Efficient Structured Test-Time Reasoning under Resource Constraints via Dynamic DAG Construction

Anonymous ACL submission

Abstract

Recent advances in language model reasoning have largely relied on frontier LLMs, whose high inference cost, API dependency, and latency limit practical deployment in resource-constrained settings. Although local and low-bit quantized models offer practical alternatives for single-GPU or on-device deployment, reliable reasoning on complex tasks remains challenging. Existing reasoning methods such as Chain-of-Thought (CoT), ReAct, and Tree of Thoughts (ToT) have been effective with large-scale LLMs, but their effectiveness does not necessarily transfer to resource-constrained local models. In this paper, we propose an efficient structured test-time reasoning framework via Dynamic DAG Construction. Inspired by cognitive insights, the framework modularizes human problem solving into reusable "Atomic Thinking" components and dynamically reassembles them into a task-specific directed acyclic graph (DAG), enabling dependency-aware reasoning while controlling unnecessary inference expansion. The framework also supports node-level tool integration, allowing external knowledge to be incorporated directly into the structured reasoning path. Evaluated on 20B–30B local models, our framework achieves performance competitive with GPT-4o across diverse reasoning benchmarks using an average of seven model calls per problem, while outperforming ToT in token efficiency and inference speed at comparable reasoning performance. The code is available at: an anonymous repository.

1 Introduction

Recent research has demonstrated human-level language understanding and reasoning capabilities through frontier-scale LLMs such as GPT-4 (OpenAI, 2023; Bubeck et al., 2023). However, deploying such models in real-world applications remains difficult due to immense computational requirements, high-cost GPU infrastructure, API dependency, and significant inference latency (Kaplan et al., 2020). These limitations have driven growing interest in resource-constrained deployment settings, including local-agent systems, single-GPU inference, on-device AI, and low-bit quantized models that can operate without distributed inference infrastructure (Xu et al., 2024; Zhao et al., 2026; Okoshi et al., 2026; Chen et al., 2025a).

However, maintaining reliable reasoning on complex tasks in such deployment environments remains challenging. Local or compressed language models provide practical alternatives to frontier-scale LLMs by reducing deployment cost and API dependency. Nevertheless, they often operate with reduced model capacity and constrained inference infrastructure, and in low-bit deployment regimes, reduced numerical precision can further lead to performance degradation and reduced reasoning reliability on complex tasks (Fu et al., 2025; Liang et al., 2026; Li et al., 2025). In particular, complex reasoning tasks require multi-step decomposition, consistent intermediate state tracking, and reliable aggregation of partial conclusions. Under these constraints, errors introduced in intermediate steps can easily propagate to the final answer (Snell et al., 2024; Wu et al., 2024).

Test-time scaling (TTS) provides a promising direction for improving reasoning reliability under practical deployment constraints by allocating additional computation during inference, rather than increasing model parameters or requiring further training (Snell et al., 2024; Wang et al., 2023; Brown et al., 2024; Zhang et al., 2025). Within this context, LLM research has explored a wide range of approaches for complex problem solving, spanning from prompting strategies to structured search frameworks. Chain-of-Thought (CoT) explicitly articulates intermediate reasoning steps in natural language to enable stepwise problem solving (Wei et al., 2022; Kojima et al., 2022). ReAct synergizes reasoning and acting to enhance the model's ability

to utilize external tools (Yao et al., 2023b). Tree of Thoughts (ToT) enables exploration over multiple reasoning paths by representing alternative hypotheses (Yao et al., 2023a; Long, 2023), and Graph of Thoughts (GoT) extends this capability by structuring the reasoning process as a graph, allowing for the aggregation and iterative refinement of intermediate thoughts (Besta et al., 2024).

Although these methods have improved reasoning performance in large-scale LLMs, their effectiveness does not necessarily transfer to resource-constrained deployment settings. CoT follows a single linear reasoning trajectory, making it vulnerable to error propagation (Wei et al., 2022). ReAct relies heavily on the model's planning reliability and may fall into inefficient action loops when action selection fails (Yao et al., 2023b). ToT improves reasoning through multi-path exploration and backtracking, but incurs substantial overhead in tokens, model calls, and execution time (Yao et al., 2023a; Long, 2023). Moreover, ToT remains limited in external tool integration due to its reliance on internal deliberation (Yao et al., 2023a). These limitations motivate a structured reasoning approach that improves reasoning reliability while explicitly controlling inference-time expansion.

