Reasoning Architectures in Large Language Models: Comparing Chain-of-Thought, Tree-of-Thought, and Graph-of-Thought Techniques

Anusha Mahesh
Department of Computer Science
CSCI 630 Fall 2025
Rochester, NY
am4556@rit.edu

Pravallika Nakarikanti
Department of Computer Science
CSCI 630 Fall 2025
Rochester, NY
pn4154@rit.edu

***Abstract*—Large language models (LLMs) have achieved remarkable performance across diverse natural language and reasoning tasks. However, their reasoning process often remains implicit, unstructured, and difficult to control. To address these limitations, researchers have developed structured prompting and reasoning frameworks such as Chain-of-Thought (CoT), Tree-of-Thought (ToT), and Graph-of-Thought (GoT). This survey examines these three techniques by focusing on how they differ in reasoning structure, search strategy, and interpretability. By comparing [1]'s Chain-of-Thought prompting (NeurIPS 2022), [2]'s Tree-of-Thought framework (NeurIPS 2023), and [3]'s Graph-of-Thought model (AAAI 2024), we analyze how reasoning evolved from linear prompting to structured search and finally to networked thought representations. We also discuss tradeoffs in accuracy, computational cost, and scalability, providing a comprehensive overview current LLM reasoning architectures and their implications for the next-generation LLM reasoning systems.**

I. Introduction

Over the last few years, large language models (LLMs) such as GPT-4, PaLM, and Claude have reshaped how we build intelligent systems. Despite their impressive capabilities, a key challenge remains: how to make LLMs reason in a way that is both reliable and transparent. Early models often "hallucinate" or make unwarranted logical leaps because their reasoning is buried within billions of parameters rather than expressed in interpretable steps.

To address this limitation, researchers have explored methods that externalize reasoning by prompting models to generate intermediate thoughts. A key insight came from [4], who showed that LLMs can reason even in zero-shot settings by simply instructing a model to "think step by step", unlocking latent reasoning abilities without task-specific training.

Following this, [1] introduced *Chain-of-Thought (CoT) prompting*, which improved performance across arithmetic, symbolic, and commonsense reasoning tasks. CoT reframes the LLM from an end-to-end predictor into a transparent, stepwise problem solver. CoT's linear process was generalized into a structured search over multiple reasoning branches. Each "thought" becomes a node in a tree, allowing the model to explore different reasoning directions and evaluate partial solutions before producing a final answer.

Building on this, [2] introduced the Tree-of-Thought (ToT) framework, which extends CoT by enabling structured exploration of multiple reasoning paths. More recently, [3] proposed *Graph-of-Thought (GoT)*, which generalizes reasoning beyond a tree to a dynamic graph. It now becomes a networked process, enabling interactions between partial thoughts. In GoT, thoughts can merge, diverge, and interact, enabling more flexible reasoning paths suitable for complex, interconnected problems like planning and scientific discovery.

Together, these techniques show an evolution from linear to branching to networked reasoning. This survey places these techniques within the broader LLM reasoning literature (*e.g,* [5]), comparing how they address interpretability, scalability, and performance.

II. Background and Motivation

Reasoning is at the heart of artificial intelligence. Traditional AI systems relied on explicit symbolic reasoning [6], where both logic and rules were hand-coded. With the rise of deep learning, the field shifted toward statistical pattern recognition, enabling systems to learn directly from data. LLMs now generate coherent and contextually appropriate text, but their reasoning remains largely implicit, with limited interpretability or guarantees of correctness.

Recent efforts toward structured reasoning techniques stems from the desire to blend the power of neural language models with the transparency and control of symbolic reasoning. Chain-of-Thought (CoT), Tree-of-Thought (ToT), and Graph-of-Thought (GoT) all aim to make intermediate reasoning steps explicit, allowing both humans and other systems to interpret or guide the model's reasoning process. This work helps bridge the gap between "black-box" neural inference and "glass-box" decision-making.

These techniques act as diagnostic layers (*aka* debugging) in the model's reasoning process: by exposing intermediate states, they make it easier to trace, evaluate, and optimize logical flow.

III. Chain-of-Thought Prompting

A. Concept and Mechanism

Chain-of-Thought (CoT) prompting, proposed by [1] in 2022, improves reasoning in LLMs by explicitly including step-by-step rationales into prompts. Instead of just providing a question, the prompt includes reasoning steps that guide the reasoning process toward a final answer.

