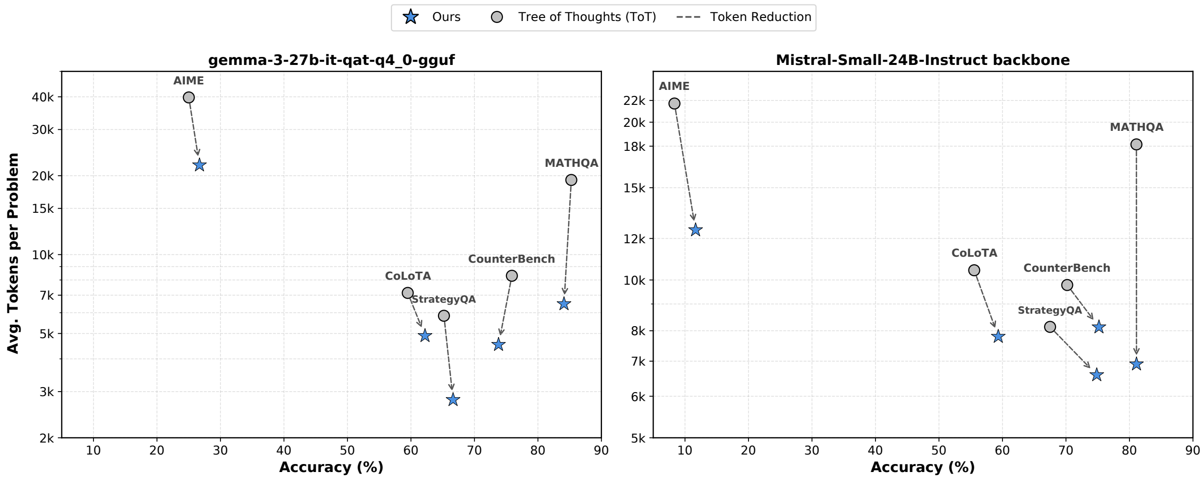

Figure 6: Comparison of average token usage per problem between our framework and ToT across all benchmarks.

B.4 Scalability Analysis Details

- **Preprocessing.** For the memory retrieval task, we use the dialogue history associated with ID 1 in the LoCoMo (Maharana et al., 2024). Each dialogue entry is normalized and concatenated into a single textual string following the format: `{speaker} : {text} [{date}]`. These dialogue strings are encoded into semantic embeddings using Sentence-BERT (all-MiniLM-L6-v2) (Reimers and Gurevych, 2019) and stored in a high-dimensional vector database.

- **Evaluation.** For question answering evaluation, we use the QA subset associated with ID 1, which consists of 199 questions. Due to the nature of long-context memory retrieval tasks, correct answers may be expressed in diverse linguistic variations while preserving the same semantic meaning. Accordingly, performance is measured by averaging the accuracy (ACC) scores assigned by three human experts, who judge whether each model output is semantically equivalent to the reference answer.

C Interpretability Details

C.1 Comparison of Summarized Reasoning Processes across CoT, ToT, and Our Framework

Figure 4 compares the summarized reasoning processes generated by CoT, ToT, and our framework on the same AIME 2025 problem.

CoT begins reasoning without an explicit structural analysis of the problem. Although it briefly attempts factorization, it quickly abandons this approach and reformulates the equation as a quadratic in x, which is then solved using the quadratic formula. During subsequent handling of integer constraints, case distinctions and boundary conditions are not systematically managed. Because the reasoning proceeds as a single linear text sequence, early strategic decisions are not revisited, causing intermediate judgment errors to propagate through later steps. As a result, portions of the solution space are omitted or miscounted, leading to an incorrect final answer.

ToT mitigates some of these issues by generating and exploring multiple reasoning paths in parallel, including both factorization-based and quadratic-form approaches. However, since inter-branch dependencies and global reasoning objectives are not explicitly structured, symmetric cases are repeatedly computed as independent branches. This redundancy leads to inflated intermediate solution counts, which are only corrected at a late stage after the aggregated result is found to violate basic range constraints. Although ToT ultimately discovers the correct factorized form, this occurs after substantial redundant exploration and inefficient backtracking.

In contrast, our framework performs explicit prior problem analysis and employs dynamic DAG construction to define reasoning objectives and inter-step dependencies before execution. The summarized reasoning process first identifies factorization as the core structural property of the equation and decomposes the problem into two independent cases accordingly. For each case, inductive abstraction is applied to derive a general solution form, followed by systematic boundary analysis to count valid integer solutions. Shared solutions are explicitly handled during aggregation, allowing the correct answer to be obtained in a single pass. Because each reasoning step is grounded in a predefined goal-oriented structure, step-level contributions and their downstream effects can be clearly traced, yielding human-interpretable explanations of the model's decision-making process.

C.2 Full Reasoning Trace Examples

Input Query :

p and q started a business investing rs. 48,000 and rs. 24,000 respectively. In what ratio will the profit earned after 2 years be divided between p and q respectively?

