ARIES: Autonomous Reasoning with Large Language Models on Interactive Thought Graph Environments

Anonymous ACL Submission

Abstract

Recent research has shown that the performance of Large Language Models (LLMs) on reasoning tasks can be enhanced by scaling test-time compute. One promising approach, particularly with decomposable problems, involves arranging intermediate solutions as a graph on which transformations are performed to explore the solution space. However, prior works rely on pre-determined, task-specific transformation schedules defined by a set of searched hyperparameters. In this work, we view thought graph transformations as actions in a Markov decision process, and investigate whether LLMs can act as a policy agents on this graph-based environment. We introduce ARIES, a multi-agent architecture for reasoning with LLMs in which LLM policy agents maintain visibility of the thought graph states and dynamically adapt the problem-solving strategy. We observe that using off-the-shelf LLMs as policy agents with ARIES can yield up to 29% higher accuracy on HumanEval relative to static transformation schedules despite bearing no search cost.

1 Introduction

Recent work showed that under a fixed compute budget for training and inference, LLM performance on reasoning tasks can be enhanced by allocating a higher proportion of compute to inference rather than training (Snell et al., 2024). This has been achieved using techniques such as Chain-of-Thoughts or Tree-of-Thought prompting (Wei et al., 2023; Yao et al., 2023a). Another promising approach involves arranging intermediate solutions (or "thoughts") in a graph, i.e. topological reasoning (Besta et al., 2024b). This proves particularly beneficial when problems can be decomposed into subproblems to be solved independently then aggregated through a sequence of graph transformations (Besta et al., 2024a), and intermediate solutions can be reliably scored using a Process Reward Model (Snell et al., 2024). Despite the benefits of topological reasoning, prior works rely on pre-determined traversal strategies parametrized by a discrete set of hyperparameters (Yao et al., 2023a; Besta et al., 2024a). This approach lacks generality, as these parameters must be tuned manually or through extensive Bayesian search to achieve high query efficiency, due to the varying characteristics of each task. To overcome this limitation, we propose viewing thought graphs as an interactive environment where graph transformations are seen as actions in a Markov Decision Process (MDP). As such, we turn to investigating action policies to effectively explore the solution space and yield a solution while learning from external feedback.

Motivated by recent improvements in LLMs, we aim to investigate whether their planning and reasoning abilities extend to selecting and executing a sequence of transformations (such as decomposition, solving, evaluation, refinement and aggregation) within the aforementioned thought graph environments. We implement ARIES, a multi-agent framework for solving reasoning problems formulated as thought graphs.

Figure 1 provides a summary of our approach - in each iteration, the policy agent monitors the thought graph state and samples from the action space to choose a graph transformation. The reasoning agent then performs these transformations and updates the thought graph state. Through a series of carefully controlled experiments against a number of benchmarks, we show that LLM-guided thought graph exploration can lead to up to 29% higher accuracy, as well as obviating search cost.

2 Topological Reasoning with Large Language Models

Given a reasoning problem defined as an ordered tuple of tokens $p = (t_1, \ldots, t_m)$, we define a thought $\tau = (t_1, \ldots, t_j)$ as a sequence of tokens sampled autoregressively from an LLM parametrized by θ,

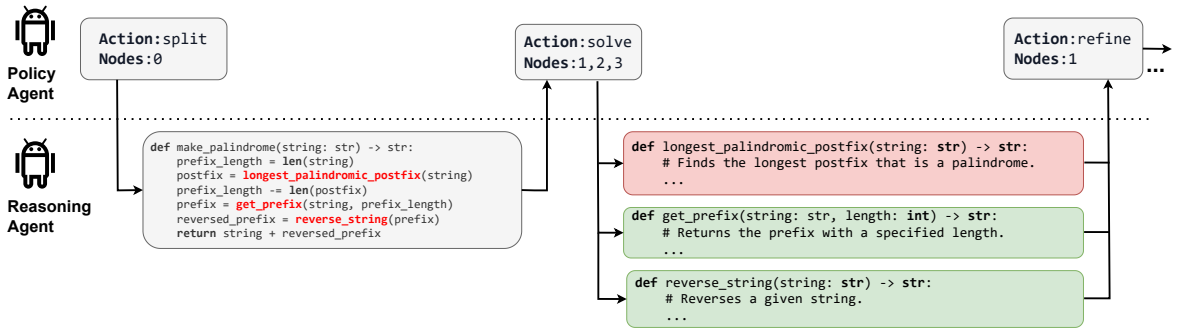

Figure 1: ARIES workflow in answering the HumanEval prompt: "Find the shortest palindrome that begins with a supplied string". First, the policy agent selects the split action, guiding the reasoning agent to decompose the problem by generating a skeleton implementation with yet-to-implement subfunctions. Since one of the solutions doesn't pass its testcases, the reasoning agent is instructed to refine it based on execution feedback.