To address these issues, we propose an efficient structured test-time reasoning framework via Dynamic DAG Construction. Instead of relying on sampling-based search as in ToT, our approach leverages cognitive prior knowledge to organize reusable Atomic Thinking units into a task-specific directed acyclic graph (DAG). This structured inference design guides the model toward a more efficient reasoning path that preserves logical consistency while minimizing unnecessary search costs. Furthermore, the node-level modular design enables direct tool integration and extensible reasoning across diverse tasks.

Our main contributions are as follows:

1. *Structured Test-Time Reasoning Framework*: We propose a dynamic DAG-based structured reasoning framework that improves reasoning reliability and efficiency in resource-constrained deployment settings.

2. *Competitive Benchmark Performance*: Using low-bit 20B–30B models, we achieve performance comparable to GPT-4o (OpenAI, 2024) across multiple reasoning benchmarks, particularly in mathematical and causal reasoning.

3. *Modular Tool Scalability*: A modular node-based design enables flexible integration of external tools, allowing external knowledge and task-specific operations to be incorporated directly into the structured reasoning process.

2 Method

We propose an efficient structured test-time reasoning framework that draws on cognitive insights into human problem-solving and constructs explicit reasoning traces in the form of a dynamic DAG.

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

To enable structured test-time reasoning in practical deployment settings, it is crucial to formalize core reasoning primitives that guide efficient inference while avoiding the inefficiencies of stochastic exploration. Drawing on 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 by 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 reasoning tasks or as a code module encompassing API calls and computational logic for external tool integration. These units serve as independent nodes in a DAG, enabling the model to operate within a structured reasoning space rather than searching an unstructured space. The library $\mathcal{U}$ is categorized into five functional groups based on the cognitive roles of its units, as shown in Table 1, and detailed prompt templates for the Atomic Thinking functions are provided in Appendix A.2.

2.2 Reasoning Framework via Dynamic DAG Construction

This section details our framework's mechanism for constructing and executing a task-specific DAG from the library $\mathcal{U}$, as shown 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 formulate a strategic plan for problem

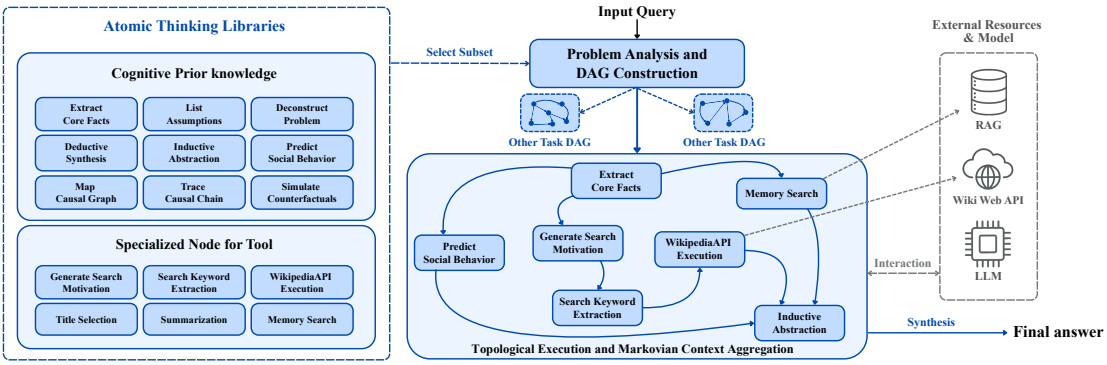

Figure 1: Our framework dynamically assembles a task-specific directed acyclic graph from the Atomic Thinking Library $\mathcal{U}$ and executes it through topological sorting. It employs a Markovian context aggregation mechanism to maintain logical consistency while ensuring token efficiency throughout 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 mitigate knowledge limitations through external information.

