Forest-of-Thought: Scaling Test-Time Compute for Enhancing LLM Reasoning

Zhenni Bi † **Kai Han** † **Chuanjian Liu** **Yehui Tang** * **Yunhe Wang** *
Huawei Noah's Ark Lab
{bizhenni, kai.han, liuchuanjian, yehui.tang, yunhe.wang}@huawei.com

Abstract

Large Language Models (LLMs) have shown remarkable abilities across various language tasks, but solving complex reasoning problems remains a challenge. While existing methods like Chain-of-Thought (CoT) and Tree-of-Thought (ToT) enhance reasoning by decomposing problems or structuring prompts, they typically perform a single pass of reasoning and may fail to revisit flawed paths, compromising accuracy. To address this, we propose a novel reasoning framework called Forest-of-Thought (FoT), which integrates multiple reasoning trees to leverage collective decision-making for solving complex logical problems. FoT utilizes sparse activation strategies to select the most relevant reasoning paths, improving both efficiency and accuracy. Additionally, we introduce a dynamic self-correction strategy that enables real-time error correction and learning from past mistakes, as well as consensus-guided decision making strategies to optimize correctness and computational resources. Experimental results demonstrate that the FoT framework, combined with these strategies, significantly enhances the reasoning capabilities of LLMs, enabling them to solve complex tasks with greater precision and efficiency.

1. Introduction

Large Language Models (LLMs) have revolutionized natural language processing by demonstrating remarkable abilities across a wide range of language tasks. Leveraging vast datasets and complex architectures, LLMs such as ChatGPT (Kojima et al., 2022; Achiam et al., 2023) and LLaMA (Touvron et al., 2023) can generate coherent essays, answer complex questions, and even engage in multi-turn dialogues with human-like fluency. These models excel at tasks requiring not only linguistic understanding but also basic reasoning, such as translating text between languages, summarizing lengthy documents, and creating code based on plain language instructions. The versatility and adaptability of LLMs have made them invaluable tools in both industry and research, simultaneously providing new avenues for addressing general-purpose problems.

How to enable LLM to successfully solve hard reasoning problems is still challengable. A seires of works have been proposed to introduce more inference at test time based on a well-trained LLM (Wei et al., 2022; Yao et al., 2024; Snell et al., 2024; OpenAI, 2024). Chain-of-Thought (CoT) (Wei et al., 2022) provides a few chain of thought demonstrations in prompting as exemplars to emerge reasoning abilities of LLMs. Tree-of-Thought (ToT) (Yao et al., 2024) allowing language models to explore multiple reasoning paths and self-evaluate to make more globally informed decisions. Graph-of-Thought (GoT) (Besta et al., 2024) advances LLM prompting by structuring information as a graph of interconnected "thoughts", enabling synergistic reasoning and feedback loops.

These method perform reasoning by richer prompt or decomposing complex problem into several easier subproblems. They only perform a single complete reasoning pass on the problem, which cannot ensure that the problem is solved or guarantee correctness. For example, in a complex mathematical word problem, a Tree-of-Thought approach might decompose the problem into smaller steps, such as isolating terms or simplifying expressions. However, while breaking down the question, it may still overlook critical details or make errors in intermediate steps, leading to an incorrect final answer. Once it completes a single reasoning path, it typically does not revisit other possible approaches if the initial path is flawed. This lack of re-evaluation can result in a solution that fails to address the full complexity of the problem, as alternative paths are often prematurely abandoned and left unexplored, thereby compromising accuracy. Instead, humans tend to repeatedly think and verify from different perspectives when dealing with complex problems to truly solve the problem and provide answers with higher accuracy.

†Equal contribution. *Corresponding author.

Preprint.

In this paper, we propose a new reasoning framework named Forest-of-Thought (FoT) to scale up test-time compute for enhancing the reasoning abilities of LLMs, as shown in Figure 1. FoT integrates multiple reasoning trees to leverage the advantages of collective decision-making in handling complex logical reasoning tasks. By utilizing sparse activation strategies, we select the most relevant reasoning paths for each tree, improving both the efficiency and accuracy of the model. To further enhance the reasoning process, we propose a dynamic self-correction strategy, which allows the model to automatically identify and correct errors during the reasoning process, leveraging both real-time correction and historical learning. Additionally, we incorporate consensus-guided decision making strategies to optimize the correctness computational resources, ensuring that the model only continues the reasoning process when necessary. Our experiments demonstrate that the proposed FoT framework, combined with these strategies, significantly improves the reasoning performance of LLMs, enabling them to solve complex tasks with greater precision and efficiency.

2. Related Works

2.1. XoT reasoning

Starting from Chain-of-Thought (CoT), XOT reasoning (e.g., ToT and GoT) began to be an important technique to enhance the reasoning ability of LLMs, leading to the development of a series of XoT reasoning algorithms.

Chain-of-Thought (CoT) (Wei et al., 2022) decomposes a problem into a series of intermediate steps, each of which provides a portion of the information needed for the final answer. This approach mimics human problem-solving strategies, involving step-by-step reasoning to reach a conclusion. However, while the CoT method performs well in many tasks, it has limitations when dealing with complex mathematical and logical problems. These intricate problems often require multidimensional, nonlinear thinking, rather than just sequential reasoning. Despite numerous subsequent studies that have improved the CoT, including Zero-Shot-CoT (Kojima et al., 2023), Self-Consistency with CoT (CoT-SC) (Wang et al., 2023), Auto-CoT (Zhang et al., 2022), VerifyCoT (Zhao et al., 2023), CoF-CoT (Nguyen et al., 2023), further exploration and optimization are still necessary to address highly complex tasks.

Least-to-Most Prompting (LtM) (Zhou et al., 2023) leads the model through a step-by-step process, progressively assisting it in constructing a solution, in contrast to methods like CoT that attempt to solve complex problems directly. This approach effectively avoids the reasoning errors that can arise when attempting to solve complex problems all at once. By decomposing complex problems into a series of simpler tasks, Program of Thought (PoT) (Chen et al., 2023), Chain of Code (CoC) (Li et al., 2024) and Buffer of Thought (BoT) (Yang et al., 2024) transform this process into a set of programmatic steps, using variable names to convey semantic information. On the other hand, Algorithm of Thought (AoT) (Sel et al., 2024) method aims to integrate these steps into a single prompt, enabling LLMs to learn how to break down problems, generate solutions, assess their feasibility, and determine the next step in the search process. This approach reduces token consumption and improves efficiency.

