

Pheromone-based Learning of Optimal Reasoning Paths

Anirudh Chari^{1,2*} Aditya Tiwari^{1*} Richard Lian^{1,2} Suraj Reddy^{1,2} Brian Zhou^{1,3}

Abstract

Large Language Models (LLMs) have demonstrated remarkable reasoning capabilities through chain-of-thought prompting, yet discovering effective reasoning methods for complex problems remains challenging due to the vast space of possible intermediate steps. We introduce Ant Colony Optimization-guided Tree of Thought (ACO-ToT), a novel algorithm that combines ACO with LLMs to discover optimal reasoning paths for complex problems efficiently. Drawing inspiration from Hebbian learning in neurological systems, our method employs a collection of distinctly fine-tuned LLM “ants” to traverse and lay pheromone trails through a centralized tree of thought, with each ant’s movement governed by a weighted combination of existing pheromone trails and its own specialized expertise. The algorithm evaluates complete reasoning paths using a mixture-of-experts-based scoring function, with pheromones reinforcing productive reasoning paths across iterations. Experiments on three challenging reasoning tasks (GSM8K, ARC-Challenge, and MATH) demonstrate that ACO-ToT performs significantly better than existing chain-of-thought optimization approaches, suggesting that incorporating biologically inspired collective search mechanisms into LLM inference can substantially enhance reasoning capabilities.

1. Introduction

Large language models (LLMs) have demonstrated remarkable capabilities in emulating human-like behavior across diverse tasks (Brown et al., 2020). These models process sequences of tokens through attention mechanisms, achieving strong performance on mathematical, log-

ical, and commonsense reasoning benchmarks (Wei et al., 2023). However, their widespread adoption faces two critical limitations: computational efficiency and reasoning accuracy (Chowdhery et al., 2022). Specifically, their reasoning capabilities remain constrained by token-level processing within the attention architecture, necessitating enhanced reasoning mechanisms for applications requiring exploration, strategic planning, or deterministic initial steps (Yao et al., 2023).

Recent research has shifted from black-box approaches toward interpretable methodologies grounded in cognitive science principles (Ling et al., 2017). Chain-of-Thought (CoT) prompting exemplifies this transition by explicitly generating intermediate reasoning steps before producing final outputs (Wei et al., 2023). This method implements a verification mechanism where the model evaluates its own reasoning, significantly improving accuracy on complex tasks (Kojima et al., 2023). However, naive CoT effectively only considers a single approach to solving a problem. Meanwhile, humans may consider many possible approaches to a problem before deciding upon a productive reasoning method (Newell & Simon, 1972), which points toward the notion of CoT “optimization.”

To build upon CoT’s initial success and address this key limitation, researchers have developed various specialized frameworks for structured exploration of the *reasoning space*. Notable derivatives include Tree-of-Thoughts (ToT) (Yao et al., 2023), Graph-of-Thoughts (GoT) (Besta et al., 2024), and Iterative Reasoning Preference Optimization (IRPO) (Bai et al., 2022). These methodologies transcend CoT’s linear progression by implementing branching, backtracking, and systematic space exploration, more accurately reflecting human problem-solving strategies. However, as reasoning tasks grow in complexity, these methods incur substantial computational overhead due to the exponential growth in intermediate reasoning steps required for accurate solutions (Yao et al., 2023).

Hebbian learning (Hebb, 1949) provides a paradigm for understanding pathway optimization in biological neural networks. The principle, commonly expressed as “neurons that fire together, wire together,” describes how repeated activation of sequences of neurons strengthens synaptic connections between those neurons, establishing preferential

*Equal contribution ¹a37.ai, San Francisco, CA, USA

²Massachusetts Institute of Technology, Cambridge, MA, USA

³Harvard University, Cambridge, MA, USA. Correspondence to: Anirudh Chari <anichari@mit.edu>.

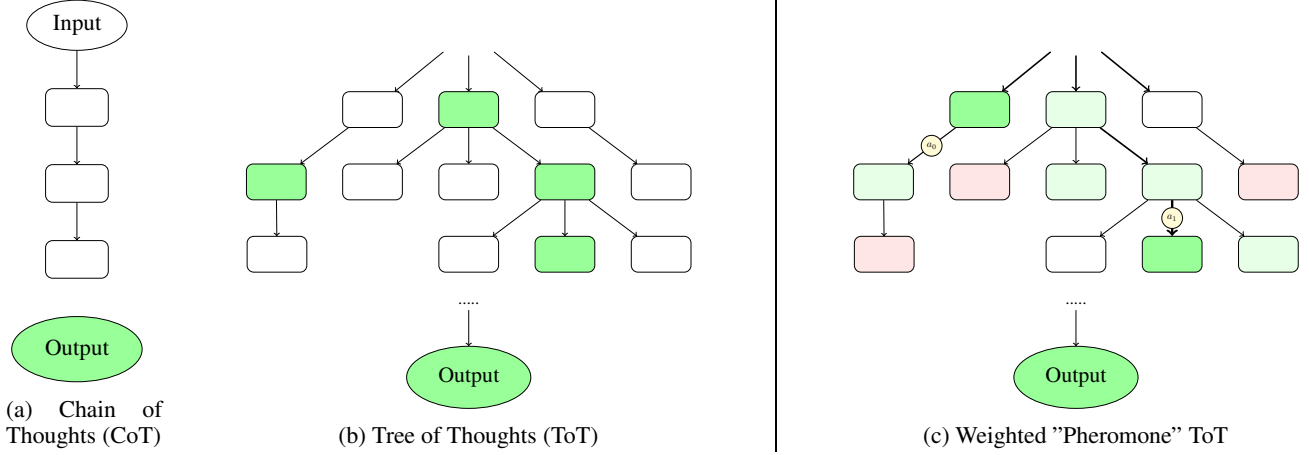


Figure 1. Comparison of approaches to complex reasoning problems with LLMs. Each rectangular node represents a *thought*—an intermediate reasoning step to solve a larger problem. On the right, our method utilizes “ants” traversing between connected reasoning steps (depicted as circles) to strengthen productive reasoning steps across iterations via pheromone trails. See algorithmic implementation for 1c in Figure 2.

pathways for efficient signal propagation (Löwel & Singer, 1992).