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

solving. At this stage, a subset $S \subseteq \mathcal{U}$ of functions required to solve the input X is selected from the Atomic Thinking Library $\mathcal{U} = \{f^{(1)}, \ldots, f^{(m)}\}$. 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. Since the same type of thinking may be used multiple times for different purposes, each node $v_i \in V$ is defined as a tuple consisting of a function and a 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 LLM-based summarization to merge 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:

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

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, avoids excessive context growth, and propagates condensed context through the DAG's explicit dependency structure. This allows the reasoning process to preserve logical consistency without unnecessary context expansion. Upon completion of all nodes, the framework synthesizes the accumulated reasoning steps to generate the final answer Y. The resulting reasoning trace $H = \{(v_t, o_t) \mid v_t \in T\}$ serves as an explainable record that demonstrates the logical basis for the conclusion.

2.3 Modular Design for Scalable Tool Integration

To address reasoning limitations under resource-constrained deployment settings, our framework integrates specialized tool nodes into the structured reasoning process. Specifically, tool-node configurations can be adapted to the requirements of each task, enabling scalable tool integration.

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 this trace rather than treated as an independent pipeline, with context shared across preceding and subsequent nodes. The details of these nodes are described in Appendix A.3.

2.3.2 Memory Retrieval Node for Long-Context Reasoning and Recall

For memory recall tasks that require retrieving specific historical information, the Atomic Thinking Library $\mathcal{U}$ is configured to include a Memory Retrieval 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 input context for generation, enabling the stored knowledge to be effectively utilized in the ongoing reasoning process.

3 Experiments

3.1 Experiment setup

Models. 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 models since they are publicly available local models known for reliable instruction following and stable structured-output generation. This low-bit 20B–30B setting is a practical testbed for resource-constrained deployment, where inference is expected to run on a single GPU without relying on external API services. The Mistral backbone is executed under a 4-bit quantization setting.

Benchmarks. To comprehensively evaluate reasoning capabilities, we use 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 on CounterBench (Chen et al., 2025b), which consists of 1,000 questions covering four types of counterfactual reasoning. For commonsense and open-domain reasoning, we use StrategyQA (Geva et al., 2021) (2,290 implicit multi-hop 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 whose QA subset (ID 1; 199 questions) requires integrating information across multiple turns and long-range dependencies.

Baselines. To validate our framework, we categorize the baselines into three groups. First, standard reasoning strategies include Zero-shot (question-only input) and Chain-of-Thought Prompting (Wei et al., 2022; Kojima et al., 2022). Second, we include advanced frameworks such as ReAct (Yao et al., 2023b) and Tree of Thoughts (Yao et al., 2023a). All hyperparameters for baseline methods

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 across benchmarks. Best scores are shown in **bold**, and 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).

follow validated and competitive settings from the existing literature. Finally, GPT-4o(Nov,24) (OpenAI, 2024) was included to assess the performance gap against frontier models. Detailed prompt settings 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

Table 2 reports the results of our framework against four major methodologies—Zero-shot, CoT Prompting, ToT, and ReAct—using the gemma-3-27b-it-qat-q4_0-gguf and Mistral-Small-24B-Instruct backbones.

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 achieves comparable performance on mathematical and causal benchmarks, while achieving an absolute improvement of over 10% on commonsense reasoning benchmarks. Unlike ToT, which is largely confined to the model'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 strong generalization not only in internal logical reasoning but also in tasks that require external or implicit world knowledge.

3.2.2 Comparison with a Large-Scale LLM

We further analyze whether the proposed framework can achieve competitive performance against a frontier model, as shown in Table 3.

In several domains, the framework brings 20B–30B backbones into a performance range that is competitive with GPT-4o, and in some cases exceeds it. On MATHQA, Mistral-Ours (0.8111) achieves more than a 2x improvement over GPT-4o (0.3853), and on the causal reasoning benchmark (CounterBench), it also surpasses GPT-4o (0.7520 vs. 0.6243). On StrategyQA, Gemma-Ours (+Web) (0.7576) attains performance comparable to GPT-4o (0.7581). This comparison demonstrates consistent performance gains over zero-shot and other baseline methods on the same backbone, while showing that the proposed framework can be competitive with large-scale LLMs in some domains.