i.e. $t_i \sim P(t_i \mid t_1, \ldots, t_{i-1}; \theta)$. This consists of a language representation of an intermediate step towards the solution. A thought sequence can hence be represented as an ordered tuple of thoughts $S = (\tau^1, \tau^2, \ldots, \tau^k)$ of length k, such that the final thought τ^k represents a candidate solution to the problem p. A thought graph G_τ can be represented as $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of thought nodes and $\mathcal{E}$ is a set of edges connecting them. Additionally, each thought τ^{ij} (j-th thought at depth i) has a value $\lambda(\tau^{ij})$ such that nodes with higher values yield valid solutions to the problem with higher probability. Thought graph exploration can be regarded as a sequence of m graph transformations as follows, where each $\phi_i : G_\tau^i \rightarrow G_\tau^{i+1}$ modifies the set of nodes and edges.

$$G_\tau^* = \phi_m(\ldots(\phi_1(\phi_0(G_\tau^0)))) \quad (1)$$

We consider a finite set of transformations; ϕ_{dec} decomposes a reasoning problem into subproblems to be solved individually, creating new nodes in the thought graph. ϕ_{sol} generates a candidate solution to a subproblem. ϕ_{ref} considers an incorrect subproblem solution, utilizing further LLM queries to refine it. ϕ_{red} removes nodes in the graph according to their values. Finally, ϕ_{agg} performs node merging to aggregate subproblem solutions into a coherent solution to the original problem. Formal definitions of each transformation are shown in Table 2 in our Appendix.

A standard (static) divide-and-conquer strategy can be parametrized by the tuple $(R_{ed}, R_{ef}, S^m, A^m, R_{ef}^m)$, where S^m, A^m, R_{ref}^m represents the multiplicity (i.e. number of attempts) of ϕ_{sol}, ϕ_{agg} and ϕ_{ref}, respectively. First, the ϕ_{dec} transformation decomposes the starting problem into B subproblems, which are then solved individually using ϕ_{sol} then aggregated using ϕ_{agg}. $R_{ed}, R_{ef} \in \{0, 1\}$ indicate whether the ϕ_{red} and ϕ_{ref} transformations are applied after aggregation. If $R_{ed} = 1$, a single aggregation attempt is kept, while others are removed from the graph. If $R_{ef} = 1$, the remaining aggregation attempts are then refined with ϕ_{ref}, and the highest-scoring attempt is kept as the final solution. See Appendix B for further details on the divide-and-conquer schedule.

3 Thought Graph Exploration as a Markov Decision Process

Beyond static schedules, the transformation of a thought graph can be generalized as a Markov decision process $(\mathcal{S}, \mathcal{A}, \mathcal{P}_a)$ where the state $s_t \in \mathcal{S}$ represents an arrangement of nodes and edges in the thought graph, the action $a \in \mathcal{A}$ indicates a transformation and target node subset (i.e. $\mathcal{A} = \{(\mathcal{V}_s, \phi) \mid \mathcal{V}_s \subset \mathcal{V}, \phi \in \Omega\}$, where Ω is the set of transformations) and the transition probability $\mathcal{P}_a(s, s')$ represents the probability that an action a applied at state s yields the expected new state s'. The optimal transformation sequence Φ is then defined as the sequence of actions that maximize the conditional probability of reaching a solution state s^+, i.e. $\Phi = (\phi_0, \ldots, \phi_n)$ that solves the following optimization problem. We bound the number of queries ($|\Phi|$) by the constant ϵ, as in the limit $|\Phi| \rightarrow \infty$, $P(s^+|s^0, \Phi) \rightarrow 1$.

$$\arg\min_{\Phi} P(s^+ \mid s^0, \Phi) \quad \text{s.t.} \quad |\Phi| < \epsilon \quad (2)$$

In this work, we hypothesize that LLMs can approximate a solution to the stated optimization problem by acting as policy agents. We develop an