Tree-of-Thought (ToT) (Yao et al., 2024) constructs a tree structure to explore various possible choices and their outcomes, where each node represents a decision point and the edges represent transitions from one state to another. Typically, a depth-first search (DFS) approach is used to gradually explore each branch. Tree Prompting (Morris et al., 2023) establishes a decision-tree-based prompting system, chaining multiple language model calls together to collaboratively complete a specific task. However, for complex problems, the depth of the tree may become very large, leading to an exponential increase in the search space and an increased computational burden. Graph-of-Thought (GoT) (Besta et al., 2024) builds upon the ToT framework by introducing an aggregation process. GoT models the reasoning process of LLMs as a graph structure, allowing information units to form arbitrary dependencies, not limited to linear or tree-based arrangements. Through the aggregation process, GoT can consolidate information from multiple paths, supporting complex dynamic path selection and backtracking. Skeleton-of-Thought (SoT) (Ning et al., 2023) reduces generation latency in large language models (LLMs) by first generating a skeleton answer outline, then completing content in parallel, achieving significant speed-ups and potential quality improvements across various question types.

2.2. Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) (Chaslot et al., 2008), a probability-based search algorithm, has achieved significant progress across various domains since its introduction in computer Go in 2006. It evaluates nodes through randomized simulations (rollouts) and incrementally builds a local game tree to identify optimal or near-optimal solutions within limited time. To enhance MCTS performance, researchers have proposed various improvements. (Browne et al., 2012) surveyed extensions like sparse activation, dynamic pruning, parallelization, and distributed computation, broadening MCTS applications. (Srinivas et al., 2012) introduced UCB1-Tuned, an improved exploration-exploitation strategy suited for high-dimensional spaces. Recently, integrating MCTS with

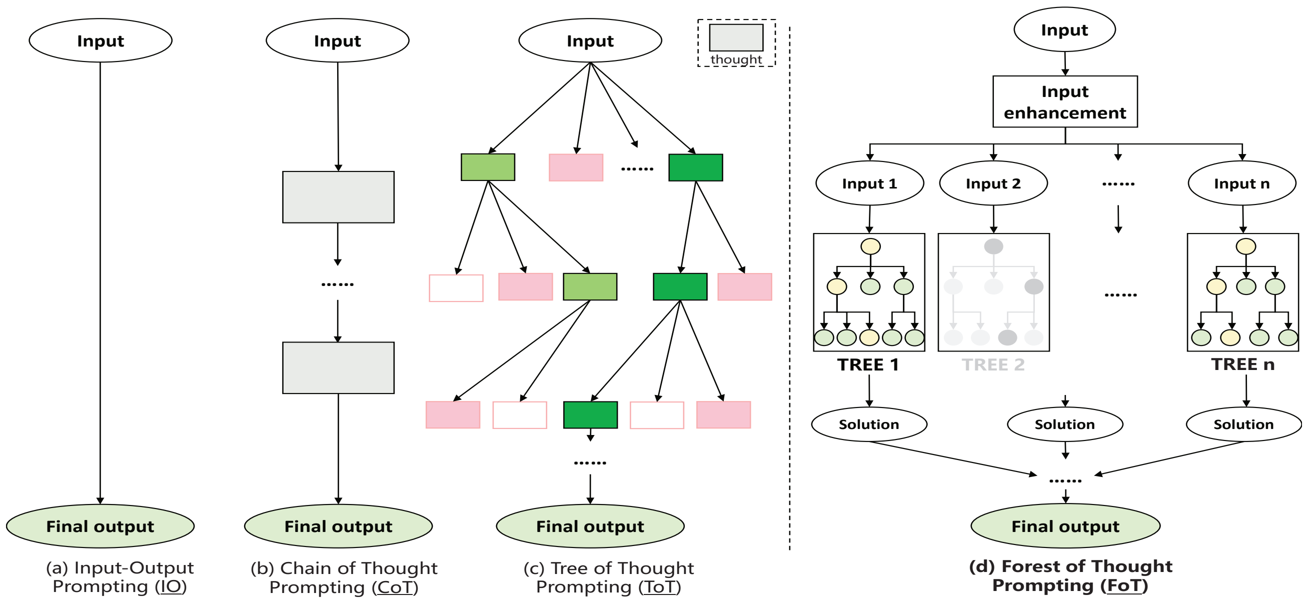
Input
Final output
(a) Input-Output Prompting (IO)
(b) Chain of Thought Prompting (CoT)
(c) Tree of Thought Prompting (ToT)
thought
Input enhancement
Input 1
Input 2
Input n
TREE 1
TREE 2
TREE n
Solution
(d) Forest of Thought Prompting (FoT)

Figure 1. Schematic illustration of various LLM reasoning approaches including IO prompting, CoT, ToT and the proposed FoT.

large language models has advanced its use in complex reasoning tasks. Methods like MCTSr (Zhang et al., 2024) combine MCTS with LLMs to guide decision-making, blending probabilistic search with linguistic reasoning. This hybrid approach enables effective multi-step logic and diverse solution paths.

3. Method

By introducing multiple reasoning trees (e.g., ToT (Yao et al., 2024) or MCTSr (Zhang et al., 2024)) for independent decision-making and employing sparse activation strategies to filter the results of key trees, we can construct an integrated framework known as the "forest of thought" to enhance the reasoning capability of LLMs, as shown in Figure 1(d) and Algorithm 1. This strategy leverages collective wisdom to compensate for individual deficiencies, thereby enhancing the model's ability to think comprehensively from multiple directions. Our experimental validation demonstrates that the practice of integrating the results of multiple reasoning trees through sparse activation strategies indeed enhances the reasoning capabilities of larger models to a certain extent. This discovery not only enriches our understanding of model integration methods but also offers novel insights and approaches for improving the mathematical reasoning abilities of large language models (LLMs).

3.1. The FoT Framework

Suppose we have n reasoning trees $T_1, T_2, \cdots, T_n$, each of which approaches the input problem from a different perspective. The root node of each tree represents the initial state or problem input, and each node represents an intermediate result or step in the reasoning process. Let the input be x. Each reasoning tree will start from the input and produce a result through different reasoning steps:

$$s_i = T_i(\varepsilon(x)), \quad i \in \{1, 2, \dots, n\} \tag{1}$$

where $\varepsilon(x)$ is the function that enhances x before it is used as input (details in the following paragraph) and $T_i(\cdot)$ represents the reasoning process of the ith tree. FoT will consider the results of these trees in a sparse activation manner and produce high-quality response via majority voting. A dynamic self-correction strategy is proposed for enhancing the accuracy (Sec. 3.2), and early stopping strategies are introduced for improving inference efficiency (Sec. 3.3).

Sparse Activation. In the reasoning process of a forest of reasoning trees, only the most relevant reasoning trees (or certain nodes within them) are selected for computation, rather than performing a complete calculation on all trees. This approach enhances efficiency while improving model accuracy by selecting the most relevant reasoning paths. Through sparse activation, we filter the activations of each reasoning tree, ensuring that only the paths of certain reasoning trees are "activated" for inference.