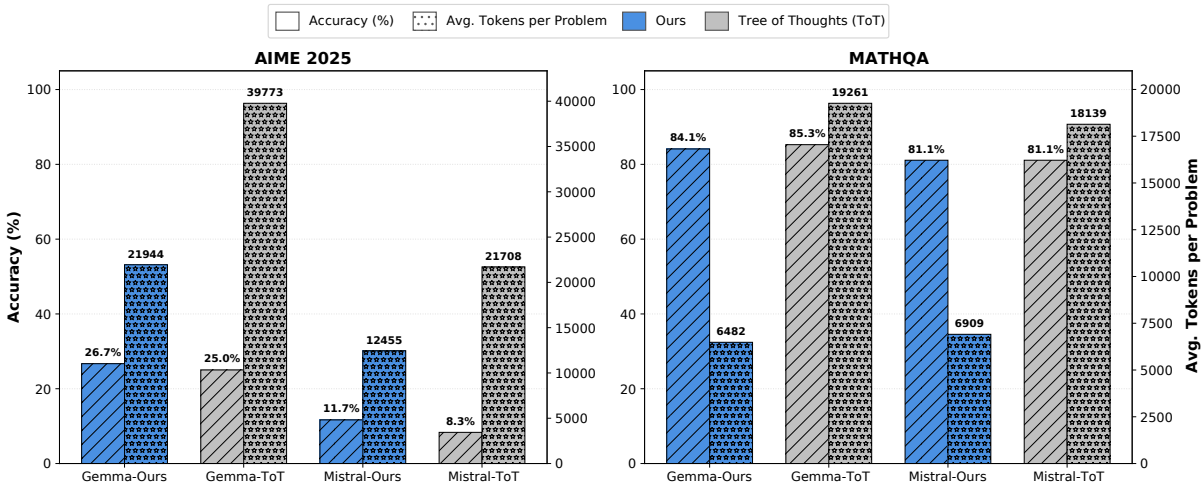

Figure 2: Comparison of accuracy and token efficiency on mathematical benchmarks (AIME 2025 and MATHQA). Results are shown for Gemma and Mistral backbones using our method (Ours) and Tree of Thoughts. The left y-axis reports accuracy, and the right y-axis reports the average number of tokens per problem.

3.2.3 Efficiency Analysis

We evaluate the efficiency of our framework with 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, differences 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 steps (Wei et al., 2022). Indeed, ToT has been reported to incur substantially higher computational costs on mathematical tasks (Yao et al., 2023a).

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

Execution time. At comparable accuracy levels (±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 takes an average of 74 seconds per problem, 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. Further efficiency details are provided in Appendix B.3.

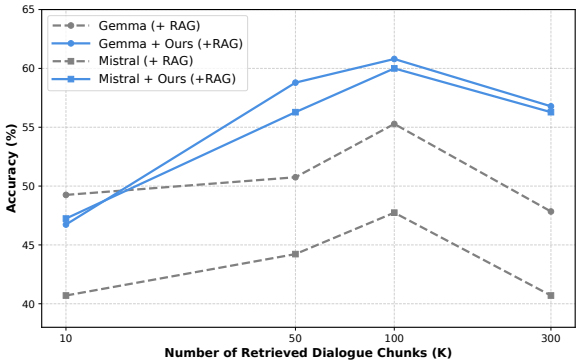

Figure 3: Scalability of memory retrieval under long-context conditions. Accuracy as a function of the number of retrieved dialogue chunks K.

3.2.4 Scalability Analysis

We evaluate scalability on the LoCoMo benchmark (Maharana et al., 2024). 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

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	**0.8411**	**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.

information. In contrast, models equipped with our framework consistently maintain higher accuracy 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. Further details are provided in Appendix B.4.

3.3 Ablation Study

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 accurate problem definition and early-stage plan formulation are key determinants of structured reasoning 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 structured reasoning process effectively complements the model's parametric knowledge when external information is required.

3.4 Interpretability

Figure 4 compares the summarized reasoning processes H generated by our framework against those produced by CoT and ToT on the same AIME 2025 problem. CoT (Wei et al., 2022) represents reasoning as a single linear text sequence, which often leads to error propagation, as 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 its downstream effects. This structured design allows the model's decision-making process to be explained in a human-interpretable manner. Further details are in Appendix C.1.