We propose adapting this neurological optimization principle to LLM reasoning through Ant Colony Optimization (ACO), which we claim sufficiently resembles Hebbian learning mechanics. Our implementation, ACO-ToT, employs specialized fine-tuned LLMs as artificial ants traversing a reasoning space. These LLM-ants deposit virtual pheromones proportional to reasoning quality. The system implements a probabilistic path selection mechanism, where pheromone concentration influences path choice while maintaining exploration-exploitation balance through stochastic selection. This collective intelligence approach gradually converges on optimal reasoning strategies while pruning inefficient paths (Blum, 2005).

Hence, our main contributions are:

- A neuroscience-inspired paradigm for search and optimization across the natural-language reasoning space via pheromone mechanics,
- **ACO-ToT**, a mixture-of-experts algorithm applying the above paradigm for dynamic CoT optimization using artificial LLM-based ants, and
- Extensive experimental validation of ACO-ToT demonstrating a mean absolute accuracy improvement of **16.6%** over existing approaches.

Section 2 contextualizes the integration of Hebbian learning principles with Tree of Thoughts prompting. Sections 3 and 4 detail our implementation methodology and analyze theoretical properties, including computational complexity. Section 5 presents our experimental protocol using

the GSM8K, ARC-Challenge, and MATH datasets, while Section 6 provides comprehensive results and analysis for comparison with other flagship methods and ablation studies for accuracy optimization. Section 7 describes related works in CoT prompting, LLM inference optimization, biological inspiration in AI, mixture of experts, and prompting techniques for increasing reasoning. Finally, Section 8 concludes our paper with a summary of findings and suggests future directions.

2. Background

2.1. Chain of Thought

Chain-of-thought (CoT) prompting enables language models to break down complex reasoning tasks into intermediate steps, significantly improving their problem-solving capabilities (Wei et al., 2023). Given an input x and desired output y , CoT introduces a sequence of intermediate thoughts z_1, \dots, z_n that bridge the reasoning gap. Each thought z_i represents a coherent language sequence sampled from the distribution $z_i \sim \pi_\theta(z_i | x, z_1, \dots, z_{i-1})$, where π_θ denotes the language model with parameters θ . The key innovation of CoT lies in its ability to decompose multi-step problems into manageable intermediate steps, allowing models to allocate computational resources according to problem complexity. This decomposition provides interpretable insights into the model’s reasoning process and enables debugging of incorrect solutions. Empirically, CoT prompting has demonstrated significant improvements across arithmetic, commonsense, and symbolic reasoning tasks (Huang et al., 2022).

2.2. Tree of Thought

Tree of Thought (ToT) extends CoT by enabling exploration of multiple reasoning paths simultaneously (Yao et al., 2023). Rather than generating a single chain, ToT maintains a tree where each node represents a state $s = [x, z_{1:i}]$ containing the input and sequence of thoughts thus far. At each state, ToT generates k candidate next thoughts and evaluates their promise toward solving the problem. The ToT framework comprises four key components: (1) thought decomposition into appropriate semantic units, (2) thought generation through sampling or sequential proposal, (3) state evaluation via independent scoring or comparative voting, and (4) tree search algorithms like breadth-first or depth-first search to explore the reasoning space. This deliberate exploration allows ToT to overcome limitations of left-to-right decoding by considering multiple paths and backtracking when necessary.

2.3. Ant Colony Optimization

Ant Colony Optimization (ACO) is a metaheuristic inspired by the foraging behavior of ant colonies (Dorigo & Di Caro, 1999). In ACO, artificial ants traverse a graph representing possible solutions, depositing pheromone trails proportional to solution quality. The probability p_{ij} of an ant choosing edge (i, j) is given by:

$$p_{ij} = \frac{(\tau_{ij})^\alpha (\eta_{ij})^\beta}{\sum_{l \in N_i} (\tau_{il})^\alpha (\eta_{il})^\beta} \quad (1)$$

where τ_{ij} is the pheromone level, η_{ij} is a heuristic value, α and β are parameters controlling their relative importance, and N_i is the set of available next nodes. After each iteration, pheromone levels are updated according to:

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \sum_{k=1}^m \Delta\tau_{ij}^k \quad (2)$$

where ρ is the evaporation rate and $\Delta\tau_{ij}^k$ is the pheromone deposited by ant k . The pheromone reinforcement mechanism in ACO resembles Hebbian learning in biological neural networks, where repeatedly activated synaptic pathways become stronger over time (Ye et al., 2023). This parallel suggests that pheromone-based path selection could naturally extend to reasoning-space search in a ToT, where promising cognitive trajectories are strengthened through repeated traversal while unproductive paths decay through evaporation. Figure 1 visually highlights the main differences between CoT, ToT, and ACO-ToT during reasoning-space exploration.

2.4. Mixture of Experts

Mixture of Experts (MoE) for LLMs leverages multiple specialized models to collaboratively work on complex

tasks. In the context of LLMs, MoE architectures consist of a set of “expert” models, each fine-tuned for specific reasoning types or domains, and a routing mechanism that determines how to weigh expert outputs for a given input (Si et al., 2023). The advantages of incorporating MoE in LLM reasoning motivate us to consider LLMs as “ants” during reasoning-space exploration (Yao et al., 2023).

3. Methods

3.1. Algorithm Overview

We propose ACO-guided Tree of Thought (ACO-ToT), a novel algorithm that combines the exploration capabilities of ToT with the collective intelligence of ACO to discover optimal reasoning paths. The algorithm employs a collection of specialized LLM “ants” to traverse a reasoning graph generated by a central LLM, converging on high-quality CoTs through iterative exploration and reinforcement. For a general overview of the algorithm, see Figure 2.