For example:

> **Q:** Roger has 5 tennis balls. He buys 2 cans of 3 balls each. How many does he have?
> **A:** Roger started with 5. Each can has 3, so 2 cans have 6. 5 + 6 = 11. The answer is 11.

Such structured prompts help the model decompose problems into intermediate steps, more closely mirroring human reasoning. [1] demonstrated that this technique significantly improves accuracy on tasks like arithmetic reasoning (GSM8K) [7], symbolic manipulation, and commonsense inference.

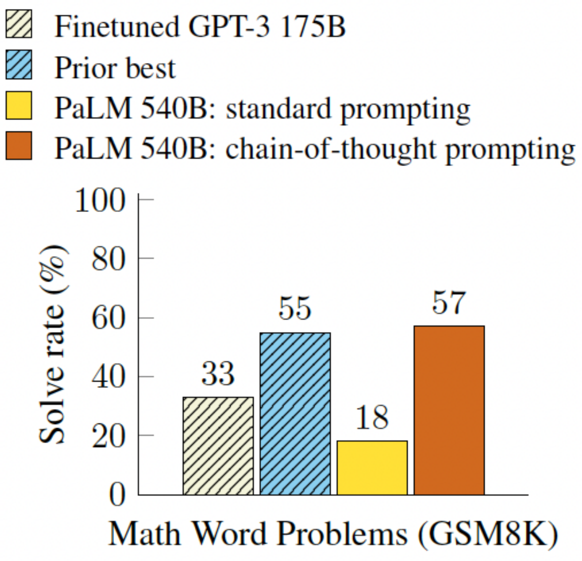


Figure 1: Comparison between standard prompting and Chain-of-Thought prompting (adapted from [1], NeurIPS 2022).
Shows how adding step-by-step reasoning enables large models like PaLM 540B to reach new state-of-the-art performance on GSM8K.

B. Implementation and Results

CoT is typically implemented through few-shot prompting, where a few examples (⟨input, reasoning, output⟩ triplets) demonstrating the reasoning process are provided. Larger models (*e.g.*, with billions of parameters) exhibit strong reasoning gains under CoT prompting, while smaller models tend to generate inconsistent or incoherent reasoning chains, showing an emergent property: reasoning ability scales with model size.

Empirical results from the paper show:

- PaLM 540B [8] using CoT prompting achieved state-of-the-art accuracy on GSM8K.
- Accuracy improved from approximately 18% (standard prompting) to over 55% with CoT.
- CoT also improved performance on commonsense reasoning (StrategyQA) and symbolic tasks.

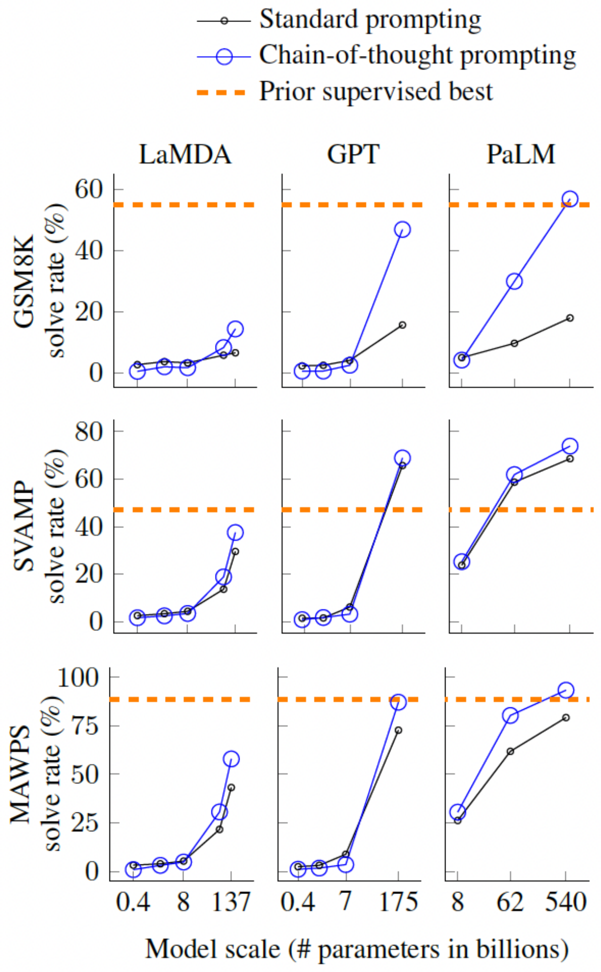


Figure 2: Performance comparison of standard prompting vs. Chain-of-Thought on multiple benchmarks (adapted from [1]).
CoT demonstrates scaling benefits with larger models, showing reasoning as an emergent capability.