3.5 Additional Analysis

In this section, we analyze how different reasoning strategies behave in resource-constrained local inference settings. As shown in Table 2, CoT and ReAct show only limited improvements over Zero-shot prompting and underperform compared to ToT, suggesting that reasoning methods effective for large-scale LLMs do not always transfer reliably to local models under resource constraints. Specifically, in CoT prompting, minor logical inconsistencies introduced in intermediate reasoning steps can be amplified throughout the reasoning chain (Wei et al., 2022), leading to error propagation issues.

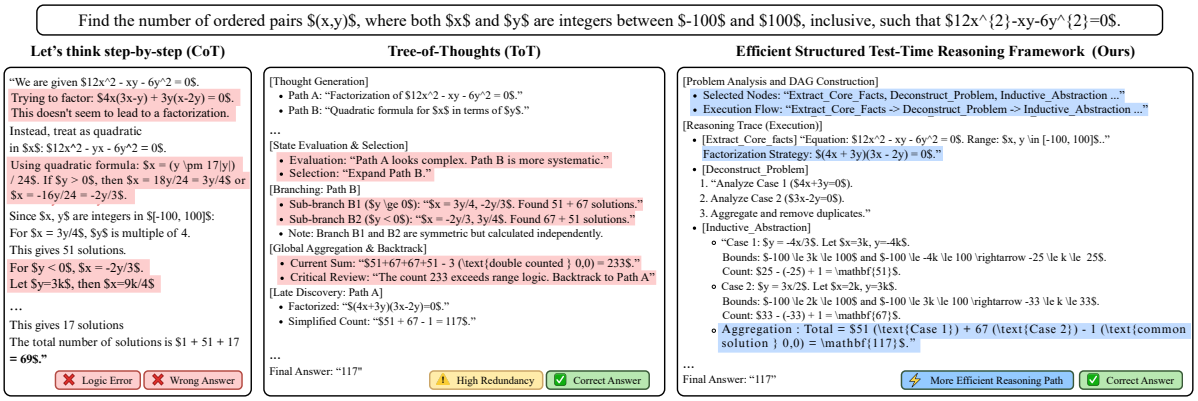

Figure 4: Comparison of summarized reasoning processes for the same AIME 2025 problem, highlighting structural differences between CoT, ToT, and our framework.

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 each problem into a tree, exploring multiple reasoning paths, and pruning unpromising branches (Yao et al., 2023a). However, this improvement comes with substantially higher inference-time overhead, as shown by its increased token usage, model calls, and execution time. These findings suggest that structured reasoning improves reliability when it explicitly controls unnecessary inference expansion in resource-constrained deployment settings.

4 Related Work

LLM Reasoning Strategies. A variety of strategies have been proposed to improve the reasoning performance of language models, including Chain-of-Thought (Wei et al., 2022; Kojima et al., 2022), Tree of Thoughts (Yao et al., 2023a; Long, 2023), and Graph of Thoughts (Besta et al., 2024). Recent DAG-based reasoning studies extend this line of work by using graph structures to organize intermediate thoughts, adjust reasoning trajectories, or improve exploration (Zhang et al., 2024a; Pandey et al., 2025; Zhang et al., 2024b; Dong et al., 2026). These approaches use search-based or graph-structured exploration to expand and refine candidate reasoning paths. In contrast, our framework uses a dynamic DAG as a dependency-aware execution structure for predefined Atomic Thinking units and tool-specific nodes, rather than a search space over candidate thoughts.

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 function-calling agents and tool-augmented language models for practical deployment (Erdogan et al., 2024; Chen et al., 2025a; Khiabani et al., 2025). However, these approaches primarily emphasize stable tool use and accurate function calling under deployment constraints, while offering limited support for high-level reasoning within a structured reasoning process. In contrast, our framework integrates tool-specific nodes directly into a dynamic DAG-based reasoning process, enabling tool use to be coordinated with structured reasoning.