3.2. Reasoning Graph Construction

Given a problem input x , we first prompt a central LLM π_c to generate a tree of thought \mathcal{T} . We augment \mathcal{T} with special start node s_0 and end node s_f to form a directed graph $G = (V, E)$, where vertices V represent reasoning states and edges E represent transitions between states. Each state $s_i \in V$ contains the problem input and accumulated reasoning chain: $s_i = [x, z_{1:i}]$. Psuedocode for ToT generation is found as Algorithm 2.

3.3. LLM Ant Colony

Motivated by MoE architectures, we maintain a collection of m distinctly fine-tuned LLMs $\{\pi_1, \dots, \pi_m\}$ that serve as specialized “ants”. Each LLM π_k is fine-tuned on different aspects of reasoning, providing diverse expertise. At each timestep t , ant k at state i chooses its next state j with probability:

$$p_{ij}^k = \frac{(\tau_{ij})^\alpha (h_{ij}^k)^\beta}{\sum_{l \in N_i} (\tau_{il})^\alpha (h_{il}^k)^\beta} \quad (3)$$

where τ_{ij} is the pheromone level on edge (i, j) , h_{ij}^k is the heuristic value computed by prompting LLM π_k for its assessment of state j , and N_i is the set of available next states.

3.4. Path Evaluation and Pheromone Update

For a complete path $P = (s_0, \dots, s_f)$, i.e., a chain of thought, we compute its quality $Q(P)$ as:

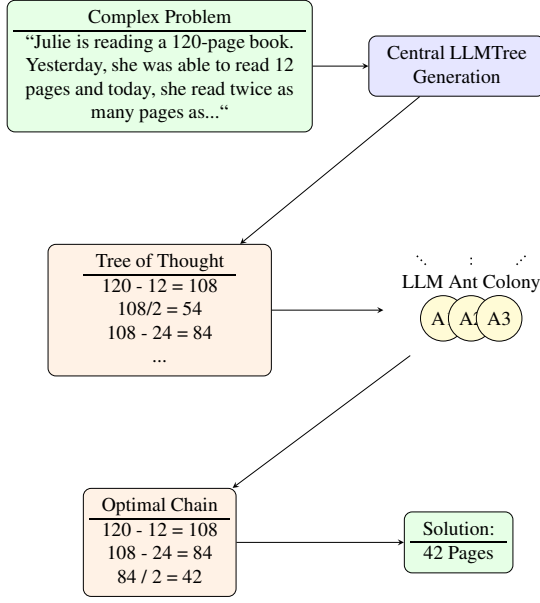


Figure 2. Example procedure for a math problem from GSM8K. The central LLM is prompted for an initial ToT, which is next explored by fine-tuned ant LLMs to discover an optimal reasoning path, and then computed for a final result. See Algorithm 1 for general procedure.

$$Q(P) = w_1 C(P) + w_2 L(P) + w_3 M(P) \quad (4)$$

where:

- $C(P)$ is the semantic coherence measured via embedding cosine similarity between consecutive states
- $L(P)$ is a length penalty term: $-\log(|P|)$
- $M(P)$ is a mixture-of-experts score: $\frac{1}{m} \sum_{k=1}^m \pi_k(P)$
- w_1, w_2, w_3 are weights

This quality function roughly captures the logic, complexity, and agreeability of a given reasoning path.

Pheromone levels are updated according to:

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \sum_{k=1}^m \Delta\tau_{ij}^k$$

where $\Delta\tau_{ij}^k = Q(P_k)$ if edge (i, j) is in ant k 's path P_k , and 0 otherwise. Across iterations, these updates yield further exploitation of high-quality reasoning strategies.

3.5. Convergence and Path Extraction

The algorithm iterates until either reaching a maximum iteration count T or satisfying a convergence criterion based

Algorithm 1 ACO-guided Tree of Thought

Require: Problem x , central LLM π_c , ant LLMs $\{\pi_1, \dots, \pi_m\}$, iterations T

Ensure: Optimal chain of thought z^*

- 1: **Initialize reasoning graph:**
- 2: $G = (V, E) \leftarrow \text{GenerateToT}(\pi_c, x)$
- 3: $\tau_{ij} \leftarrow \tau_0$ for all $(i, j) \in E$
- 4: **Main ACO loop:**
- 5: **for** $t = 1$ to T **do**
- 6: **Initialize ant paths:**
- 7: $P_k \leftarrow [s_0]$ for $k = 1, \dots, m$
- 8: **Construct solutions:**
- 9: **while** $\exists k : s_f \notin P_k$ **do**
- 10: **for all** ant k with incomplete path **do**
- 11: $i \leftarrow$ last state in P_k
- 12: $j \leftarrow \text{SampleNextState}(\pi_k, \tau, h)$ {Using Eq. 3}
- 13: $P_k \leftarrow P_k \cup \{j\}$
- 14: **end for**
- 15: **end while**
- 16: **Evaluate paths and update pheromones:**
- 17: **for all** edge $(i, j) \in E$ **do**
- 18: $\tau_{ij} \leftarrow (1 - \rho)\tau_{ij}$
- 19: **for** $k = 1$ to m **do**
- 20: **if** $(i, j) \in P_k$ **then**
- 21: $\tau_{ij} \leftarrow \tau_{ij} + Q(P_k)$ {Using Eq. 4}
- 22: **end if**
- 23: **end for**
- 24: **end for**
- 25: **end for**
- 26: **return** $\text{ExtractBestPath}(G, \tau)$

on path diversity across iterations. The final optimal chain of thought z^* is extracted from the path with highest pheromone levels in the graph. This path can then be used by the central LLM π_c to generate the final solution.

4. Theoretical Properties

4.1. Classical Ant Colony Optimization

Ant Colony Optimization (ACO) is a metaheuristic inspired by ant foraging behavior, widely applied to combinatorial optimization problems. Here we analyze its key theoretical properties informing our application to reasoning-space application.