Variants such as “Zero-Shot-CoT” further simplified the process by appending the phrase “Let’s think step by step.” to the prompt. This small change of eliminating handcrafted examples often unlocks reasoning behavior even without supervised examples.

C. Strengths and Limitations

Strengths:

- Simple to implement: no model retraining required.
- Improves interpretability by exposing reasoning traces.
- Works across domains (math, logic, commonsense).

These strengths are consistent with findings from [1], which demonstrated that explicit step-by-step rationales substantially improve LLM performance on arithmetic, commonsense, and symbolic reasoning tasks without having to training them.

Limitations:

However, as noted in [1] CoT remains constrained by:

- Linear reasoning structure: cannot explore multiple alternative paths.
- Once the model commits to one reasoning chain, it cannot backtrack.
- Increased output length (hence slightly higher inference cost).

In short, CoT transformed LLMs from black-box predictors into visible, stepwise solvers. However, its linear nature limits the ability to explore or improve alternative reasoning paths.

IV. Tree-of-Thought Framework

A. Concept and Mechanism

The Tree-of-Thought (ToT) framework, introduced by [2]. This work, extends the CoT idea into a structured search problem. Instead of generating a single reasoning chain, the ToT model produces multiple partial reasoning paths ("thoughts") which branch out like a tree from a single prompt, forming a search tree. Each node in the tree represents a partial reasoning step; branches represent alternative approaches.

At each stage, the model:

- Generates candidate "thoughts".
- Evaluates their promise (using heuristics or self-evaluation).
- Expands the most promising ones while pruning weaker paths, similar to beam search or Monte Carlo Tree Search.

This iterative process lets LLMs simulate human-like brainstorming, trying different reasoning paths and discarding weaker ones before settling on a final answer.

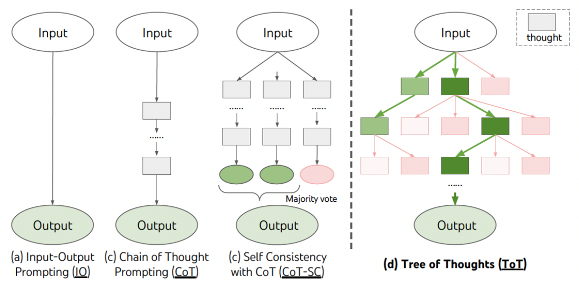

Input
Input
Input
Input
thought
Majority vote
Output
Output
Output
Output
(a) Input-Output Prompting (IO)
(c) Chain of Thought Prompting (CoT)
(c) Self Consistency with CoT (CoT-SC)
(d) Tree of Thoughts (ToT)

Figure 3: Tree-of-Thought schematic (adapted from [2], NeurIPS 2023).
Shows how ToT explores multiple branches, evaluates partial reasoning paths, and prunes or backtracks for improved decision-making.

B. Implementation and Results

The authors evaluated ToT on three challenging tasks:

- Game of 24 (arithmetic planning)
- Creative writing generation
- Mini crosswords (lexical reasoning)

Results:

- GPT-4 solved only 4% of Game-of-24 tasks using CoT, but 74% with ToT.
- ToT's deliberate search and evaluation significantly outperformed both standard and CoT prompting.
- ToT generalized well across reasoning, planning, and creative domains.

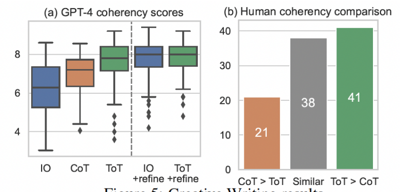
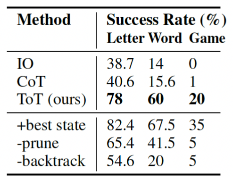


Figure 5: Creative Writing results.



Method	Success Rate (%)		
	Letter	Word	Game
IO	38.7	14	0
CoT	40.6	15.6	1
ToT (ours)	**78**	**60**	**20**
+best state	82.4	67.5	35
-prune	65.4	41.5	5
-backtrack	54.6	20	5

Table 3: Mini Crosswords results.

Figure 4: Comparative success rates of CoT vs. ToT performance across three tasks. (adapted from [2])
ToT dramatically increases performance through multi-path reasoning and backtracking.

C. Strengths and Limitations:

Strengths:

- Enables backtracking and multiple reasoning hypotheses.
- Offers interpretable reasoning trees that can be visualized and audited.
- Significantly boosts problem-solving accuracy in complex domains.