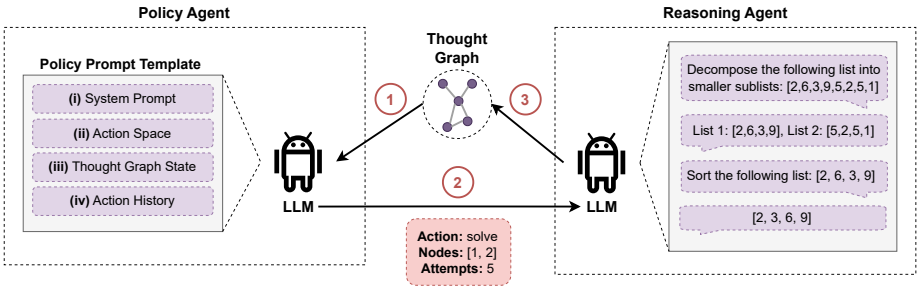

Figure 2: Multi-agent framework for reasoning over thought graphs. First, (1) the policy agent samples an action and subset of nodes given a prompt including (i-ii) general instructions and (iii-iv) an overview of the exploration state. The sample is then (2) passed to the reasoning agent, which finally (3) updates the thought graph state.

interactive framework consisting of a policy agent and a reasoning agent, as shown in Figure 2. In each iteration, (1) the policy agent selects an action from the action space, (i.e. the transformations in Table 2). The policy agent then (2) directs the reasoning agent to perform the selected action. Finally, (3) the reasoning agent updates the thought graph. The process is repeated until a solution is found or a maximum number of iterations is reached.

The policy agent is invoked using the prompt template shown in Figure 2. (i) The system prompt outlines the problem setting, input format and expected behaviour from the policy agent. (ii) A task-specific list of actions, describing the preconditions and effects of each transformation, provides a semantic understanding of the action space. (iii) The current state of the graph is provided in a textual format, enumerating all nodes and edges. Finally, (iv) the action history in the current trial is included, promoting continuity in the strategies outlined in previous steps.

In-Context Action Selection: Prior work has shown that reasoning abilities of LLMs are enhanced when prompted to output a verbose sequence of steps before the solution (Wei et al., 2023; Wang et al., 2023). This mechanism can be seen as enabling in-context task learning from some extracted innate world knowledge. Hence, our policy agent is instructed to generate a detailed analysis on the state of the thought graph and exploration history before sampling the action space.

Policy Agent Ensembles: Given the stochastic nature of LLM next-token prediction, we observe high variability in the chosen action over several invocations of a policy agent from the same thought graph state. To enhance robustness, we democratize action selection over an ensemble of agents, meaning a parametrizable number of LLM queries are performed concurrently at every iteration. The selected action is takes as the most frequent proposal among the ensemble. See Appendix G for ablation studies on the impact of policy agent ensemble size on reasoning performance.

4 Experiments

In this section, we outline the benchmarks, baselines and metrics used to evaluate ARIES against state-of-the-art methods for reasoning with LLMs. See also Appendix F for an evaluation of failure modes of our methodology.

Experimental Setup: We evaluate Llama-3.1-70B and Llama-3.1-405B as policy and reasoning agents, hosted with SGLang at a temperature of 1. We set the policy agent ensemble size to 5 in all experiments, as explained in Section G. Llama-3.1-70B was hosted with $8\times$ A6000 GPUs. Llama-3.1-405B was hosted using $16\times$ H100 GPUs distributed over 4 nodes. The total cost was approximately 3k GPU hours.

Benchmarks: We run our main evaluation on HumanEval (Chen et al., 2021). Additionally, we consider the set intersection problem at various levels of difficulty quantified by the size of the sets, leading to three benchmarks: set-intersection32/64/128. Despite its simplicity, this has been shown to be a challenging benchmark for LLMs with direct prompting (Besta et al., 2024a).

Baselines: We use static transformation schedules as the baseline, following (Besta et al., 2024a). For each individual task, we carefully tune the hyperparameters using Bayesian optimization. We compare against baselines with several search compute budgets by considering three variants: $\text{GoT}_{25\%}$, $\text{GoT}_{50\%}$ and $\text{GoT}_{100\%}$, where the percentage corresponds to the number of trials spent until the hyperparameter search converges. See Ap-