For each reasoning tree T_i, each layer considers the top-scoring nodes for further reasoning. Nodes with the highest scores are selected to split into child nodes at the next layer. If the nodes at a certain level of the tree cannot produce valid outputs, the tree's splitting process will terminate early, and the activation indicator value will be set to

0. Otherwise, the tree will continue splitting until the specified depth, and the activation indicator value will be set to 1. Formally, the activation indicator φ_i for the reasoning tree T_i can be represented as

$$\varphi_i = \begin{cases} 1, & \text{if tree } T_i \text{ is activated} \\ 0, & \text{if tree } T_i \text{ is not activated} \end{cases} \tag{2}$$

For each tree T_i, node-level activation is determined at each layer. Let the activation score of a node k at layer l of tree T_i be $s_{i,l,k}$. A node is activated if its score is among the top m scores at that layer:

$$\mathcal{A}_{i,l} = \{k \mid s_{i,l,k} \text{ is in the top } m \text{ at layer } l\} \tag{3}$$

where $\mathcal{A}_{i,l}$ is the set of activated nodes at layer l for tree T_i.

The output of the forest, incorporating both tree-level and node-level activations, is defined as:

$$y = \sum_{i=1}^{n} \varphi_i \cdot \left(\sum_{l=1}^{L_i} \sum_{k \in \mathcal{A}_{i,l}} f_{i,l,k}(x) \right), \tag{4}$$

where the total number of trees in the forest is indicated by n. The total number of layers in the tree T_i is indicated by L_i. In tree T_i, $f_{i,l,k}(x)$ denotes the reasoning function of node k at layer l. The set of activated nodes at layer l in tree T_i is indicated by $\mathcal{A}_{i,l}$.

Input Data Augmentation. When confronted with complex problems, our cognitive processes often transition from rapid and intuitive "fast thinking" to more in-depth and systematic "slow thinking". This shift serves not only to address immediate challenges but also to leverage the relevant prior knowledge stored in our brains, enabling a more comprehensive analysis and resolution of problems. The slow thinking process integrates and evaluates this prior knowledge, allowing us to examine the problem from multiple perspectives and arrive at more reasonable solutions.

Inspired by this efficient problem-solving approach in humans, we have adopted a similar methodology to enhance the problem-solving capabilities of large language models. Unlike traditional "few-shot learning", which merely presents a few examples for the model to imitate, this approach emphasizes extracting the most relevant portions from the model's extensive knowledge base to assist its understanding and problem-solving. Specifically, when presented with a question or task x, the model initially attempts to find an answer through responding quickly. If this process fails to yield a satisfactory solution, the model initiates a deeper cognitive mechanism, filtering out highly relevant background information and prior experiences from a knowledge base $\mathcal{B}$ with several pre-collected samples:

$$i_{max} = \arg\max_i \left(Sim \left(\text{TF}(x), \text{TF}(\mathcal{B}_i) \right) \right) \tag{5}$$

$$x = x \oplus \mathcal{B}_{i_{max}} \tag{6}$$

where $TF(\cdot)$ represents the TF-IDF transformation of the text which is fast yet effective, $Sim(\cdot)$ computes the cosine similarity between two vectors, and $\oplus$ means the concatenation of two text strings. This process mirrors human slow thinking, where the integration and analysis of pertinent information enable the model to construct a more complex cognitive framework, thereby enhancing its problem-solving capabilities and accuracy.

Algorithm 1 Forest of Tree (FoT)

```
Require: Input $x$, LLM $p_\theta$, $n$ reasoning trees $\{T_i()\},\ i = 1, 2, \cdots, n$;
 1: $\mathcal{S}_0 \leftarrow \{\}$
 2: for $i = 1, \cdots, n$ do
 3:     obtain result of the $i$-th tree with input enhancement:
        $s_i, \varphi_i \leftarrow T_i(\varepsilon(x))$;
 4:     if activator indicator $\varphi_i == 1$ then
 5:         Dynamic self-correction:
            $s'_i \leftarrow self\text{-}correct(s_i, x)$ (Sec. 3.2);
 6:         update result set: $\mathcal{S}_i \leftarrow \mathcal{S}_{i-1} \cup \{s'_i\}$;
 7:     else
 8:         continue;
 9:     end if
10: end for
    Return Decision Making $CGED(\mathcal{S}_n)$ (Sec. 3.3).
```

3.2. Dynamic Self-Correction Strategy

To improve the probability for giving correct answer in each tree, we introduce the dynamic self-correction strategy. For the initial result s_i from a reasoning tree, the self-correction strategy evaluates its correctness and effectiveness, and assigns corresponding scores upon completion of each reasoning step. The score can be obtained with a LLM p_θ:

$$score_i \sim p_\theta(score_i \mid s_i, x). \tag{7}$$

Once a step's score falls below the predefined threshold, the strategy automatically triggers a correction mechanism. This mechanism first reviews and analyzes past failure cases (denoted as $\mathcal{C}$) to identify the causes of low scores and common error patterns, subsequently attempting to correct errors and optimize the reasoning direction. The correction process can be performed by the LLM itself:

$$s'_i \sim p_{\theta_1}(s'_i \mid \mathcal{C}, s_i, x). \tag{8}$$

By employing this learning-from-history and real-time correction mechanism, the model not only avoids repeating mistakes on the same issues but also finds effective solutions to new and complex problems more swiftly and accurately. This continuous learning and optimization process significantly enhances the model's reasoning capabilities and efficiency, enabling it to perform more intelligently and effectively across a variety of complex tasks.

Especially for math tasks, we have incorporated a correction method based on mathematical rule matching defined as function $F(\cdot)$, in addition to the aforementioned mechanisms. The framework relies on predefined mathematical rules to perform rapid error correction: $s_i' = F(s_i)$, enabling it to quickly identify and rectify errors in mathematical expressions. For instance, the framework can detect common computational errors such as division by zero, mismatched parentheses, or misuse of symbols, and immediately correct them. This approach not only enhances the accuracy of the framework but also significantly reduces the time required for error detection and correction, making the entire reasoning process more efficient. The process details of the proposed dynamic self-correction strategy are shown in Algorithm 2.

Algorithm 2 Dynamic Self-Correction Strategy

```
Require: Input context $x$, LM $p_\theta$, mathematical rule correction function $F$, priori knowledge sets $\mathcal{C}$;
 1: $s_i \leftarrow p_\theta(s_i \mid x)$;
 2: $score_i \leftarrow p_\theta(score_i \mid s_i, x)$;
 3: if $score_i <$ threshold then
 4:     if $F$ is not None then
 5:         $s_i' \leftarrow F(s_i)$;
 6:     else
 7:         $s_i' \leftarrow p_\theta(s_i' \mid \mathcal{C}, s_i, x)$;
 8:     end if
 9:     update result: $s_i \leftarrow s_i'$;
10: end if
```

3.3. Decision Making Strategy

To address complex mathematical problems, we designed the Consensus-Guided Expert Decision (CGED) strategy to ensure high accuracy and reliability in the final answers generated by FoT. The CGED approach combines collective intelligence with expert judgment, guiding the reasoning process from consensus-based decision-making to expert evaluation.