5 Conclusion

In this work, we propose an Efficient Structured Test-Time Reasoning Framework via Dynamic DAG Construction, inspired by cognitive insights. Our approach achieves performance competitive with GPT-4o on mathematical and causal reasoning tasks while demonstrating higher efficiency than Tree of Thoughts. These results suggest that structured test-time reasoning can substantially improve reasoning reliability and efficiency in resource-constrained deployment settings, and may serve as a key foundation for future single-GPU deployable or on-device AI systems aiming to achieve high-level intelligence.

6 Limitations

The proposed Efficient Structured Test-Time Reasoning Framework improves reasoning reliability and efficiency in resource-constrained deployment settings. 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 faces limitations stemming from its plan-then-execute paradigm, which may restrict real-time adaptation or plan revision during the execution stage. This static nature motivates the incorporation of 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 test-time reasoning under practical resource constraints.

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.

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.

Jiangwen Dong, Zehui Lin, Wanyu Lin, and Mingjin Zhang. 2026. S-DAG: A subject-based directed acyclic graph for multi-agent heterogeneous reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

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.

Yao Fu, Xianxuan Long, Runchao Li, Haotian Yu, Mu Sheng, Xiaotian Han, Yin Yu, and Pan Li. 2025. Quantized but deceptive? a multi-dimensional truthfulness evaluation of quantized LLMs. *arXiv preprint*, arXiv:2508.19432.

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.

Zhen Li, Yupeng Su, Runming Yang, Congkai Xie, Zheng Wang, Zhongwei Xie, Ngai Wong, and Hongxia Yang. 2025. Quantization meets reasoning: Exploring LLM low-bit quantization degradation for mathematical reasoning. *arXiv preprint*, arXiv:2501.03035.

Yesheng Liang, Haisheng Chen, Zihan Zhang, Zhijian Liu, and Song Han. 2026. ParoQuant: Pairwise rotation quantization for efficient reasoning LLMs. In *Proceedings of the International Conference on Learning Representations*.

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.

Yasuyuki Okoshi, Hikari Otsuka, Daichi Fujiki, and Masato Motomura. 2026. Towards quantization-aware training for ultra-low-bit reasoning LLMs. In *Proceedings of the International Conference on Learning Representations*.

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.

Tushar Pandey, Ara Ghukasyan, Oktay Goktas, and Santosh Kumar Radha. 2025. Adaptive graph of thoughts: Test-time adaptive reasoning unifying chain, tree, and graph structures. *arXiv preprint*, arXiv:2502.05078.

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, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th 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.

Zhenyu Wu, Qingkai Zeng, Zhihan Zhang, Zhaoxuan Tan, Chao Shen, and Meng Jiang. 2024. Large language models can self-correct with key condition verification. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Jiajun Xu, Zhiyuan Li, Wei Chen, Qun Wang, Xin Gao, Qi Cai, and Ziyuan Ling. 2024. On-device language models: A comprehensive review. *arXiv preprint*, arXiv:2409.00088.

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.

Yifan Zhang, Yang Yuan, and Andrew Chi-Chih Yao. 2024a. On the diagram of thought. *arXiv preprint*, arXiv:2409.10038.

Zhenyu Zhang and 1 others. 2024b. DARG: Dynamic evaluation of large language models via adaptive reasoning graph evolvement. *arXiv preprint*, arXiv:2406.17271.

Changsheng Zhao, Ernie Chang, Zechun Liu, Chia-Jung Chang, Wei Wen, Chen Lai, Sheng Cao, Yuandong Tian, Raghuraman Krishnamoorthi, Yangyang Shi, and Vikas Chandra. 2026. MobileLLM-R1: Exploring the limits of sub-billion language model reasoners with open training recipes. In *Proceedings of the International Conference on Learning Representations*.

A Implementation Details

This appendix describes the implementation details of the proposed reasoning framework, including the system prompt for problem analysis and DAG construction, as well as 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 the logical dependencies among nodes. The Atomic Thinking Library $\mathcal{U}$ is predefined, but it can be flexibly extended by adding specialized nodes, such as web search 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 Atomic Thinking Function Prompt Templates

For reproducibility, we provide the fixed instruction templates used for each Atomic Thinking function in Table 5. Each function is implemented as an independent prompt-based module.

A.3 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, as shown in Table 6. 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. In 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.