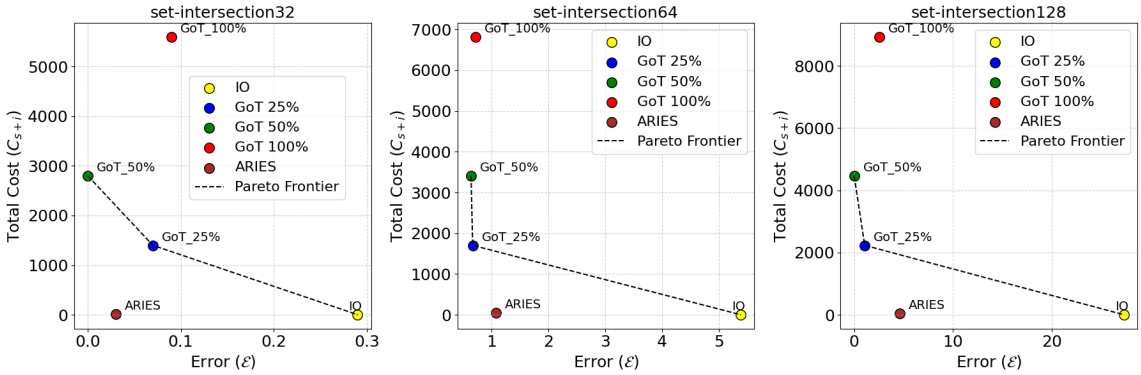

Figure 3: Pareto frontiers in total query cost (C_{s+i}) and task error ($\mathcal{E}$) for set intersection tasks at various difficulty levels. The total cost is the number of queries expended at search and inference time. Llama-3.1-405B was used for the reasoning and policy agents. Our results (ARIES) have pushed the Pareto frontiers forward in each task.

Table 1: Task accuracy (↑), search and inference costs (C_s and C_i, ↓) on the HumanEval task. C_s and C_i are measured in number of queries. We use LLaMA-3.1-405B and GPT-4o as the underlying reasoning and policy agents. Direct IO means direct prompting, GoT follows Besta et al. (2024a), and ReACT follows Yao et al. (2023b).

Method	**LLaMA-3.1-405B**			**GPT-4o**		
	Acc (%)	Search Cost (C_s)	Inference Cost (C_i)	Acc (%)	Search Cost (C_s)	Inference Cost (C_i)
Direct IO	77.4	0	**1**	84.1	0	**1**
$\text{GoT}_{25\%}$	66.3	1160	34.8	75.6	1160	19.8
$\text{GoT}_{50\%}$	67.5	2368	24.3	73.8	2368	18.5
$\text{GoT}_{100\%}$	60.1	4742	8.17	69.5	4742	6.8
ReACT	79.9	0	5.6	60.4	0	16.4
ARIES	**89.0**	**0**	31.6	**95.1**	**0**	35.4

pendix D for details on the full search methodology.

We also consider an IO (Input-Output) baseline, i.e. direct LLM prompting, and *smolagents* (Roucher et al., 2025), which is an implementation of the ReACT algorithm (Yao et al., 2023b). In the latter, the reasoning agent iteratively generates a candidate solution then reasons over execution feedback to refine subsequent candidates.

Reported metrics: For HumanEval, we report the task accuracy, while for list sorting and set intersection we report error function value $\mathcal{E}$. Details on the definition for the error function for each task can be found in Appendix C. Additionally, we report both the search cost C_s and inference cost C_i, measured as the number of LLM queries.

4.1 Results

HumanEval: Our key findings are shown in Table 1. It can be seen that by formulating this code generation task as a Markov decision process with an off-the-shelf LLM policy agent, we achieve up to 28.9% higher accuracy than the most query-efficient static schedule baseline with Llama-405b. Across both models, the inference cost is comparable to $\text{GoT}_{25\%}$, although the latter is obtained after 1160 LLM queries while ARIES avoids any search time requirement. Relative to the ReACT baseline, we achieve 9.1% higher accuracy with Llama-405b. However, this baseline does not generalize well across models, leading to a 9.1% accuracy degradation relative to $\text{GoT}_{100\%}$ with GPT-4o.

Set Intersection: In Figure 3, we plot a Pareto curve showing viable trade-off points in task error and query cost for the set intersection task. Our approach extends the existing Pareto frontier constructed by considering static schedule baselines and direct prompting. In the set-intersection32 task, we achieve a $2.3\times$ error reduction relative to GoT_{25} while also achieving $116\times$ lower overall cost.

5 Conclusion

We introduce ARIES, a multi-agent architecture for topological reasoning. By viewing thought graph transformations as actions in a Markov decision process, we show off-the-shelf LLMs can drive efficient action policies without task-specific tuning, leading to state-of-the-art accuracy at reduced inference costs.