Selecting the Optimal Leaf Node. In the FoT method, each independent tree generates one or more possible answers through its unique reasoning path. Subtrees vote for the candidate answer that receives the most support. If a consensus cannot be reached, the math expert evaluates the reasoning processes and selects the final answer to ensure its accuracy and validity.

FoT Decision-Making Process. During reasoning, each activated tree produces an optimal solution for its path. These answers are then filtered through majority voting and expert evaluation. For complex tasks, if answers from multiple trees conflict, an LLM expert examines the reasoning processes and makes a final decision based on their expertise. This process enhances robustness by reducing errors and inconsistencies.

4. Experiments

We evaluate the proposed FoT method on the widely-used LLM reasoning benchmarks including Game of 24, GSM8K and MATH.

4.1. Experimental setups

For the Game of 24, our FoT is built using the ToT as the reasoning tree. Beyond the ToT-based FoT, we developed an MCTSr-based FoT to address more complex and diverse mathematical problems, including those in the GSM8K and MATH benchmarks.

In all the three benchmarks, the Llama3-8B-Instruct (at Meta, 2024) is used as the base language model. In addition, we extend our method on Mistral-7B (Jiang et al., 2023), and GLM-4-9B (GLM et al., 2024) on GSM8K benchmark to verify its generalization.

4.2. Game of 24

The Game of 24 originates from ToT (Yao et al., 2024), where the objective is to construct an arithmetic expression using each of the four given numbers exactly once, such that the expression evaluates to 24. The 24-point game process is shown in Figure 2. We conducted multiple inference experiments on a V100-32G GPU using Llama3-8B, optimizing parameters such as the number of trees in the forest, activation mechanisms, and sparsity settings. We evaluated the impact of different configurations on inference speed and accuracy.

Results. In Table 1, we compare the results of increasing the number of leaf nodes at each layer in the ToT framework to allow for a greater diversity of potential reasoning outcomes. The hypothesis was that by expanding the leaf nodes, ToT would be able to explore more reasoning paths, leading to improved accuracy. However, when the number of leaf nodes per layer was increased to 32, the experimental results did not show significant improvement. It appears that the system reached a reasoning bottleneck, where further expansion of the leaf nodes did not lead to substantial gains in performance. This suggests that beyond a certain threshold, simply increasing the number of available reasoning paths does not necessarily translate into better reasoning outcomes, likely due to diminishing returns from excessive node expansion.

In contrast, FoT approach demonstrated a notable improvement in accuracy without a significant increase in computa-

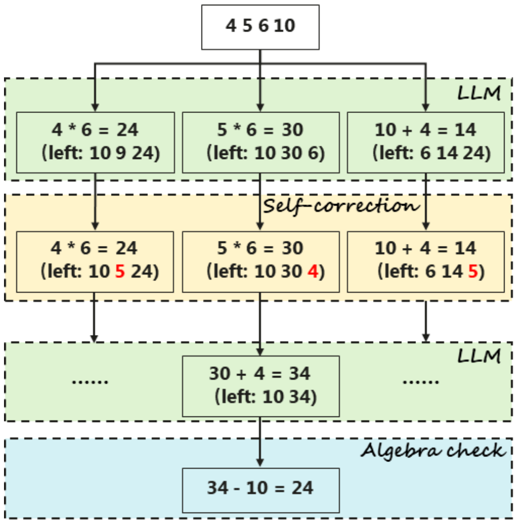
4 5 6 10
LLM
4 * 6 = 24 (left: 10 9 24)
5 * 6 = 30 (left: 10 30 6)
10 + 4 = 14 (left: 6 14 24)
Self-correction
4 * 6 = 24 (left: 10 5 24)
5 * 6 = 30 (left: 10 30 4)
10 + 4 = 14 (left: 6 14 5)
LLM
......
30 + 4 = 34 (left: 10 34)
......
Algebra check
34 - 10 = 24

Figure 2. Illustration of FoT in Game of 24 task.

tional cost. Specifically, when the number of trees was increased from 2 to 4, the accuracy of FoT improved by 14%, showing a significant boost in reasoning performance. Unlike the ToT method, which seemed to encounter a plateau as the number of leaf nodes increased. This suggests that the diversity of reasoning paths provided by multiple trees in FoT is more effective than simply expanding the complexity of individual trees, highlighting the advantages of the FoT framework in achieving scalable and efficient reasoning improvements.

4.3. GSM8K Benchmark

In addition to ToT, we evaluated the benefits of integrating multiple methods into the FoT framework on the GSM8K (Cobbe et al., 2021) dataset.

Results. As shown in Figure 3, we constructed forests using various methods, including Zero-Shot-CoT, MCTSr with one turn, MCTSr with 4 rollouts, and MCTSr with 8 rollouts. The experimental results demonstrate that as the number of trees in the forest increases, the advantages of the multi-method forest approach become more pronounced. Notably, the 4-rollouts MCTSr with 2 trees achieved 3.2% higher accuracy compared to the 8-rollouts MCTSr. Additionally, it outperformed the 8-rollouts MCTSr with 2 trees, highlighting its clear advantage. These findings indicate that increasing the diversity of reasoning outcomes has a more significant impact on performance than merely extending the depth of individual trees.

When using MCTSr with 2 trees in the FoT framework, the accuracy improvement is notably significant, highlighting the substantial benefit of introducing even a small amount of diversity in reasoning paths. However, as the number of trees continues to increase, the rate of improvement gradually diminishes. This phenomenon suggests that the initial addition of trees provides a major boost by expanding the range of explored solutions, but subsequent increases contribute less incrementally to the overall accuracy. The diminishing returns can be attributed to overlapping reasoning paths or the fact that additional trees may explore regions of the solution space that are already well-covered, thereby reducing their impact. This highlights the importance of balancing tree diversity and computational efficiency to achieve optimal performance in the FoT framework.

Table 1. Performance Comparison between ToT and FoT. ToT* denotes the application of the self-correction method. Average infer times refers to the mean number of inference operations performed by the ToT or FoT across the test set, calculated as the total number of inference operations divided by the number of test instances. b denotes the breadth limit of ToT, and n is the number of reasoning trees.