B Experimental Details

B.1 Experimental Setup

To ensure a fair and reproducible comparison, all methods were evaluated under a unified inference setting with *max_new_tokens=3000* and *temperature=0.1* per call. The same prompt template for each Atomic Thinking function was used across all benchmarks, and the detailed templates are provided in Appendix A.2.

- **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., 2022) 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 is used to determine the final answer.

- **Tree of Thoughts.** For ToT, hyperparameters are set according to the standard ToT configuration used in prior work (Yao et al., 2023a). 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 the original ToT work and tailored to each benchmark.
  As shown in Table 2, this configuration yields substantial improvements over the backbone across multiple benchmarks, confirming that ToT serves as a strong structured reasoning baseline. Since ToT reaches a reasoning performance level comparable to our method in several tasks, we compare efficiency in this comparable-performance regime, focusing on model calls, token usage, and execution time.

- **Frontier Models.** For comparison with large-scale frontier models, GPT-4o (2024-11-20) (OpenAI, 2024) and GPT-5 (Medium) (OpenAI, 2025) are evaluated under a zero-shot setting, as reported in Table 7.

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

As shown in Table 7, GPT-5 (Medium) achieves the highest overall performance across most benchmarks, reflecting the strong reasoning capabilities of frontier-scale LLMs. Nevertheless, when the proposed framework is applied to 20B–30B local language models, the resulting system attains performance comparable to GPT-5 (Medium) on multiple tasks and notably surpasses it on the causal

Atomic Thinking Function	Prompt Template
`Extract_core_facts`	Extract and summarize only the core facts and essential information needed to solve this problem. Eliminate irrelevant details. Focus on conditions, constraints, key variables, and goals.
`List_assumptions`	List all explicit and implicit assumptions you're making to approach this problem. Include hidden premises, logical defaults, and anything taken for granted.
`Deconstruct_problem`	Break the problem into a step-by-step sequence of actions or sub-problems. Clearly explain what needs to be done at each step to move toward the solution.
`Trace_causal_chain`	Organize the events, actions, or changes in chronological order. For each step, explain the cause-and-effect relationship: what caused it and what resulted.
`Predict_social_behavior`	Based on common human behavior or general patterns, infer what a typical person or agent would do in this scenario. Note how the expected behavior matches or conflicts with what is observed.
`Map_causal_graph`	Identify the key elements (nodes) and describe a causal graph showing how they influence each other. Use directed edges to indicate cause-and-effect relationships.
`Simulate_counterfactuals`	Imagine one key element was different. What would have happened if that element had been changed, removed, or reversed? Provide one or more counterfactual scenarios and reason through their consequences.
`Deductive_synthesis`	Start by identifying the big-picture structure or overall pattern. Then analyze the details within that framework using a top-down reasoning approach.
`Inductive_abstraction`	Begin from specific details or clues, and infer the larger structure, pattern, or conclusion that they support. Use a bottom-up reasoning strategy.

Table 5: Atomic Thinking function prompt templates.

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

reasoning benchmark, CounterBench.

These results suggest that structured reasoning execution can improve reasoning reliability in resource-constrained local inference settings and partially narrow the gap with frontier LLMs without relying on model scaling or external API-based inference.

B.3 Detailed Efficiency Analysis

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

This analysis focuses on mathematical reasoning tasks, where differences in computational cost across reasoning frameworks tend to be most pronounced. We therefore evaluate average execution time on the AIME and MATHQA benchmarks, with the results shown in Table 8. 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 is likely because, under the Mistral-AIME setting, both methods achieve low accuracy and often fail to complete successful reasoning, leading to shorter execution times. However, this behavior is not observed consistently across other benchmarks.

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,

	MATHQA	AIME2025	CounterBench	StrategyQA	CoLoTA
GPT-5 (Medium)	**0.8885**	**0.3670**	0.6860	**0.7721**	**0.6768**
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 7: Performance comparison with GPT-5. Despite the difference in model size, our method surpasses GPT-5 on the CounterBench task. Best scores are shown in **bold**, and second-best scores are underlined.

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

Table 8: Average execution time (seconds per problem) comparison between our framework and ToT on mathematical reasoning benchmarks. Lower is better; **bold** indicates the fastest method.

whereas ToT typically requires 25–26 calls.