These advantages align with [2], which showed that multi-path exploration and pruning dramatically improve problem-solving accuracy, enabling LLMs to outperform CoT on tasks requiring deep search and strategic reasoning.

Limitations: The authors note that ToT's improved accuracy comes at the cost of substantial computational overhead, since it requires generating, scoring, and expanding multiple reasoning branches. Its key limitations include:

- High inference cost: each branch expansion triggers additional model calls.
- Strong dependence on heuristic quality for scoring and pruning thoughts.
- Poor scalability on tasks with deep or wide search spaces, where branch growth becomes exponential.

ToT is a shift from static reasoning to a more dynamic exploratory nature, balancing both breadth and depth in a manner much more like human problem-solving.

V. Graph-of-Thought Framework

A. Concept and Mechanism

While ToT introduces structured branching and backtracking, its tree structure still enforces a hierarchy. However, real-world reasoning often involves merging ideas, reusing partial solutions, or connecting insights across contexts. The *Graph-of-Thought (GoT)*, proposed by [3], addresses this by representing reasoning as a dynamic graph.

In GoT, nodes are "thoughts" (textual or symbolic statements) and edges represent relationships such as causality, dependencies, or analogies. This design supports bidirectional reasoning: one thought can influence multiple others, and feedback loops can emerge naturally.

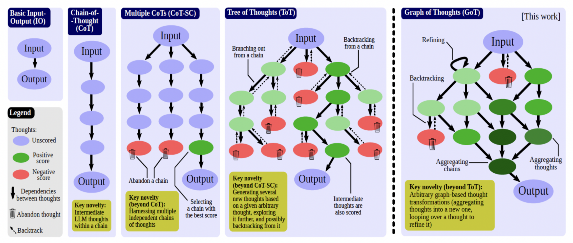
B. Implementation and Results

GoT models reasoning as a graph G = (V, E) where:

- Each vertex (v) is a thought (a piece of reasoning text or symbolic state).
- Edges (E) represent dependencies – how one thought leads to or modifies another.

In GoT, the model interacts with an evolving "thought graph" through these transformations:

- **Aggregation:** merging multiple thoughts into a new one, linking thoughts that share semantic or causal relationships.
- **Refinement:** iteratively improving or correcting a thought, pruning redundant or low-value thoughts.
- **Expansion:** generating new thoughts from existing ones.



Basic Input-Output (IO)
Chain-of-Thought (CoT)
Multiple CoTs (CoT-SC)
Tree of Thoughts (ToT)
Graph of Thoughts (GoT)
[This work]
Input
Output
Legend
Thoughts:
Unscored
Positive score
Negative score
Dependencies between thoughts
Abandon thought
Backtrack
Key novelty: Intermediate LLM thoughts within a chain
Key novelty (beyond CoT): Harnessing multiple independent chains of thoughts
Abandon a chain
Selecting a chain with the best score
Branching out from a chain
Backtracking from a chain
Key novelty (beyond CoT-SC): Generating several new thoughts based on a given arbitrary thought, exploring it further, and possibly backtracking from it
Intermediate thoughts are also scored
Refining
Backtracking
Aggregating chains
Aggregating thoughts
Key novelty (beyond ToT): Arbitrary graph-based thought transformations (aggregating thoughts into a new one, looping over a thought to refine it)

Figure 5: Comparison of CoT, ToT, and GoT (adapted from [3]).
GoT aggregates, refines, and revisits thoughts through a fully connected reasoning network.

These graph transformations support flexible and reusable reasoning for tasks like code synthesis, document merging, or scientific hypothesis generation. Each thought can be processed by a separate LLM instance, allowing distributed reasoning over the graph.

1) Results: The authors reported that GoT outperformed ToT across several tasks on sorting, keyword extraction, and document operations. According to this work:

- GoT improved sorting accuracy by 62% over ToT.
- Reduced computational cost by 31% through reusing intermediate reasoning steps.
- Provided a modular and extensible framework supporting multiple LLMs (GPT-3.5, GPT-4, LLaMA 2).