Method	Average infer times	Success
IO prompt	1	3%
CoT prompt	1	3%
ToT (b=5)	13.74	5.26%
FoT (b=1, n=5)	14.49	9.47%
ToT* (b=5)	13.74	56.25%
ToT* (b=8)	16.82	74.71%
ToT* (b=16)	18.92	74.74%
ToT* (b=32)	22.15	76.84%
FoT* (b=5, n=2)	19.04	77.89%
FoT* (b=5, n=4)	23.64	91.58%
FoT* (b=5, n=8)	25.64	96.84%
FoT* (b=5, n=16)	26.99	100%

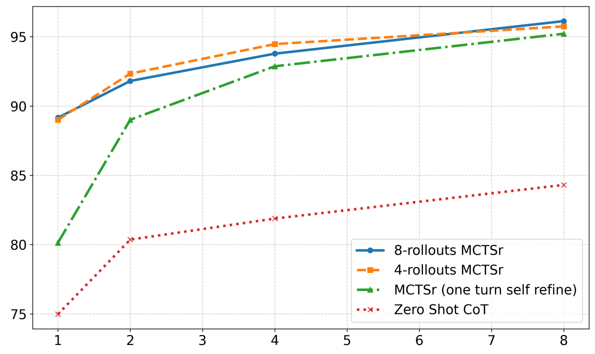

Figure 3. Benefit analysis of FoT: the return on wisdom growth. The x-axis represents the number of subtrees in the forest, while the y-axis indicates the accuracy on the GSM8K dataset.

Scaling laws in FoT across various base models. In addition to the previously discussed research, we extended our exploration to evaluate the performance of different base models of similar size when reasoning with FoT framework. Specifically, we conducted experiments using three mod-

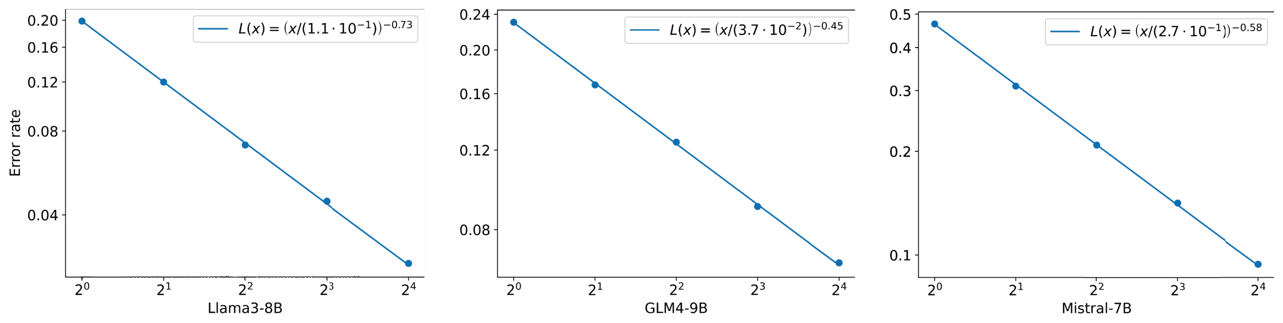

Figure 4. Comparative analysis of FoT gains across Llama, Mistral and GLM models. The x-axis represents the number of activated subtrees in FoT, while the y-axis indicates the error rate on the GSM8K dataset. The relationship between the number of activated subtrees in FoT and accuracy has been validated across multiple models, revealing a trend consistent with the scaling test-time comput for enhancing LLM reasoning.

els: Mistral-7B, Llama3-8B, and GLM-4-9B. These experiments aimed to assess how the FoT scales in accuracy as the number of activated subtrees increases.

As shown in Figure 4, the results demonstrated a clear scaling law: as the number of subtrees in FoT increased, model accuracy improved significantly. This scaling behavior aligns with the theoretical expectations that activating additional subtrees enhances the diversity of reasoning paths, enabling the model to refine its problem-solving capabilities. Each additional subtree contributed incrementally to the overall performance, collectively boosting the reasoning capacity of the framework.

The observed trend suggests that the relationship between the number of activated subtrees and accuracy is both positive and predictable, adhering to scaling law principles. This consistency indicates that the FoT method efficiently utilizes computational resources to maximize reasoning accuracy. Moreover, the experiments underscore that as computational capacity is allocated to activating more subtrees, the performance gains follow a diminishing-but-consistent trajectory, demonstrating the robustness and scalability of the FoT approach.

This study not only confirms the effectiveness of FoT across various base models but also provides a theoretical foundation for optimizing resource allocation in complex reasoning tasks. It highlights the adaptability of FoT to different architectures while adhering to scaling principles, making it a versatile tool for enhancing reasoning performance in large language models.

4.4. MATH Benchmark

This section presents the results of applying the FoT algorithm across various complexity levels on the MATH dataset (Hendrycks et al., 2021). The dataset is stratified into five difficulty levels, ranging from level 1 (easiest) to level 5 (most challenging).

Table 2. FoT's experimental findings on the math dataset. The results of our experimental reproduction are indicated by MCTSr#. The number of subtrees in FoT that are activated is denoted by n. LLaMA3-8B-Instruct is utilized in the experiments.

Data	**Zero-Shot CoT**	**MCTSr#**	**FoT** (n=2)	**FoT** (n=4)
Level-1 437	57.21	67.51	73.68	**77.35**
Level-2 894	40.60	47.43	52.01	**57.27**
Level-3 1131	27.32	34.22	39.35	**45.09**
Level-4 1214	16.64	21.66	26.28	**31.05**
Level-5 1324	7.10	8.38	11.10	**14.50**
MATH 5000	24.36	29.74	33.88	**38.42**

Results. The experimental results are shown in Table 2. The FoT (Forest of Thought) method demonstrates a consistent improvement over the MCTSr across all difficulty levels from 1 to 5 in the mathematics dataset. FoT (n=4) shows about 10% improvement from Level-1 to Level-5. This consistent enhancement highlights the effectiveness of the FoT method in handling problems of varying complexity, from the simplest to the most challenging tasks.

4.5. Ablation Studies of Stopping Strategies

This paper compares three methods for selecting the optimal leaf node from subtrees, highlighting their effectiveness and performance:

Random Selection: A leaf node is selected randomly as the final answer. This method is simple and requires minimal computation but often results in unstable and less accurate outcomes.

Table 3. Subtree decision-making process. "Random" refers to selecting a random answer from the leaf nodes as the tree's answer, while "Score" indicates that the answer is chosen based on the highest score among the leaf nodes. The accuracy of the experimental results on the GSM8K benchmark is displayed. The average inference times (avg infer times) typically refers to the ratio of the total number of inference steps to the number of test samples in GSM8K.

Method	Accuracy	Avg infer times
Random	77.73	7.00
Score	77.86	7.00
CGED	**78.62**	7.28

Score-based Selection: The leaf node with the highest score is chosen as the final answer. The score is derived from the maximum value retained during MCTSr node expansion, reflecting the most promising path explored.

CGED Selection: This method integrates collective reasoning with expert judgment. It considers leaf node scores while incorporating expert evaluation, ensuring optimal decisions even in complex or ambiguous cases. By synthesizing insights from multiple reasoning paths, CGED enhances accuracy and reliability.