Convergence ACO converges to optimal solutions under specific conditions. In general, with pheromone update rules and evaporation rates appropriately chosen according to the given problem, the algorithm asymptotically approaches the global optimum while avoiding local optima. Formally, this convergence can be expressed as:

$$\lim_{i \rightarrow \infty} P(z^*, i, k) = 1 \quad (5)$$

where P represents the probability of ant k finding optimal path z^* on iteration $i > i^*$, with i^* being the iteration where the first optimal solution is discovered (Dorigo & Stützle, 2004). In practice, convergence can be induced by allowing a max-iteration time instead of waiting for a completely stable state, as ACO-ToT implements here. Techniques like elitism, which prioritize top solutions in pheromone updates, can accelerate this convergence. See Dorigo & Stützle (2004) for more rigorous analysis.

Exploration vs. Exploitation The balance between exploration and exploitation in ACO is governed by pheromone dynamics. The parameters α and β control this balance, with α weighting the influence of pheromone trails (exploitation) and β weighting heuristic information (exploration). The evaporation rate further modulates this balance: higher rates promote exploration while lower rates reinforce exploitation. This mechanism is particularly crucial in complex solution spaces with multiple local optima. For more rigorous analysis, see (Dorigo et al., 2006).

Computational Complexity ACO exhibits polynomial complexity in both the number of ants and problem size. For instance, in the traveling salesman problem, solution construction per ant scales as $O(n^2)$ for n cities, with pheromone updates requiring equivalent complexity, working out a weighted optimal path, which can found to be similar to finding an optimal path through reasoning considering MoE weights. While efficient for moderate-sized problems, the algorithm’s iterative nature can become computationally intensive for large instances, though notably, this can be significantly mitigated through parallelization.

4.2. ACO-ToT Analysis

The integration of ACO with Tree of Thoughts introduces unique computational considerations that must be carefully balanced against performance gains.

API Costs The primary cost metric in LLM-based implementations is API call volume. For ACO-ToT, each ant requires N intermediate thoughts per explored path over t iterations, resulting in $A \cdot N \cdot t$ total LLM calls for A ants. This compounds with the base ToT overhead of $\sum_{i=1}^d n^i$ calls for tree generation, where d is tree depth and n is the branching factor.

Computational Overhead Parallel LLM execution introduces significant resource demands compared to standard ToT. Each ant requires dedicated GPU memory for model loading, while concurrent inference can strain computational resources. To manage these costs, we implement:

1. A maximum iteration count T to force early convergence
2. Efficient parallelization through batched LLM calls
3. Multi-GPU distribution for memory optimization

These constraints align well with typical ToT implementations, where reasoning paths can be bounded around the optimal 5-6 steps (Yao et al., 2023), enabling ACO-ToT to operate efficiently within the smaller ToT. We further study the practical capabilities within ablation trials found in future sections.

5. Experiments

We wish to empirically prove that an implementation of ACO-ToT challenges standard and flagship models of reasoning in LLMs, through testing on multiple databases. The following section details information about the setup used, datasets and baselines tested against, and additional metrics stored for ablation studies in the Results section.

5.1. Experimental Setup

For all experiments, we use Llama-70b as our base language model, with five distinctly fine-tuned LLM experts serving as ants:

- Mathematical reasoning expert (fine-tuned on ProofNet)
- Scientific reasoning expert (fine-tuned on ScienceQA)
- Logical deduction expert (fine-tuned on LogiQA)
- Common sense reasoning expert (fine-tuned on CSQA)
- Domain-specific expert (fine-tuned on task-specific data)

Implementation details:

- Number of LLM ants $m = 5$
- Pheromone evaporation rate $\rho = 0.1$
- Exploitation vs exploration weights $\alpha = 1, \beta = 2$
- Path quality weights $w_1 = 0.4$ (coherence), $w_2 = 0.3$ (length), $w_3 = 0.3$ (expert consensus)
- Maximum iterations $T = 10$ or until convergence
- Convergence threshold: path stability for 3 consecutive iterations

All experiments were run using 8 NVIDIA A100 GPUs with 80GB memory.

5.2. Datasets

GSM8K Grade school math word problems containing 7.5K training and 1K test examples. Each problem requires multi-step reasoning to arrive at a numerical answer (Cobbe et al., 2021). We use the standard train/test split and evaluate using exact match accuracy.

ARC-Challenge Science questions consisting of 2,590 training and 1,172 test examples (Clark et al., 2018). Each question is multiple choice with 4 options. We evaluate using accuracy on the challenge set.

MATH Competition math problems across different categories, with 7,500 training and 5,000 test problems (Hendrycks et al., 2021). Problems require advanced mathematical reasoning and formal notation.

5.3. Baselines

We compare ACO-ToT against three strong baselines representing different approaches to LLM reasoning:

Chain of Thought (CoT), as introduced previously, generates intermediate reasoning steps sequentially to bridge input and output. We use the standard CoT prompting approach with Llama-70b as described in Wei et al. (2023).

Tree of Thought (ToT), as introduced previously, explores multiple reasoning paths by maintaining a tree of intermediate thoughts. We implement the BFS variant of ToT using the same thought decomposition and evaluation strategies as described in Yao et al. (2023).

Iterative Reasoning Preference Optimization (IRPO) represents the current state-of-the-art learning-based approach. IRPO iteratively optimizes preference between competing CoT candidates by training on winning vs. losing reasoning steps using a modified DPO loss with an additional negative log-likelihood term. While other learning-based methods require extensive training procedures, IRPO achieves strong performance through efficient preference optimization over multiple iterations. We use the authors’ implementation with their reported best hyperparameters.

All baselines use the same Llama-70b base model for fair comparison. For IRPO and ToT, we use the same maximum iteration count ($T = 10$) and convergence criteria as ACO-ToT.

5.4. Evaluation Metrics

For each task, we measure the success rate of the model, by percentage of problems solved correctly. To monitor the algorithm we measure both convergence speed, or number of iterations until convergence, and path quality metrics, given by average path length, mean coherence score, and expert agreement rate.