6 Limitations

6.1 Assumptions and Robustness

The ARIES framework introduces a novel approach to reasoning with large language models (LLMs) through interactive thought graph environments. However, several strong assumptions underlie our methodology. Firstly, we assume that thought graph transformations can be effectively modeled as a Markov decision process (MDP) with well-defined state transitions. While this formulation enables structured reasoning, it may not fully capture the complexities of more ambiguous or highly interconnected problems. Additionally, our approach assumes that off-the-shelf LLMs can act as reliable policy agents without additional fine-tuning. This assumption holds for certain problem domains but may degrade in tasks requiring domain-specific knowledge or long-horizon planning.

Our empirical evaluation is constrained to specific reasoning tasks, including HumanEval and set intersection. While these benchmarks serve as valuable test cases for structured reasoning, they do not necessarily generalize to all problem types, particularly those with weakly defined intermediate states or multi-modal reasoning requirements. Furthermore, our evaluation primarily focuses on LLaMA-3.1 and GPT-4o models, and results may not be directly transferable to other architectures.

6.2 Potential Risks

The ARIES framework introduces both opportunities and challenges in autonomous reasoning. One primary risk is the potential for incorrect or biased reasoning paths due to the stochastic nature of LLM-generated decisions. Although our policy agent ensembles mitigate some of this variability, they do not fully eliminate erroneous transformations, particularly in deeper decomposition settings. The framework's reliance on existing LLMs also means that any biases present in the underlying models could propagate into the reasoning process, potentially leading to unfair or misleading outcomes.

Another concern is the environmental impact associated with inference-heavy approaches. While ARIES improves query efficiency relative to static transformation schedules, it still necessitates a significant number of LLM queries to achieve high accuracy. As LLMs scale, the energy consumption required for these inference tasks could become a sustainability concern, particularly in high-throughput applications.

6.3 Failure Modes

Our empirical findings regarding the failure modes of our methodology (Appendix F) highlight two major failure modes: (1) inadequate LLM parameter sizes and (2) increasing decomposition depth. Smaller models (e.g., LLaMA-3.1-70B) struggle to act as policy agents effectively, demonstrating subpar reasoning capabilities compared to larger counterparts. This suggests that autonomous policy-driven thought graph exploration may require models beyond a certain scale threshold to function reliably. Additionally, as the depth of problem decomposition increases, ARIES exhibits a decline in performance, primarily due to errors in aggregating intermediate solutions. This limitation indicates that current LLMs may have difficulties managing extended reasoning chains, which presents a barrier to scalability.

References

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

Maciej Besta, Florim Memedi, Zhenyu Zhang, Robert Gerstenberger, Guangyuan Piao, Nils Blach, Piotr Nyczyk, Marcin Copik, Grzegorz Kwaśniewski, Jürgen Müller, Lukas Gianinazzi, Ales Kubicek, Hubert Niewiadomski, Aidan O'Mahony, Onur Mutlu, and Torsten Hoefler. 2024b. Demystifying chains, trees, and graphs of thoughts. *Preprint*, arXiv:2401.14295.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. 2025. 'smolagents': a smol library to build great agentic systems. `https://github.com/huggingface/smolagents`.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Preprint*, arXiv:2206.07682.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Table 2: Thought graph transformations. Each transformation is defined as $\phi(G_\tau, m, S) = (V \cup V^+ \setminus V^-, E \cup E^+ \setminus E^-)$, where $G_\tau = (V, E)$ is a thought graph, $S \subset V$ is a subset of nodes, m is the multiplicity (number of attempts), and $\mathcal{E}$, $\mathcal{R}$, $\mathcal{A}$ represent arbitrary functions for node expansion, refinement and aggregation, respectively. The sets V^+, V^-, E^+, E^- are defined as follows.

Transformation	**Symbol**	$\mathbf{V}^+$	$\mathbf{V}^-$	$\mathbf{E}^+$	$\mathbf{E}^-$
Decompose	ϕ_{dec}	$\{\mathcal{E}(v) \mid v \in S\}$	$\emptyset$	$\{(u, v) \mid u \in S, v \in V^+\}$	$\emptyset$
Solve	ϕ_{sol}	$\{\mathcal{S}(v) \mid v \in S\}$	$\emptyset$	$\{(u, v) \mid u \in S, v \in V^+\}$	$\emptyset$
Refine	ϕ_{ref}	$\{\mathcal{R}(t) \mid t \in S\}$	$\emptyset$	$\{(u, v) \mid u \in S, v \in V^+\}$	$\emptyset$
Reduce	ϕ_{red}	$\emptyset$	S	$\emptyset$	$\{(u, v) \mid u \in S \vee v \in S\}$
Aggregate	ϕ_{agg}	$\mathcal{A}(S)$	$\emptyset$	$\{(u, v) \mid u \in S, v \in V^+\}$	$\emptyset$

A Thought Graph Transformations

The full set of considered transformations is shown in Table 2.