Experimental results in Table 3 show that CGED significantly outperforms the other methods, achieving 0.9% higher accuracy than Random Selection and 0.8% higher than Score-based Selection. While Random Selection offers simplicity, its randomness reduces accuracy. In contrast, CGED demonstrates greater stability and precision, particularly for complex, multi-dimensional problems.

In Figure 5, we compare the forest stopping strategies based on Majority Vote, Math Expert, and CGED. In the FoT method, when two subtrees are activated, the CGED strategy demonstrates similar accuracy to the Majority Vote and Math Expert strategies. However, as the number of activated subtrees increases, the performance differences between these strategies become more pronounced. Notably, with five activated subtrees, the CGED strategy achieves a clear improvement in accuracy, outperforming the Majority Vote and Math Expert strategies by 2%. This highlights the effectiveness of the CGED method in scenarios with a higher number of activated subtrees.

4.6. Ablation Studies of Self-Correction methods

The advantages of incorporating a dynamic self-correction module into the Zero-Shot-CoT and ToT approaches are summarized in Table 4. Specifically, adding this dynamic adjustment mechanism improves the efficacy of the CoT method by 5%. Remarkably, the accuracy of the ToT approach increases from an error-prone initial state in single-step reasoning to over 50% after integrating the self-correction mechanism. This improvement is particularly significant because errors in ToT's single-step reasoning tend to propagate through subsequent reasoning steps, severely impairing overall performance. Consequently, the dynamic self-correction process plays a critical role in ToT by promptly addressing early-stage mistakes, preventing their compounding effects, and significantly enhancing the final accuracy.

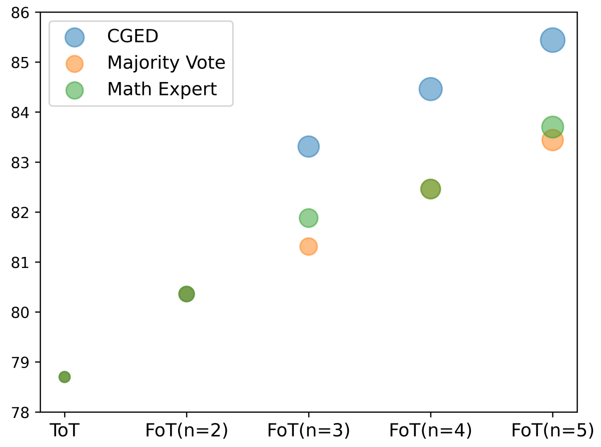

Figure 5. Stopping strategies in FoT. The x-axis represents the number of activated subtrees in the FoT, while the y-axis indicates the accuracy achieved on the GSM8K dataset.

Table 4. Results of dynamic self-correction strategies. * denotes the application of the self-correction method. The experiments were conducted using the Llama-3-8B-Instruct model.

Method	Benchmark	Accuracy
zero-shot-cot	GSM8K	74.98
zero-shot-cot*	GSM8K	**79.98**
ToT	Game of 24	5.26
ToT*	Game of 24	**56.25**

5. Conclusion

This paper introduces a novel method, the Forest of Thought (FoT), aimed at significantly enhancing the reasoning capabilities of large language models (LLMs). FoT leverages a structured framework that integrates multi-path exploration and dynamic activation of reasoning paths, addressing key limitations in existing LLM reasoning paradigms. This enables the model to achieve robust and efficient problem-solving across complex tasks while generating diverse reasoning outcomes without relying on backpropagation or fine-tuning.

By integrating multiple independent "thinking" models, such as the Tree of Thought and Monte Carlo Trees, FoT effectively tackles complex problems and incorporates a sparse activation mechanism to substantially improve computational efficiency. Additionally, this paper systematically investigates the scaling relationship between reasoning time and accuracy in LLMs, providing a theoretical foundation for optimizing their reasoning performance.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

at Meta, A. Introducing meta llama 3: The most capable openly available llm to date. 2024. URL https://ai.meta.com/blog/meta-llama-3/.

Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012. doi: 10.1109/TCIAIG.2012.2186810.

Chaslot, G., Bakkes, S., Szita, I., and Spronck, P. Monte-carlo tree search: a new framework for game ai. In *Proceedings of the Fourth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE'08, pp. 216–217. AAAI Press, 2008.

Chen, W., Ma, X., Wang, X., and Cohen, W. W. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. 2023. URL https://arxiv.org/abs/2211.12588.

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

GLM, T., :, Zeng, A., Xu, B., Wang, B., Zhang, C., Yin, D., Zhang, D., Rojas, D., Feng, G., Zhao, H., Lai, H., Yu, H., Wang, H., Sun, J., Zhang, J., Cheng, J., Gui, J., Tang, J., Zhang, J., Sun, J., Li, J., Zhao, L., Wu, L., Zhong, L., Liu, M., Huang, M., Zhang, P., Zheng, Q., Lu, R., Duan, S., Zhang, S., Cao, S., Yang, S., Tam, W. L., Zhao, W., Liu, X., Xia, X., Zhang, X., Gu, X., Lv, X., Liu, X., Liu, X., Yang, X., Song, X., Zhang, X., An, Y., Xu, Y., Niu, Y., Yang, Y., Li, Y., Bai, Y., Dong, Y., Qi, Z., Wang, Z., Yang, Z., Du, Z., Hou, Z., and Wang, Z. Chatglm: A family of large language models from glm-130b to glm-4 all tools. 2024. URL https://arxiv.org/abs/2406.12793.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. 2021. URL https://arxiv.org/abs/2103.03874.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b. 2023. URL https://arxiv.org/abs/2310.06825.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. 2023. URL https://arxiv.org/abs/2205.11916.

Li, C., Liang, J., Zeng, A., Chen, X., Hausman, K., Sadigh, D., Levine, S., Fei-Fei, L., Xia, F., and Ichter, B. Chain of code: Reasoning with a language model-augmented code emulator. 2024. URL https://arxiv.org/abs/2312.04474.

Morris, J. X., Singh, C., Rush, A. M., Gao, J., and Deng, Y. Tree prompting: Efficient task adaptation without fine-tuning. 2023. URL https://arxiv.org/abs/2310.14034.

Nguyen, H. H., Liu, Y., Zhang, C., Zhang, T., and Yu, P. S. Cof-cot: Enhancing large language models with coarse-to-fine chain-of-thought prompting for multi-domain nlu tasks. 2023. URL https://arxiv.org/abs/2310.14623.

Ning, X., Lin, Z., Zhou, Z., Wang, Z., Yang, H., and Wang, Y. Skeleton-of-thought: Large language models can do parallel decoding. *Proceedings ENLSP-III*, 2023.