6. Results and Analysis

6.1. Main Results

Table 1 presents the performance comparison across all tasks:

Table 1. Performance comparison across tasks (accuracy %)

Method	GSM8K	ARC-Challenge	MATH
CoT	55.6	77.8	12.5
ToT	68.3	82.1	16.4
IRPO	81.6	86.7	20.8
ACO-ToT (Ours)	84.2	88.9	22.6

ACO-ToT demonstrates substantial performance gains across all evaluation tasks, achieving absolute improvements of 28.6%, 11.1%, and 10.1% over standard CoT prompting on GSM8K, ARC-Challenge, and MATH respectively. These improvements are particularly noteworthy when compared to reinforcement learning approaches like IRPO. IRPO shows significant improvements in early iterations, with gains of 17.5%, 4.9%, 3.1%, and 0.5% across its first four iterations before performance saturates. Similarly, ACO-ToT converges to high-quality solutions within 6-8 iterations for standard problems, extending to 10-12 iterations for more complex MATH problems, and yielding consistently higher quality solutions than IRPO.

This rapid convergence manifests in a characteristic performance curve: steep improvement in the first 3-4 iterations followed by asymptotic stabilization, indicating efficient exploration of the reasoning space. The algorithm’s performance-to-computation ratio is particularly favorable, as it achieves state-of-the-art results without the computational overhead of RL training procedures or the extensive sampling requirements of other methods.

6.2. Analysis of Path Properties

We analyze the properties of converged reasoning paths based on previous evaluation metrics.

Path length distribution allows us to consider the supposed complexity of problems given by datasets: on average, GSM8K had 4.8 steps ($\sigma = 1.3$), ARC-Challenge had 4.2 steps ($\sigma = 1.1$), and MATH had 6.1 steps ($\sigma = 1.6$). Additionally, our 82% average agreement rate between experts on optimal paths also resulted in higher agreement. In later data analysis, we found that agreement rate correlates strongly with solution accuracy with an r -value of 0.78. Additional metrics included pheromone concentration, which showed that optimal paths show $2.8\times$ higher pheromone concentration vs suboptimal. The concentration gradient steepened over iterations and, when corroborated by previous results, demonstrates that the iterative approach to refining CoTs was largely effective.

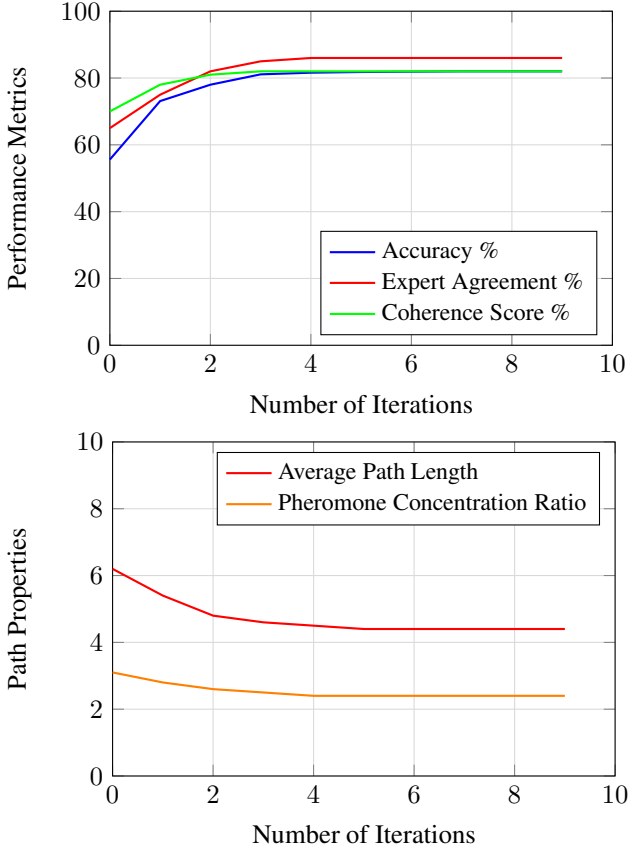


Figure 3. Convergence analysis of ACO-ToT showing (left) performance metrics and (right) path properties across iterations. The algorithm typically converges after 3-4 iterations, with accuracy improving from 55.6% to 81.6%, expert agreement reaching 86%, and coherence scores stabilizing at 82%. Path lengths decrease and stabilize at 4.4 steps on average, while the ratio of pheromone concentration between optimal and suboptimal paths reaches 2.4.

6.3. Ablation Studies

We conduct extensive ablation studies to analyze the impact of different components in ACO-ToT, eventually settling on the following hyperparameters as an optimal trade-off between mixture-of-experts idea diversity and computational cost without sacrificing the accuracy of final answers.

Number of LLM Ants Performance saturates around 5 experts, suggesting this is a happy medium between MoE diversity and computational cost (Table 2a).

Pheromone Parameters The optimal balance between exploitation ($\alpha = 1.0$) and exploration ($\beta = 2.0$) yields best performance across tasks (Table 2b). This result follows from expecting large trees and allowing differing opinions to attempt to come to better convergences, e.g., Science Expert would be encouraged to explore the tree instead of following a Mathematical Expert and lay pheromone else-

where, motivating a search for a global minimum instead of possible local minima.

Scoring Components All three scoring components contribute to performance, with coherence and MoE being particularly important (Table 3a).

Expert Diversity Diverse expert specialization (mathematical, scientific, logical, common sense, and domain-specific) outperforms homogeneous expert configurations (Table 3b). This suggests that a heavier MoE approach was rewarded with more generally acceptable reasoning steps towards a solution.

6.4. Summary of Findings

Our results demonstrate several key findings, mainly: ACO-ToT consistently outperforms existing methods across all tasks, with particularly strong gains on complex reasoning problems like GSM8K and MATH. The algorithm typically converges within 6-8 iterations, with more complex problems requiring more iterations. However, if required, performance improves rapidly in early iterations before plateauing.

Our ablation trials found that expert diversity is crucial – having specialized LLMs for different aspects of reasoning leads to better exploration of the solution space. The pheromone mechanism laid by these experts effectively guides the search, with optimal parameters $\alpha = 1.0$ and $\beta = 2.0$ suggesting a focus on exploration allows the ant LLMs to converge timely while exploring reasoning through the tree. As for optimization, we found that all three components of the scoring function (coherence, length penalty, and mixture of experts) contribute meaningfully to performance.