B Static Thought Graph Transformation Schedule

Algorithm 1 represents a standard divide-and-conquer strategy. The function $\Delta(G^a_\tau, G^b_\tau)$ outputs all nodes present in the first graph $G^a_\tau = (\mathcal{V}_a, \mathcal{E}_a)$ but not in the second $G^b_\tau = (\mathcal{V}_b, \mathcal{E}_b)$, defined formally as follows.

$$\Delta(G^a_\tau, G^b_\tau) = \{v \mid v \in \mathcal{V}_a \ \ \& \ \ v \notin \mathcal{V}_b\} \quad (3)$$

Algorithm 1 Static Thought Graph Transformation Schedule

Require: Starting graph G^0_τ, allow reduce R_{ed}, allow refine R_{ef}
Require: Solve multiplicity S^m, aggregate multiplicity A^m, and refine multiplicity R^m_{ef}
$G^{dec}_\tau \leftarrow \phi_{dec}(G^0_\tau, 1, \{0\}))$
$G^{sol}_\tau \leftarrow \phi_{sol}(G^{dec}_\tau, S^m, \Delta(G^{dec}_\tau, G^0_\tau))$
$G^{agg}_\tau \leftarrow \phi_{agg}(G^{sol}_\tau, A^m, \Delta(G^{sol}_\tau, G^{dec}_\tau))$
if R_{ed} **then**
  $G^{red}_\tau \leftarrow \phi_{red}(G^{agg}_\tau, 1, \Delta(G^{agg}_\tau, G^{sol}_\tau))$
else
  $G^{red}_\tau \leftarrow G^{agg}_\tau$
end if
if R_{ef} **then**
  $G^{ref}_\tau \leftarrow \phi_{ref}(G^{red}_\tau, R^m_{ef}, \Delta(G^{red}_\tau, G^{agg}_\tau))$
  $G^*_\tau \leftarrow \phi_{red}(G^{ref}_\tau, 1, \Delta(G^{ref}_\tau, G^{red}_\tau))$
else
  $G^*_\tau \leftarrow G^{red}_\tau$
end if
Return: G^*_τ

C Benchmarks

We consider two popular tasks for topological reasoning with LLMs, which are amenable to a divide-and-conquer strategy (i.e. decomposition, solving subproblems and merging): list sorting and set intersection. Despite their simplicity, prior works have shown that these tasks are extremely challenging for LLMs with direct prompting (Besta et al., 2024a).

As mentioned in Section 4, the core results are reported using HumanEval and set-intersection. The list-sorting task was used for illustration purposes in estimating transition probabilities, performing ablation studies and evaluating failure modes.

Sorting: involves sorting a list of numbers between 0 and 9 in ascending order. The error function $\mathcal{E} = X + Y$ has its subterms defined in Equation 4, where a is the input list and b is a candidate solution. X corresponds to the number of incorrectly sorted pairs, while Y corresponds to the frequency difference between a and b for each digit.

$$\begin{aligned} X &= \sum_{i=1}^{m-1} \text{sign}(\max(b_i - b_{i+1}, 0)) \\ Y &= \sum_{i=0}^{9} ||\{b_p : b_p = i\}| - |\{a_q : a_q = i\}|| \end{aligned} \quad (4)$$

Set Intersection: involves finding the intersection of sets A and B. The error function is defined in Equation 5, where C is the candidate solution. The first and second terms correspond to missing and extra elements, respectively.

$$\mathcal{E} = |(A \cap B) \setminus C| + |C \setminus (A \cap B)| \quad (5)$$

D Static Schedule Parameter Search

As described in Section 2, a static transformation can be characterized using a set of discrete param-

eters. We ran bayesian search using using Tree-structured Parzen Estimator (TPE) sampling to determine each parameter, establishing strong baselines for each task.

Table 3: Search space for each parameter characterizing a static transformation.