OpenAI. Openai o1 system card. 2024. Accessed: 2024-9-12.

Sel, B., Al-Tawaha, A., Khattar, V., Jia, R., and Jin, M. Algorithm of thoughts: Enhancing exploration of ideas in large language models. 2024. URL https://arxiv.org/abs/2308.10379.

Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. W. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, May 2012. ISSN 1557-9654. doi: 10.1109/tit.2011.2182033. URL http://dx.doi.org/10.1109/TIT.2011.2182033.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. 2023. URL https://arxiv.org/abs/2203.11171.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837, 2022.

Yang, L., Yu, Z., Zhang, T., Cao, S., Xu, M., Zhang, W., Gonzalez, J. E., and Cui, B. Buffer of thoughts: Thought-augmented reasoning with large language models. 2024. URL https://arxiv.org/abs/2406.04271.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Zhang, D., Huang, X., Zhou, D., Li, Y., and Ouyang, W. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. 2024. URL https://arxiv.org/abs/2406.07394.

Zhang, Z., Zhang, A., Li, M., and Smola, A. Automatic chain of thought prompting in large language models. 2022. URL https://arxiv.org/abs/2210.03493.

Zhao, R., Li, X., Joty, S., Qin, C., and Bing, L. Verify-and-edit: A knowledge-enhanced chain-of-thought framework. 2023. URL https://arxiv.org/abs/2305.03268.

Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q., and Chi, E. Least-to-most prompting enables complex reasoning in large language models. 2023. URL https://arxiv.org/abs/2205.10625.

A. Example Prompts - Game of 24

In the 24-point game, we have designed three types of prompts to guide the problem-solving process:

1. **Propose Prompt**: This prompt is intended to assist players in breaking down the original problem in the first step. It guides players to select two numbers from the given four for calculation, thereby simplifying the problem to the remaining two unused numbers.
2. **Step-2 Prompt**: Based on the result of the first step, this prompt further guides players to choose two numbers from the remaining three for calculation, ultimately simplifying the problem to just one number left.
3. **Value Prompt**: This prompt is used to score each step's response, ensuring that each solution is both reasonable and efficient.

Table 5. Game of 24 Propose Prompt

Propose Prompt:
Let's play a game called 24. You'll be given four integers, and your objective is to use each number only once, combined with any of the four arithmetic operations (addition, subtraction, multiplication, and division) and parentheses, to achieve a total of 24.
Provide four integers as input. Randomly pick 2 full permutations of the 4 input numbers, perform all four basic arithmetic operations (addition, subtraction, multiplication and division), and list the results and remaining integers after each operation.
See <Examples> See Table 8 (Propose Prompt) </Examples>
Input: '4 5 6 10'
Possible next steps:

Response:
4 + 5 = 9 (left: 6 10 9),
10 - 4 = 6 (left: 6 5 6),
5 - 6 = -1 (left: -1 4 10),
4 * 6 = 24 (left: 24 5 10),
10 / 5 = 2 (left: 2 4 6),
4 - 10 = -6 (left: -6 5 6),
5 - 4 = 1 (left: 1 6 10),
10 - 5 = 5 (left: 5 4 6)

Table 6. Game of 24 Value Prompt

Value Prompt:
Evaluate if given numbers can reach 24 (sure/likely/impossible).
See <Examples> See Table 8 </Examples>
Input: '4 + 5 = 9 (left: 6 10 9)'
Response:
impossible

Value Prompt:
Evaluate if given numbers can reach 24 (sure/likely/impossible).
See <Examples> See Table 8 </Examples>
Input: '4 - 10 = -6 (left: -6 5 6)'
Response:
sure

Table 7. Game of 24 Step-2 Prompt

Step-2 Prompt:
Provide three integers as input. Randomly pick 2 full permutations of the 3 input numbers, perform all four basic arithmetic operations (addition, subtraction, multiplication and division), and list the results and remaining integers after each operation.
See <Examples> See Table 8 (Step-2 Prompt) </Examples>
Input: -6 5 6
Possible next steps:

Response:
-6 + 5 = -1 (left: 5 -1)
5 * 6 = 30 (left: 6 30)
-6 / 5 = -1.2 (left: 5 -1.2)
6 - 5 = 1 (left: 1 5)
5 - 6 = -1 (left: -1 5)

B. Example Prompts - GSM8K and MATH

We present the prompts utilized for the GSM8K task, since the prompts for the MATH task are essentially the same, with only minor variations in how answers are extracted. In Table 10, we present the results of incorporating a self-correction prompt mechanism into the reasoning process. This mechanism is designed to enhance the quality of the answers generated by the model. The self-correction prompt operates in two stages: first, it assigns a score to the generated answer based on predefined evaluation criteria, such as logical consistency, accuracy, and relevance. Then, depending on the score, the model is prompted to revise or correct its answer if the evaluation indicates potential flaws or suboptimal reasoning.

By iteratively refining its output, the model learns to identify errors and generate more accurate and coherent responses. This approach not only improves the reliability of the results but also provides a structured framework for integrating feedback into the reasoning process. The experiments summarized in Table 10 demonstrate that the self-correction prompt significantly enhances the model's performance across various tasks, reducing error rates and improving overall answer quality. This highlights the potential of self-correction as a powerful tool for boosting reasoning robustness in complex scenarios.

Table 8. Game of 24 problem decomposition example.

Propose Prompt
<Examples >
Input: 2 8 8 14
Possible next steps:
2 + 8 = 10 (left: 8 10 14)
8 / 2 = 4 (left: 4 8 14)
14 + 2 = 16 (left: 8 8 16)
2 * 8 = 16 (left: 8 14 16)
8 - 2 = 6 (left: 6 8 14)
2 - 8 = -6 (left: -6 8 14)
14 - 8 = 6 (left: 2 6 8)
14 / 2 = 7 (left: 7 8 8)
14 - 2 = 12 (left: 8 8 12)
2 * 14 = 28 (left: 8 8 28)
</Examples>
Step-2 Prompt
<Examples >
Input: 2 8 8
Possible next steps:
2 + 8 = 10 (left: 8 10)
8 / 2 = 4 (left: 4 8)
2 * 8 = 16 (left: 8 16)
8 - 2 = 6 (left: 6 8)
2 - 8 = -6 (left: -6 8)
</Examples>
Algebra Check
Input: 8 10
Output:
8 + 10 = 18
8 - 10 = -2
8 * 10 = 80
8 / 10 = 0.8
impossible

In Table 11, we introduce a prompt mechanism designed for expert selection, aimed at further refining the model's outputs in scenarios involving multiple plausible answers. When multiple answers align with majority consistency, a mathematics expert evaluates these responses based on domain-specific knowledge and selects the most appropriate or optimal answer.