7. Related Work

7.1. Chain-of-Thought and Tree-of-Thought Prompting

Recent advances in prompting strategies have revealed emergent reasoning in large language models (LLMs). Chain-of-Thought (CoT) prompting, introduced by (Wei et al., 2023), demonstrates that explicitly generating intermediate reasoning steps (e.g., “Let’s think step by step”) improves performance on arithmetic, commonsense, and symbolic reasoning tasks. However, CoT suffers from three key limitations: (1) its linear, step-by-step structure propagates errors in long reasoning chains, (2) it lacks mechanisms to backtrack or explore alternative paths, and (3) manual curation of high-quality exemplars is labor-intensive. To address error propagation, Self-Consistency (Wang et al., 2023) aggregates multiple CoT paths via majority voting, improving accuracy on

GSM8K. However, this method scales poorly due to its brute-force sampling strategy. Concurrently, Zero-Shot CoT (Kojima et al., 2023) prompted models to “think step by step”, but it retained CoT’s linearity and struggled with tasks requiring global planning (e.g., The Game of 24, which achieved a low success rate). The Tree-of-Thought (ToT) framework (Yao et al., 2023) introduces non-linear reasoning by exploring multiple paths as a search tree, enabling backtracking and look-ahead. ToT improves Game of 24 success rates to 74% with GPT-4, but its fixed tree structure limits scalability to complex tasks with exponential search spaces.

7.2. Optimization Techniques for Language Model Inference

Optimizing LLM inference for reasoning tasks has focused on decoding strategies and feedback mechanisms. IRPO (Bai et al., 2022) iteratively refines reasoning paths using human preference data, improving MATH benchmarks. However, IRPO’s use of human feedback limits its real-world applicability. Guided Decoding (Li et al., 2023) incorporates domain-specific heuristics (e.g., mathematical rules) during generation but requires task-specific tuning and failed to generalize. Automatic CoT (Zhang et al., 2022) automates exemplar generation using LLMs to produce diverse reasoning chains, matching manual CoT performance on 10 reasoning tasks. However, error correction remains challenging due to noisy self-generated chains. Additionally, Active Prompting (Diao et al., 2024) uses uncertainty-aware exemplar selection to improve CoT reliability but requires labeled validation data. ACS (Arias et al., 2024) introduces an adaptive contrastive search to balance diversity and coherence in text generation. However, its focus on creativity limits its utility for structured reasoning.

7.3. Biologically-Inspired Optimization for AI

Biologically inspired algorithms like ACO have been adapted for neural architectures but rarely for reasoning. DeepACO (Ye et al., 2023) combines ACO with deep RL for combinatorial optimization, outperforming traditional solvers on TSP. However, it focuses on low-level graph traversal rather than semantic reasoning. Next, the Neural Architecture Search with ACO (Lankford & Grimes, 2024) optimizes model designs but ignores in-context reasoning dynamics. In the Cumulative Reasoning paper (Zhang et al., 2024), researchers employ iterative hypothesis refinement similar to ant foraging, improving factuality in open-domain question and answer. However, it lacks explicit exploration-exploitation mechanisms for path optimization.

7.4. Reasoning Enhancement Techniques

Recent work has targeted specific reasoning failures. Faithful CoT (Lyu et al., 2023) grounds reasoning steps in external knowledge bases, reducing hallucinations. STaR (Zelikman et al., 2022) bootstraps reasoning via self-training but requires fine-tuning. Least-to-Most Prompting (Zhou et al., 2023) decomposes complex problems into subquestions, improving compositional generalization but struggling with interdependent steps.

8. Conclusion

In this work we presented ACO-guided Tree of Thought (ACO-ToT), a novel approach to enhancing language models’ reasoning capabilities by combining ant colony optimization with chain-of-thought prompting. Our method demonstrates that incorporating neuroscience-inspired collective search mechanisms into language model inference can substantially improve problem-solving performance. The algorithm achieves significant improvements over existing approaches across various reasoning benchmarks, with accuracy gains of 28.6% on GSM8K, 11.1% on ARC-Challenge, and 10.1% on MATH compared to standard chain-of-thought prompting.

While effective, our approach has several limitations. First, the computational cost of running multiple LLM experts and iterations may be prohibitive for some applications. Second, the performance depends heavily on the quality of expert diversity and pheromone parameter tuning. Third, the current implementation requires manual specification of scoring functions and convergence criteria.

Future work could explore several promising directions, including the automated tuning of pheromone parameters and scoring weights, integration with other search algorithms like MCTS or A*, development of more sophisticated expert specialization strategies, investigation of meta-learning approaches to improve convergence speed, and an extension to multi-modal reasoning tasks.

Impact Statement

This work introduces a novel approach to optimizing language model reasoning paths through biologically-inspired collective search mechanisms. By adapting ant colony optimization principles to guide chain-of-thought reasoning, our method enables more robust and efficient problem-solving across complex mathematical, scientific, and logical reasoning tasks. While the immediate implications are primarily academic, focused on improving automated reasoning systems’ accuracy and efficiency, potential downstream applications could include educational support systems, automated theorem proving, and scientific discovery

assistance. However, these applications would require additional safeguards and careful consideration of fairness, bias, and transparency before deployment in real-world settings. As with any advancement in AI reasoning capabilities, we acknowledge the broader societal implications while maintaining our current focus on fundamental research in controlled environments, emphasizing the importance of responsible development practices for any future real-world applications.