C. Strengths and Limitations

Strengths:

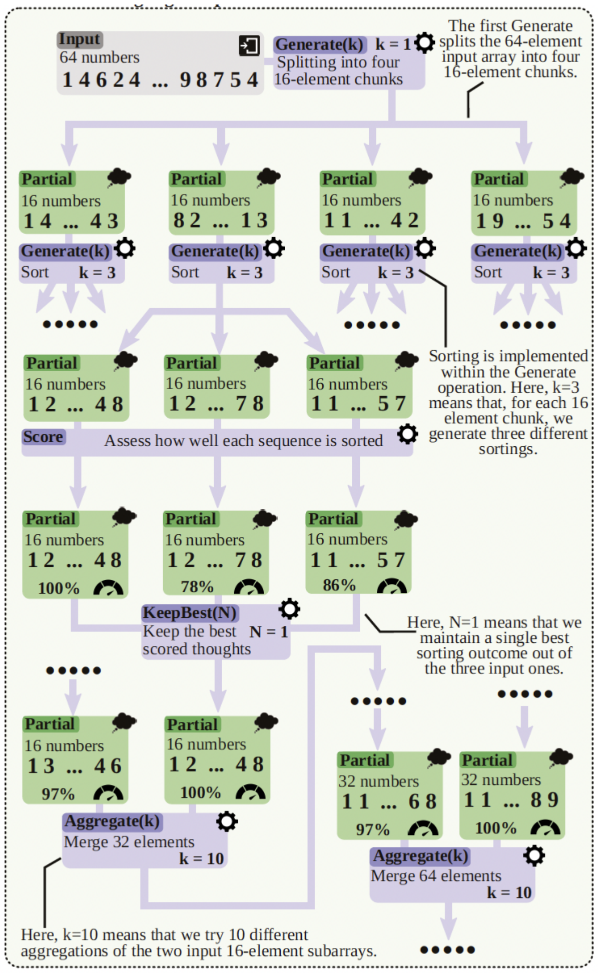
- Most flexible reasoning structure: supports merging, backtracking, and iterative refinement.
- Better cost-to-performance ratio.
- Scales naturally to collaborative or distributed reasoning.

These strengths are consistent with [3], which showed that GoT's graph-based reasoning allows models to reuse partial results, compose thoughts more effectively, and achieve higher accuracy across tasks such as sorting, document transformation, and keyword extraction—all while reducing redundant computation compared to ToT.

Limitations: As highlighted by [3], GoT's flexibility comes with notable overhead. The system must actively manage an evolving reasoning graph, prevent redundant cycles, and synchronize multiple LLM modules, all of which add engineering and computational complexity –

- Large, continuously growing thought graphs can become difficult to track and optimize.
- Requires a controller to monitor graph structure, enforce coherence, and eliminate redundant or looping thoughts.
- Coordination across multiple LLM nodes introduces communication overhead and slows down inference.

In essence, GoT represents a transformation of reasoning from a "path" or "tree" into a living network of interconnected ideas.



Input
64 numbers
1 4 6 2 4 ... 9 8 7 5 4
Generate(k) k = 1
Splitting into four 16-element chunks
The first Generate splits the 64-element input array into four 16-element chunks.
Partial
16 numbers
1 4 ... 4 3
8 2 ... 1 3
1 1 ... 4 2
1 9 ... 5 4
Generate(k)
Sort k = 3
Sorting is implemented within the Generate operation. Here, k=3 means that, for each 16 element chunk, we generate three different sortings.
1 2 ... 4 8
1 2 ... 7 8
1 1 ... 5 7
Score
Assess how well each sequence is sorted
100%
78%
86%
KeepBest(N)
Keep the best N = 1 scored thoughts
Here, N=1 means that we maintain a single best sorting outcome out of the three input ones.
1 3 ... 4 6
97%
1 2 ... 4 8
100%
Aggregate(k)
Merge 32 elements k = 10
32 numbers
1 1 ... 6 8
97%
1 1 ... 8 9
100%
Merge 64 elements k = 10
Here, k=10 means that we try 10 different aggregations of the two input 16-element subarrays.

Figure 6: The GoT reasoning process. (adapted from [3]) Shows how GoT's directed graph allows aggregation, refinement, and feedback loops—capturing complex reasoning patterns beyond linear or tree-based models.