This approach leverages expert judgment to ensure the final selected answer not only adheres to logical consistency but also aligns with mathematical rigor and best practices. By integrating expert selection into the prompt design, we enhance the reliability and precision of the model's reasoning process.

Table 9. Game of 24 value examples.

Value Prompt
<Examples >
Input: 4 10 30
(30 + 4) - 10 = 24
sure
Input: 4 9 11
9 + 11 + 4 = 20 + 4 = 24
sure
Input: 5 7 8
5 + 7 + 8 = 12 + 8 = 20
(8 - 5) * 7 = 3 * 7 = 21
I cannot obtain 24 now, but numbers are within a reasonable range
likely
Input: 5 6 6
5 + 6 + 6 = 17
(6 - 5) * 6 = 1 * 6 = 6
I cannot obtain 24 now, but numbers are within a reasonable range
likely
Input: 10 10 11
10 + 10 + 11 = 31
(11 - 10) * 10 = 10
10 10 10 are all too big
impossible
Input: 1 3 3
1 * 3 * 3 = 9
(1 + 3) * 3 = 12
1 3 3 are all too small
impossible
Input: -1 4 7
(-1 + 7) * 4 = 24
sure
Input: 4 10 30
30 + 4 - 10 = 24
sure
</Examples>

C. Evaluating the Scaling of FoT Compute Across Different Baseline Methods.

Tabel 6 demonstrate a clear trend: as the number of activated subtrees grows, the error rate decreases, reflecting the enhanced reasoning capability and robustness of the FoT approach. The scaling law comparison across various methods underscores the efficiency of FoT in utilizing additional computational resources, particularly when contrasted with other models that exhibit diminishing returns or slower error reduction as complexity increases. This suggests that the FoT framework scales more effectively with increased subtree activation, making it a powerful tool for addressing challenging reasoning tasks like those in GSM8K.

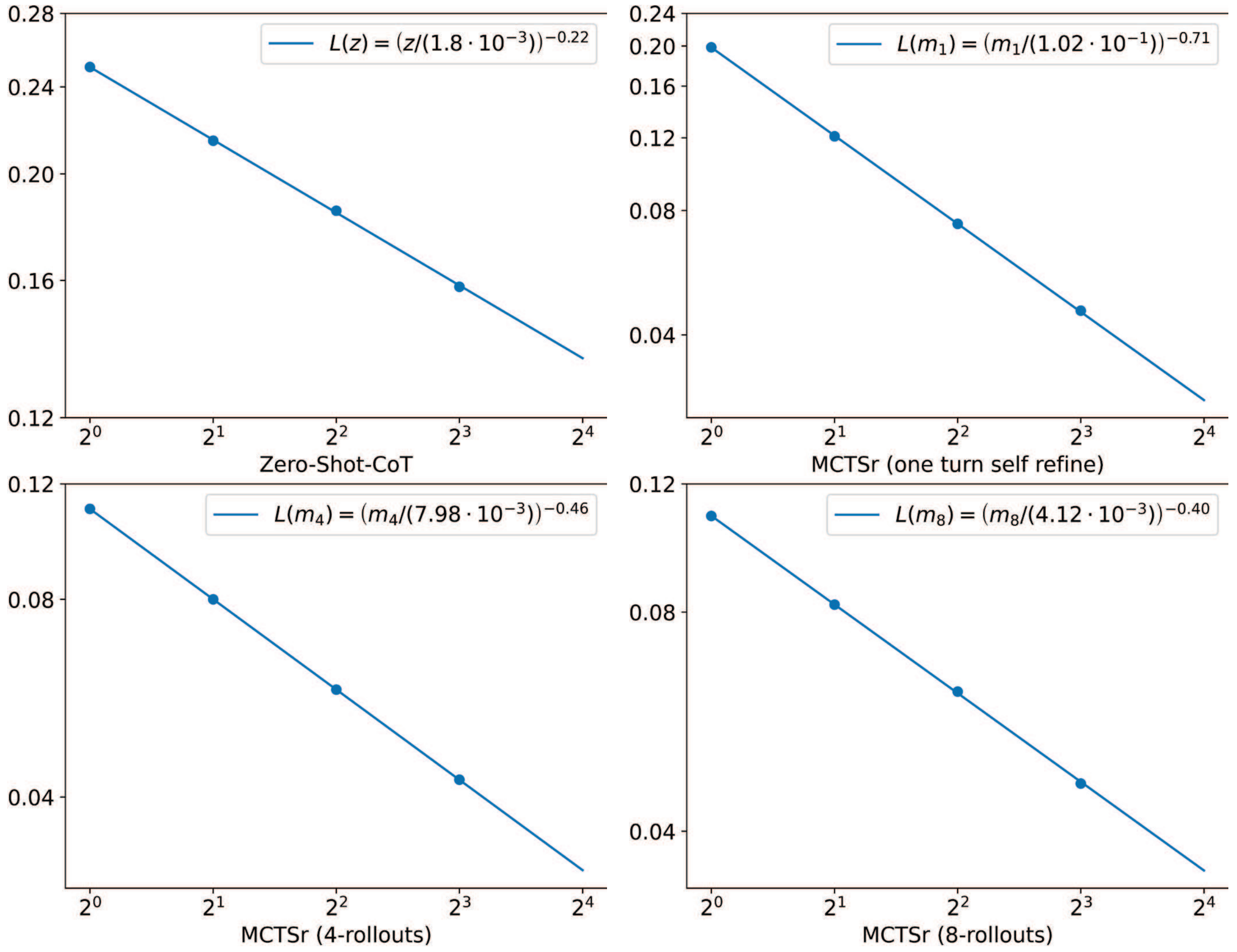

Figure 6. The comparison of the scaling laws for FoT across different methods (eg.zero-shot-cot, MCTSr (one-turn-self-refine), 4-rollouts MCTSr, 8-rollouts MCTSr) is presented, with the x-axis representing the number of activated subtrees within the FoT framework and the y-axis indicating the error rate on the GSM8K benchmark dataset.

Table 10. Self-Correction Prompt

Value Prompt: Question: {question} Answer:{answer} Analyze this answer Strictly and Critic, point out every flaw for ervery possible imperfect to minus every possible score! You need to be very harsh and mean in calculating grades, and never give full marks to ensure that the marks are authoritative. Output a score between [0,100], ig. from 0 to 100. Response format:[Analyst]...[Score]...
Self-Correction Prompt: ”Question: {question} Please refine the your answer according to your Reflection or Feedback. The response should begin with [reasoning process]...[Verification]... and end with end with <ans_format> Let’s think step by step.”

Table 11. Math Expert Prompt

Math Expert Prompt: ”You are a highly specialized mathematics expert, proficient in solving mathematical problems, and always able to select the most accurate answer from the given options. **Question:** {question} **Answers:** {answers_list} Which of the following answers is the most accurate? The response should begin with <ans_format>.”