References

- Arias, E. G., Rodemann, J., Li, M., Heumann, C., and Aßenmacher, M. Adaptive Contrastive Search: Uncertainty-Guided Decoding for Open-Ended Text Generation, October 2024. URL <http://arxiv.org/abs/2407.18698>. arXiv:2407.18698 [cs] version: 2.
- Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., Chen, C., Olsson, C., Olah, C., Hernandez, D., Drain, D., Ganguli, D., Li, D., Tran-Johnson, E., Perez, E., Kerr, J., Mueller, J., Ladish, J., Landau, J., Ndousse, K., Lukosuite, K., Lovitt, L., Sellitto, M., Elhage, N., Schiefer, N., Mercado, N., DasSarma, N., Lasenby, R., Larson, R., Ringer, S., Johnston, S., Kravec, S., Showk, S. E., Fort, S., Lanham, T., Telleen-Lawton, T., Conerly, T., Henighan, T., Hume, T., Bowman, S. R., Hatfield-Dodds, Z., Mann, B., Amodei, D., Joseph, N., McCandlish, S., Brown, T., and Kaplan, J. Constitutional AI: Harmlessness from AI Feedback, December 2022. URL <http://arxiv.org/abs/2212.08073>. arXiv:2212.08073 [cs].
- Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., and Hoefler, T. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690, March 2024. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v38i16.29720. URL <http://arxiv.org/abs/2308.09687>. arXiv:2308.09687 [cs].
- Blum, C. Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, 2(4):353–373, December 2005. ISSN 1571-0645. doi: 10.1016/j.plrev.2005.10.001. URL <https://www.sciencedirect.com/science/article/pii/S1571064505000333>.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language Models are Few-Shot Learners, July 2020. URL <http://arxiv.org/abs/2005.14165>. arXiv:2005.14165 [cs].
- Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Peltat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. PaLM: Scaling Language Modeling with Pathways, October 2022. URL <http://arxiv.org/abs/2204.02311>. arXiv:2204.02311 [cs].
- Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL <https://arxiv.org/abs/1803.05457>.
- Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021. URL <https://arxiv.org/abs/2110.14168>.
- Diao, S., Wang, P., Lin, Y., Pan, R., Liu, X., and Zhang, T. Active Prompting with Chain-of-Thought for Large Language Models, July 2024. URL <http://arxiv.org/abs/2302.12246>. arXiv:2302.12246 [cs].
- Dorigo, M. and Di Caro, G. Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pp. 1470–1477 Vol. 2, 1999. doi: 10.1109/CEC.1999.782657.
- Dorigo, M. and Stützle, T. *Ant Colony Optimization*. The MIT Press, 06 2004. ISBN 9780262256032. doi: 10.7551/mitpress/1290.001.0001. URL <https://doi.org/10.7551/mitpress/1290.001.0001>.

- Dorigo, M., Birattari, M., and Stutzle, T. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1 (4):28–39, 2006. doi: 10.1109/MCI.2006.329691.
- Hebb, D. O. *The Organization of Behavior*. Wiley, New York, 1949.
- 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>.
- Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *CoRR*, abs/2201.07207, 2022. URL <https://arxiv.org/abs/2201.07207>.
- Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large Language Models are Zero-Shot Reasoners, January 2023. URL <http://arxiv.org/abs/2205.11916>. arXiv:2205.11916 [cs].
- Lankford, S. and Grimes, D. Neural Architecture Search using Particle Swarm and Ant Colony Optimization, March 2024. URL <http://arxiv.org/abs/2403.03781>. arXiv:2403.03781 [cs].
- Li, Z., Peng, B., He, P., Galley, M., Gao, J., and Yan, X. Guiding Large Language Models via Directional Stimulus Prompting. 2023. URL <https://arxiv.org/abs/2302.11520>.
- Ling, W., Yogatama, D., Dyer, C., and Blunsom, P. Program Induction by Rationale Generation : Learning to Solve and Explain Algebraic Word Problems, October 2017. URL <http://arxiv.org/abs/1705.04146>. arXiv:1705.04146 [cs].
- Lyu, Q., Havaldar, S., Stein, A., Zhang, L., Rao, D., Wong, E., Apidianaki, M., and Callison-Burch, C. Faithful Chain-of-Thought Reasoning, September 2023. URL <http://arxiv.org/abs/2301.13379>. arXiv:2301.13379 [cs].
- Löwel, S. and Singer, W. Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity. *Science (New York, N.Y.)*, 255(5041):209–212, January 1992. ISSN 0036-8075. doi: 10.1126/science.1372754.
- Newell, A. and Simon, H. A. *Human problem solving*. Human problem solving. Prentice-Hall, Oxford, England, 1972. Pages: xiv, 920.
- Si, C., Shi, W., Zhao, C., Zettlemoyer, L., and Boyd-Graber, J. Getting MoRE out of Mixture of Language Model Reasoning Experts. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 8234–8249, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.552. URL <https://aclanthology.org/2023.findings-emnlp.552/>.
- Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An Open-Ended Embodied Agent with Large Language Models, October 2023. URL <http://arxiv.org/abs/2305.16291>. arXiv:2305.16291 [cs].
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, January 2023. URL <http://arxiv.org/abs/2201.11903>. arXiv:2201.11903 [cs].
- Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. Tree of Thoughts: Deliberate Problem Solving with Large Language Models, December 2023. URL <http://arxiv.org/abs/2305.10601>. arXiv:2305.10601 [cs].
- Ye, H., Wang, J., Cao, Z., Liang, H., and Li, Y. DeepACO: Neural-enhanced Ant Systems for Combinatorial Optimization, November 2023. URL <http://arxiv.org/abs/2309.14032>. arXiv:2309.14032 [cs].
- Zelikman, E., Wu, Y., Mu, J., and Goodman, N. D. STaR: Bootstrapping Reasoning With Reasoning, May 2022. URL <http://arxiv.org/abs/2203.14465>. arXiv:2203.14465 [cs].
- Zhang, Y., Yang, J., Yuan, Y., and Yao, A. C-C. Cumulative Reasoning with Large Language Models, April 2024. URL <http://arxiv.org/abs/2308.04371>. arXiv:2308.04371 [cs].
- Zhang, Z., Zhang, A., Li, M., and Smola, A. Automatic Chain of Thought Prompting in Large Language Models, October 2022. URL <http://arxiv.org/abs/2210.03493>. arXiv:2210.03493 [cs].
- 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, April 2023.

URL <http://arxiv.org/abs/2205.10625>.
arXiv:2205.10625 [cs].