	Parameter	Search Space
R_{ed}	Allow reduction	$\{0, 1\}$
R_{ef}	Allow refinement	$\{0, 1\}$
S^m	Solve multiplicity	$\{1, 5, 10, 15, 20\}$
A^m	Aggregate multiplicity	$\{1, 5, 10, 15, 20\}$
R^m_{ef}	Refine multiplicity	$\{1, 5, 10, 15, 20\}$

The search space is shown in Table 3. We run multi-objective search to concurrently minimize the task-specific error function $\mathcal{E}$ (Section C) and associated cost, measured as $|\Phi(\omega)|$ where $\Phi(\omega) = (\phi_0, \ldots, \phi_m)$ is a tuple enumerating thought graph transformations, as a function of the schedule parameters $\omega \in \Omega$, where Ω is the search space. Note that $|\Phi(\omega)|$ correlates with the number of LLM queries, meaning this formulation aims to minimize exploration cost.

In selecting parameter configurations, we use the cost function in Equation 6, such that the objectives of cost and error minimization are balanced through the scalar constant $\alpha \in (0, 1)$. We aim to assign equal importance to the cost and error objectives by tuning α independently for each task such that the mean value of the first term matches the second term, i.e. $\alpha E[\mathcal{E}] = (1-\alpha)E[|\Phi(\omega)|)]$, or equivalently $\alpha = \frac{E[|\Phi(\omega)|]}{E[\mathcal{E}+|\Phi(\omega)|]}$ where E denotes the expected value. The expectations are obtained with random sampling.

$$\min_{\omega} [\alpha\mathcal{E} + (1-\alpha)|\Phi(\omega)|] \tag{6}$$

Search was conducted separately on Llama-3.1-70B and Llama-3.1-405B. For sorting and set intersection tasks, search is conducted separately for each difficulty level, ensuring the chosen parameters are adapted to the task. Note that we present three search checkpoints GoT_n for $n \in \{25, 50, 100\}$, where n corresponds to the percentage of trials until convergence. We define the convergeance point as the first iteration where a rolling window J of size 20 matches the condition $J^k = J^{k-1}$. This enables comparing our proposed LLM-guided approach to optimized search schedules at various search budgets.

Table 4: Results from GoT static schedule parameter search on Llama-3.1-405B.

Task	Alpha (α)	GoT_{25}	GoT_{50}	GoT_{100}
sorting32	0.99	0.38	0.38	0.37
sorting64	0.96	4.85	4.49	3.84
sorting128	0.84	28.76	25.76	24.36
set32	0.99	0.16	0.16	0.12
set64	0.99	0.71	0.51	0.31
set128	0.98	3.51	3.51	2.99

The complete search results for Llama-3.1-405B are shown in Table 4. It can be seen that tasks with higher decomposition depth incur lower values of α due to the higher magnitude of the error function. sorting64, sorting128 and set-intersection64 show a smooth decline in the cost function, while the remaining tasks remain at local minima until close to the end of the search. The non-convexity of the search space highlights the cost associated to optimize the parameter set associated with static transformations.

E Transition Probability Profiling

In this section, we estimate the transition probabilities for each thought graph transformation across a number of tasks to gain insight into factors impacting a thought graph formulation of each reasoning problem. For ϕ_{ref}, we define a successful transition when $\mathcal{E} = 0$ for the resulting node, considering only cases when the transformation is executed on nodes previously containing errors. In transformations requiring LLM calls, the transition probability between two states is a random process governed by the token distribution parametrized by the LLM. When LLM calls are not required, i.e. the transformation is implemented through simple node manipulation, the transition probability is 1.

The results are summarized in Table 7. We observe the refinement transformation has notably low success probability, particularly in coding and sorting tasks. Additionally, sorting is the only task with non-deterministic aggregation, which is a potential error source. We note that the performance of a thought graph formulation depends on the ability of the policy agent to capture the success profile of various transformations for a task, and adapt the exploration strategy accordingly.

Table 5: Failure mode 1 results. Mean value of the error $\mathcal{E}$ (↓) for benchmarks with low decomposition depth. Llama-3.1-70B was used for the reasoning and policy agents.

Method	Direct Prompting	GoT$_{25\%}$	GoT$_{50\%}$	GoT$_{100\%}$	ARIES
sorting32	2.2	0.82	0.95	**0.73**	1.29
set-intersection32	1.05	0.41	**0.0**	0.37	1.22

Table 6: Failure mode 2 results. Mean value of the error $\mathcal{E}$ (↓) and search cost C in terms of number of queries (↓). Both the reasoning and policy agents are LLaMA-405B.