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.

B.4 Scalability Analysis Details

- **Preprocessing.** For the memory retrieval task, we use the dialogue history associated with ID 1 in 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 sequence of text, 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 in-

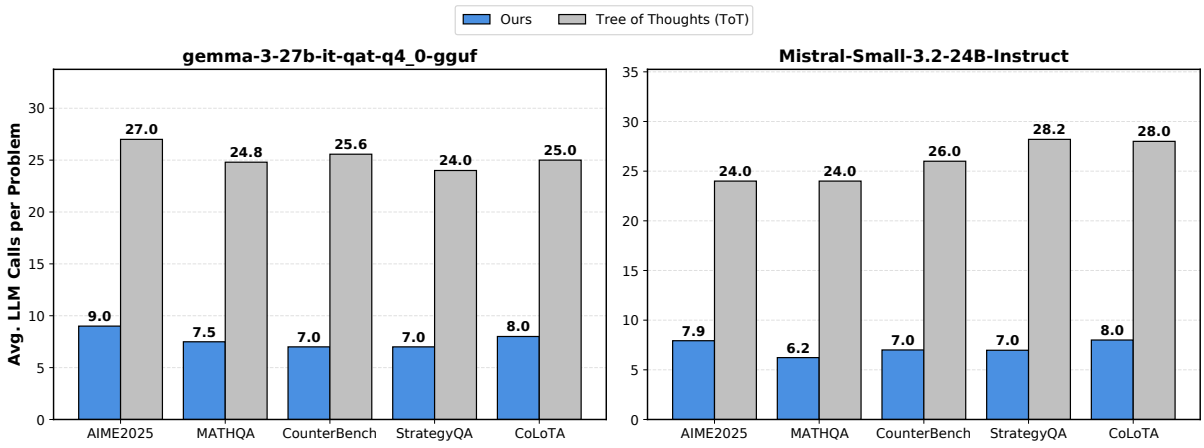

Figure 5: Average number of LLM calls per problem comparison between our framework and ToT across all benchmarks.

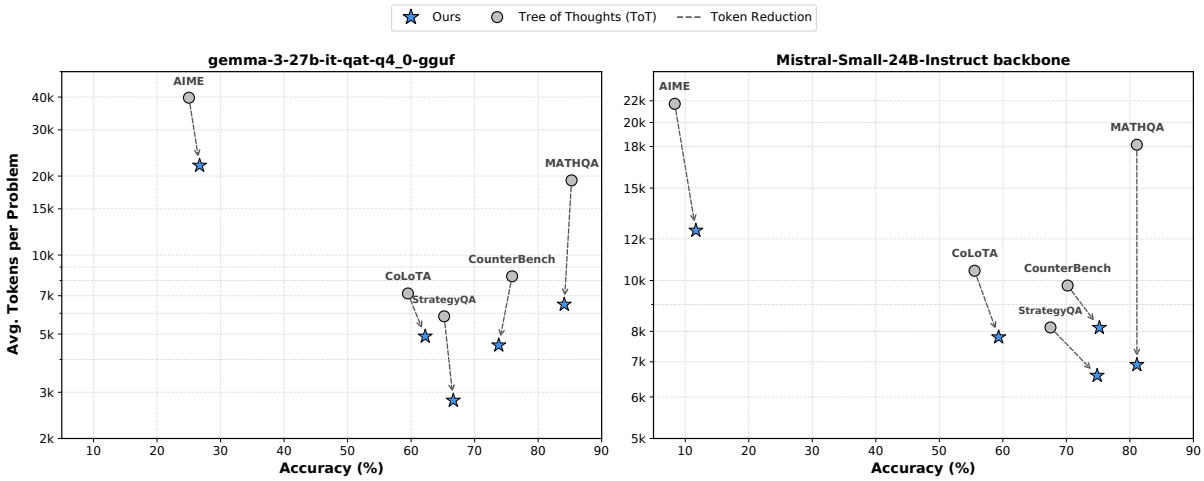

Figure 6: Comparison of average token usage per problem between our framework and ToT across all benchmarks.

correct 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.\" }"
    }]
  }
]
```