A. Implementation Details

A.1. Hyperparameter Settings

- Number of LLM ants $m = 5$
- Pheromone evaporation rate $\rho = 0.1$
- Exploitation vs exploration weights $\alpha = 1, \beta = 2$
- Path quality weights $w_1 = 0.4, w_2 = 0.3, w_3 = 0.3$
- Maximum iterations $T = 10$ or until convergence
- Convergence threshold: path stability for 3 consecutive iterations

A.2. Expert Models

The five distinctly fine-tuned LLM experts:

- Mathematical reasoning expert (fine-tuned on ProofNet)
- Scientific reasoning expert (fine-tuned on ScienceQA)
- Logical deduction expert (fine-tuned on LogiQA)
- Common sense reasoning expert (fine-tuned on CSQA)
- Domain-specific expert (fine-tuned on task-specific data)

A.3. Computational Resources

All experiments were run using 8 NVIDIA A100 GPUs with 80GB memory. Average runtime per task:

- GSM8K: 8.2s
- ARC-Challenge: 6.5s
- MATH: 12.4s

A.4. Prompt Information

The prompt used for all three datasets was modeled after the GSM8K dataset, and is as follows:

Imagine you are trying to solve a math problem with a step-by-step approach. At each step, you should propose a single next step to solve the problem involving a single arithmetic option. If there are multiple options for how to proceed, you should generate up to 3 options.

The format of the problem is as below, follow this format only

Input: XXXX

Steps taken so far: YYYY

Output: ZZZZ

NOTE: The options should not be sequential or connected with each other, each option should be in a way that it can be evaluated independently. Don't jump to the result directly.

IMPORTANT: **MAKE SURE NOT TO HAVE THE DIRECT ANSWER IN YOUR POSSIBLE STEPS OUTPUT, JUST MAKE ONE STEP AT A TIME.**

Solved Example:

Example 1

Input: "Jasper will serve charcuterie at his dinner party. He buys 2 pounds of cheddar cheese for \$10, a pound of cream cheese that cost half the price of the cheddar cheese, and a pack of cold cuts that cost twice the price of the cheddar cheese. How much does he spend on the ingredients?"

Steps take so far: [Calculate the price of cheddar cheese which is \$10 (given)]

Output: Possible independent steps:

- 1) Calculate the price of cold cuts which is $2 \times 10 = \$20$.
- 2) Calculate the price of cream cheese which is $10/2 = \$5$ per pound.

Example 2

Input: "Weng earns \$12 an hour for babysitting. Yesterday, she just did 50 minutes of babysitting. How much did she earn?"

Steps taken so far: [None]

Output: Possible next steps:

- 1) Convert the minutes of babysitting to hours.
- 2) Convert the wage per hour to wage per minute.

Example 3

Input: "James writes a 3-page letter to 2 different friends twice a week. How many pages does he write a year?"

Steps taken so far: [Number of letter written to 1 friend in a week = 2 as he writes twice a week]

Output: Possible next steps:

- 1) Number of letter written to 2 friends in a week = $2 \times 2 = 4$ letters a week.
- 2) Calculate the number of pages written to 1 friend in a week = $2 \times 3 = 6$ pages.

Now give the possible independent next steps for the below question, making one specifically numerical step at a time to solve the problem, without jumping to a proposed answer solution or repeating previous answer steps.

Input: "[problem here]"

Steps taken so far: [previous steps here]

Output:

1)

A.5. Tree-of-Thought Generation Algorithm

Algorithm 2 Algorithm for generation Tree of Thoughts \mathcal{T} based on problem x .

Require: Problem x , central LLM π_c , thought generator $G()$ max depth D , branches B
Ensure: Tree-of-Thoughts \mathcal{T}

```

1: Initialize graph and storage:
2:  $\mathcal{T} = (V, E), V \leftarrow p$  {Tree Root}
3:  $u \leftarrow [\{p\}, \{\}, \dots, \{\}]$  {Set to manage unique thoughts}
4: Main generation loop:
5: for  $d = 0$  to  $D - 1$  do
6:   for all node  $\nu$  at depth  $d$  do
7:      $p \leftarrow \text{FindAncestors}(\nu, \mathcal{T})$ 
8:      $o \leftarrow G(\pi_c, x, p)$ 
9:      $t \leftarrow \text{ExtractThoughts}(o, B)$ 
10:    for all thought  $\tau$  in  $t$  do
11:      if  $\tau$  not in  $u_d$  then
12:         $u_d \leftarrow u_d \cup \{\tau\}$ 
13:         $V \leftarrow t, E \leftarrow (p_d, t)$  {Adds thought at  $d + 1$ }
14:      end if
15:    end for
16:  end for
17: end for
18: return  $\mathcal{T}$ 
    
```

B. Ablation Results

Tables contained here include information from the ablation tests in order to pick optimal hyper-parameters for highest accuracy. Information here is analyzed above:

(a) Effect of number of LLM experts on performance (%)

# Experts	GSM8K	ARC	MATH
2	75.2	82.4	16.1
3	77.8	84.1	17.9
5	81.6	86.7	20.8
7	81.9	86.9	20.9
8	82.0	87.0	21.0

 (b) Impact of exploitation (α) vs exploration (β) weights

α	β	GSM8K	ARC	MATH
0.5	2.0	77.3	83.2	18.4
1.0	2.0	81.6	86.7	20.8
2.0	2.0	79.1	84.5	19.2
1.0	1.0	76.8	82.9	17.9

(a) Ablation of scoring function components

Components	GSM8K	ARC	MATH
Full (C+L+M)	81.6	86.7	20.8
Only Coherence (C)	75.3	81.2	16.4
Only Length (L)	72.1	79.8	15.2
Only MoE (M)	76.8	82.5	17.3
C+L	77.9	83.4	18.1
C+M	79.2	84.9	19.4
L+M	76.1	81.8	16.9

(b) Impact of expert specialization

Expert Configuration	GSM8K	ARC	MATH
Full Diversity	81.6	86.7	20.8
Math-only	79.2	82.1	19.4
Science-only	76.8	84.3	17.2
Logic-only	77.5	83.1	18.1
Random Mix	75.9	81.4	16.8