Method	Direct Prompting		GoT$_{25\%}$		GoT$_{50\%}$		GoT$_{100\%}$		ARIES	
Metrics	$\mathcal{E}$	C	$\mathcal{E}$	C	$\mathcal{E}$	C	$\mathcal{E}$	C	$\mathcal{E}$	C
sorting32	0.6	**1**	0.74	825	0.82	1650	0.28	3300	**0.22**	20
sorting64	5.07	**1**	2.22	1671	**2.74**	3343	3.46	6687	9.15	48
sorting128	12.75	**1**	13.96	2444	**12.65**	4888	18.65	9776	32.74	48

Table 7: Esimated transition probabilities for each thought graph transformation, taken as the number of successful state transitions in a static schedule.

	$\phi_{\mathbf{sol}}$	$\phi_{\mathbf{ref}}$	$\phi_{\mathbf{red}}$	$\phi_{\mathbf{agg}}$
HumanEval	0.77	0.29	1	1
sorting32	0.57	0.12	1	0.60
set-intersection32	0.75	0.71	1	1

F Failure Modes

In this section, we perform a number of empirical studies aiming to understand the main limiting factors impacting the performance of LLM policy agents on interactive thought graphs. We find there are two major failure modes, described as follows.

❖ *Failure mode 1: LLM Parameter Count*

We find that LLMs with insufficiently large parameter sizes exhibit limited performance when utilized as policy agents on thought graph environments. We deploy Llama-3.1-70B as policy and reasoning agents in sorting and set intersection tasks, against which the larger LLM (Llama-405B) was shown to perform well as a policy agent. As shown in Table 5, LLM-guided graph exploration (ARIES) did not outperform static schedule baselines in this scenario. These findings are consistent with (Wei et al., 2022), which demonstrated that zero-shot chain-of-thought reasoning abilities emerges in models beyond 175B parameters.

❖ *Failure mode 2: Decomposition Depth*

We examine the impact of decomposition depth by analyzing the results in the sorting task, shown in Table 6. We observe LLM policy agents lead to a 21% performance improvement relative to the most optimized static baseline in sorting32, which has a decomposition depth of 2. However, as discussed in Appendix E, the sorting task presents a particular challenge due to the lower success probability of the aggregation transformation. As the complexity and decomposition depth of a task increases, the policy agent is required to apply a higher number of aggregation transformations. Therefore, we observe up to $4.12\times$ and $2.6\times$ performance deterioration in sorting64 and sorting128, respectively. Through empirical analysis, we observe that in the latter tasks, the ϕ_{agg} transformation constitutes 86% and 68% of all policy agent errors, respectively. As such, we conclude that high decomposition depths present a significant failure mode for LLM-guided thought graph exploration, particularly in tasks with low success transition probabilities for the aggregation transformation.

G Ablation Studies

As discussed in Section 3, two factors that impact the performance of LLMs as policy agents in interactive thought graph environments are the size of the ensemble and the use of chain of thought reasoning to enhance the planning abilities of the policy agent. In this section, we aim to understand the impact of each factor by evaluating sorting tasks over a range of ensemble sizes from 1 to 15, with and without CoT prompting in the policy agent.

As shown in Figure 4, as the ensemble size increases to 5, CoT prompting leads to large performance improvements, though the benefits start diminishing beyond this point. Without CoT prompt-

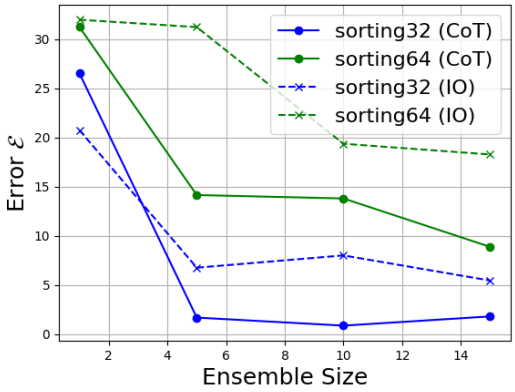

Figure 4: Mean error (y-axis) obtained in the sorting32 task over a sweep of ensemble sizes (x-axis). Llama-3.1-70B was used as the policy agent.

ing, the trend is less consistent, and larger ensemble sizes sometimes yield worse performance. Additionally, errors without CoT are higher for both tasks at any ensemble size. This highlights the necessity of CoT prompting in enhancing the LLM policy agent's ability to adapt from feedback and drive thought graph transformations.