VI. Discussion and Comparison

The progression from CoT to ToT to GoT architectures represents a shift from reactive language modeling toward deliberate cognitive modeling. As models evolve, reasoning architectures like GoT point to the next generation of reasoning systems that operate as a collaborative network of thoughts, reflecting aspects of human-like cognition. As implemented in ([9]), combining the power of these reasoning strategies

TABLE I
COMPARATIVE EVALUATION OF CoT, ToT, AND GoT

Aspect	**Chain-of-Thought (CoT)**	**Tree-of-Thought (ToT)**	**Graph-of-Thought (GoT)**
Accuracy & Reliability	Moderate; effective on arithmetic and symbolic reasoning but can drift logically.	High; explores multiple branches, improving success rates on complex multi-step tasks.	Very high; integrates and reuses thoughts, yielding the most robust reasoning.
Computational Cost	Low; single-pass inference with slightly longer outputs.	High; requires multiple model calls per branch expansion.	Moderate; reuses intermediate reasoning to reduce redundant computation.
Performance & Practicality	Best for latency-sensitive or production settings due to low overhead.	Appropriate for research or high-stakes reasoning (e.g., theorem proving, knowledge-base tasks).	Excels in exploratory or autonomous planning tasks; balances quality with efficiency.
Interpretability & Scalability	Produces readable linear chains that are easy to trace but limited in scope.	Provides tree visualizations of alternative paths, aiding debugging and analysis.	Offers semantic graphs connecting related ideas, ideal for deep audits, safety, and scalable reasoning.

with the factual, structured nature of Knowledge Graphs (KGs) helps large language models "think with evidence" rather than just "think out loud." This technique makes models smarter and more dependable by reducing hallucinations and offering organizations greater reliability and control. It also provides a practical way to use knowledge graphs, which are often difficult to use directly for open-ended natural language questions. While each architecture does have distinct advantages, their trade-offs become clearer when compared across three areas: accuracy and reliability, computational cost, and interpretability and scalability (see Table 1).

A. Analysis

- CoT laid the foundation for interpretable reasoning by revealing intermediate steps. However, its linear structure restricts exploration.
- ToT introduced structured search and backtracking, substantially improving reasoning quality but at a significant computational expense.
- GoT balanced both worlds: retained interpretability while enhancing efficiency and adaptability through graph-based connections.

VII. IMPLEMENTATION CONSIDERATIONS

The integration of these frameworks requires careful system design. [1] implement CoT via prompt templates with few-shot examples, requiring no additional infrastructure and offering minimal overhead with linear reasoning. [2] implement ToT through search orchestration that manages branch generation, evaluation, and pruning across multiple LLM calls, providing structured exploration at higher computational cost. [3] implement GoT using graph structures where thoughts are vertices and edges represent dependencies, requiring coordination to prevent cycles and manage transformations while balancing flexibility with graph management complexity.

Each approach represents a trade-off between implementation simplicity and reasoning control. The choice reflects priorities around accuracy, latency, and system maintainability: CoT suits latency-sensitive applications requiring transparency, ToT fits problems where reasoning quality justifies computational expense, and GoT addresses tasks where thought reusability and iterative refinement outweigh graph coordination overhead.

VIII. CONCLUSION

Chain-of-Thought, Tree-of-Thought, and Graph-of-Thought each represent progressive steps toward more transparent and effective reasoning in large language models. CoT makes reasoning visible; ToT makes it strategic; and GoT makes it interconnected.

The gradual move from linear to graph-based reasoning reveals a growing emphasis on deliberation, feedback, and compositionality in AI systems. Understanding these techniques not only provide insights into current LLM behavior but also serves as a roadmap for designing future reasoning architectures that are accurate, interpretable, and efficient.

REFERENCES

[1] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[2] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," *Advances in neural information processing systems*, vol. 36, pp. 11 809–11 822, 2023.

[3] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawski, L. Gianinazzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczyk *et al.*, "Graph of thoughts: Solving elaborate problems with large language models," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 38, no. 16, 2024, pp. 17 682–17 690.

[4] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *Advances in neural information processing systems*, vol. 35, pp. 22 199–22 213, 2022.

[5] B. Wang, S. Min, X. Deng, J. Shen, Y. Wu, L. Zettlemoyer, and H. Sun, "Towards understanding chain-of-thought prompting: An empirical study of what matters," *arXiv preprint arXiv:2212.10001*, 2022.

[6] M. Fang, S. Deng, Y. Zhang, Z. Shi, L. Chen, M. Pechenizkiy, and J. Wang, "Large language models are neurosymbolic reasoners," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 38, no. 16, 2024, pp. 17 985–17 993.

[7] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano *et al.*, "Training verifiers to solve math word problems," *arXiv preprint arXiv:2110.14168*, 2021.

[8] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.

[9] A. Amayuelas, J. Sain, S. Kaur, and C. Smiley, "Grounding llm reasoning with knowledge graphs," *arXiv preprint arXiv:2502.13247*, 2025.