Full Reasoning Trace for a Math Task (MATHQA)


```
[
  {
    "step": 0,
    "actions": [
      {
        "node": "Extract_core_facts",
        "reason": "To identify the key information needed to solve the problem.",
        "inputs": [],
        "result": "p invests rs. 48,000\nq invests rs. 24,000\nTime period = 2 years\nFind the ratio of profit earned
      between p and q."
      },
      {
        "node": "Trace_causal_chain",
        "reason": "To understand the sequence of events and their impact on the profit calculation.",
        "inputs": [],
        "result": "1. Investment amounts: P invested Rs. 48,000 and Q invested Rs. 24,000.\n2. Time period: The business
      runs for 2 years.\n3. Profit division: Profit is proportional to investment since time is equal.\n4. Ratio
      calculation: 48,000 : 24,000 = 2 : 1.\n5. Conclusion: Profit is divided in the ratio 2 : 1."
      }
    ]
  },
  {
    "step": 1,
    "actions": [
      {
        "node": "List_Assumptions",
        "reason": "To state any underlying assumptions made while solving the problem.",
        "inputs": [
          {
            "source": "Extract_core_facts",
            "type": "result"
          }
        ],
        "result": "Assumptions:\n1. Profit is proportional to investment and time.\n2. Both p and q invest for the full
      2 years.\n3. No additional investments or withdrawals occur.\n4. Profit is distributed at the end of the
      period.\n5. The rate of return is the same for both partners."
      }
    ]
  },
  {
    "step": 2,
    "actions": [
      {
        "node": "Deconstruct_Problem",
        "reason": "To solve the problem in a structured manner.",
        "inputs": [
          {
            "source": "Extract_core_facts",
            "type": "result"
          },
          {
            "source": "List_Assumptions",
            "type": "result"
          }
        ],
        "result": "Step 1: Compare investments of p and q.\nStep 2: Use proportionality since time is equal.\nStep 3:
      Simplify the ratio.\nStep 4: Select the correct option."
      }
    ]
  },
  {
    "step": 3,
    "actions": [
      {
        "action": "final_answer",
        "inputs": [
          { "source": "Extract_core_facts", "type": "result" },
          { "source": "Trace_causal_chain", "type": "result" },
          { "source": "List_Assumptions", "type": "result" },
          { "source": "Deconstruct_Problem", "type": "result" }
        ],
        "result": {
          "final_synthesis": "c",
```

```
        "reason": "Since both partners invest for the same duration, the profit ratio equals the investment ratio
      48,000 : 24,000, which simplifies to 2 : 1."
        }
      }
    ]
  }
]
```


Input Query :

Ziklo causes Blaf, Blaf causes Trune, Trune causes Vork, and Vork causes Lumbo. Would Lumbo occur if not Ziklo instead of Ziklo?

Full Reasoning Trace for a Causal Task (CounterBench)


```
[
  {
    "step": 0,
    "actions": [{
      "node": "Extract_core_facts",
      "reason": "To identify the key relationships provided in the problem.",
      "inputs": [],
      "result": "Ziklo causes Blaf.\nBlaf causes Trune.\nTrune causes Vork.\nVork causes Lumbo."
    }]
  },
  {
    "step": 1,
    "actions": [{
      "node": "Trace_causal_chain",
      "reason": "To establish the order of events based on the given causal relationships.",
      "inputs": [{"source": "Extract_core_facts", "type": "result"}],
      "result": "1. Ziklo causes Blaf.\n2. Blaf causes Trune.\n3. Trune causes Vork.\n4. Vork causes Lumbo."
    }]
  },
  {
    "step": 2,
    "actions": [
      {
        "node": "Simulate_counterfactuals",
        "reason": "To determine if Lumbo would still occur without Ziklo.",
        "inputs": [{"source": "Trace_causal_chain", "type": "result"}],
        "result": "If not Ziklo, then Blaf would not occur. If Blaf would not occur, then Trune would not occur. If
      Trune would not occur, then Vork would not occur. If Vork would not occur, then Lumbo would not occur."
      },
      {
        "node": "List_assumptions",
        "reason": "To explicitly state any underlying assumptions made during the reasoning process.",
        "inputs": [{"source": "Trace_causal_chain", "type": "result"}],
        "result": "Assumptions:\n1. The causal chain is complete and unidirectional: Ziklo -> Blaf -> Trune -> Vork ->
      Lumbo is the only way for Lumbo to occur.\n2. The relationships are deterministic.\n3. 'Instead of Ziklo' means
      Ziklo does not occur, and nothing else changes.\n4. The system is closed: No external factors influence the chain
      of events."
      }
    ]
  },
  {
    "step": 3,
    "actions": [{
      "action": "final_synthesis",
      "inputs": [
        {"source": "Extract_core_facts", "type": "result"},
        {"source": "Trace_causal_chain", "type": "result"},
        {"source": "Simulate_counterfactuals", "type": "result"},
        {"source": "List_assumptions", "type": "result"}
      ],
      "result": "{ \"final_answer\": \"No\", \"reason\": \"If Ziklo does not occur, then Blaf, Trune, and Vork will not
      occur, and therefore Lumbo will not occur based on the given causal chain.\" }"
    }]
  }
]
```