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

Anonymous ACL submission

Abstract

Recent research has shown that LLM performance 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 which are subject to a set of searched hyperparameters. In this work, we view thought graph transformations as actions in a Markov decision process, and implement policy agents to drive effective action policies for the underlying reasoning LLM agent. In particular, we investigate the ability for another LLM to act as a policy agent on thought graph environments and introduce ARIES, a multi-agent architecture for reasoning with LLMs. In ARIES, reasoning LLM agents solve decomposed subproblems, while policy LLM agents maintain visibility of the thought graph states, and dynamically adapt the problem-solving strategy. Through extensive experiments, we observe that using off-the-shelf LLMs as policy agents with no supervised fine-tuning (SFT) can yield up to 29% higher accuracy on HumanEval relative to static transformation schedules, as well as reducing inference costs by 35% and avoid any search requirements. We also conduct a thorough analysis of observed failure modes, highlighting that limitations on LLM sizes and the depth of problem decomposition can be seen as challenges to scaling LLM-guided reasoning.

1 Introduction

Prior works have shown that Large Language Models (LLMs) are subject to the emergence of abilities as their parameter count grows (Wei et al., 2022), which spurred significant interest in training increasingly larger models. However, 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 shift towards inference-time compute scaling can be intuitively understood through the Dual Process Theory, which postulates the existence of two distinct modes of reasoning in humans - (1) a fast, intuitive mode and (2) a slow, deliberate mode (Evans and Frankish, 2009). While the autoregressive decoding procedure of LLMs resembles System 1, prior works used LLMs in System 2 reasoning by inducing models to thoroughly explore a problem, such as using chain of thoughts, ahead of providing a solution to the user query (Wei et al., 2023).

System 2 reasoning can be induced in LLMs by querying models fine-tuned on extensive reasoning traces (Muennighoff et al., 2025). While such single-query approaches have been shown effective in improving the quality of complex sequential logic, an alternative approach involves *performing multiple queries with the same LLM* and arranging intermediate solutions (or “thoughts") in a specified topology, i.e. topological reasoning (Besta et al., 2024b). This approach yields benefits in problems where intermediate solutions can be reliably scored through a Process Reward Model (PRM) (Snell et al., 2024) or using real feedback from external environments (Yao et al., 2023a). Additionally, a graph formulation has shown promising results in problems displaying the property of decomposability into subproblems that can be solved independently then aggregated through a sequence of graph transformations (Besta et al., 2024a). In this work, we focus on *problems with the decomposability property* and in environments where external feedback is viable and useful, such as using LLMs to solve coding problems.

Despite the benefits of topological reasoning, prior works rely on pre-determined traversal strategies parametrized by a discrete set of hyperparameters. This approach lacks generality, as these pa-

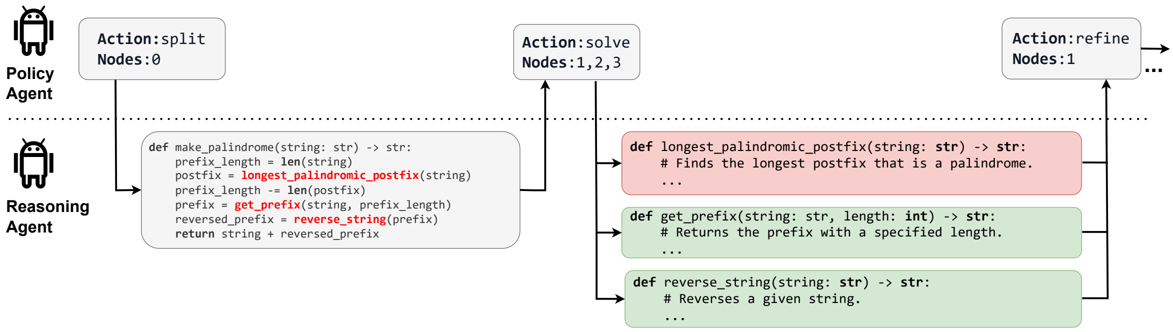

Figure 1: ARIES workflow in answering the HumanEval prompt: "Find the shortest palindrome that begins with a supplied string". The policy agent selects an action based on the thought graph state, which is executed by the reasoning agent. First, the split action generates a skeleton implementation calling yet-to-implement subfunctions, decomposing the problem. Then, the agent is instructed to generate a solution for each subfunction. Since one of the solutions doesn't pass its testcases, the reasoning agent is instructed to refine it based on execution feedback.

rameters must be tuned manually or through extensive Bayesian search to achieve high query efficiency, due to the varying characteristics of each task. With this limitation in mind, we hypothesize that the generalization of artificial problem-solving towards (or beyond) human-like abilities in arbitrary domains requires a mechanism for autonomous traversal of a solution space, falling outside the constrained scope of static schedules shown in Tree-of-Thoughts (Yao et al., 2023a) and Graph-of-Thoughts (Besta et al., 2024a).

To this end, we propose viewing thought graphs as an interactive environment where a sequence of graph transformations is seen as actions in a Markov Decision Process (MDP). Considering this state-action formulation, an effective action policy should explore the solution space to yield a solution while learning from external feedback. Such a mechanism would present a step towards general intelligent agents capable of leveraging existing world knowledge while adapting to out-of-distribution tasks.

Motivated by recent improvements in LLM planning and reasoning (Wei et al., 2023; Yao et al., 2023b), we aim to investigate whether existing LLMs have the capability to act as autonomous reasoning agents by formulating thought graphs as interactive environments. We propose the use of LLM policy agents (i.e. LLM-based action planners) to autonomously execute a set of transformations, including thought proposal, evaluation, aggregation and refinement. As such, we consider the following research questions: **(1)** Can LLMs act as policy agents and effectively utilize feedback from thought graph environments to dynamically tune their exploration strategies? **(2)** Can this approach match the performance of static transformation schedules extensively optimized for a given task? And finally, **(3)** What are the failure modes of using existing LLMs as policy agents in guiding thought graph exploration (i.e. factors affecting the ability to produce coherent exploration plans)?

We investigate the aforementioned questions by implementing 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. In summary, our contributions are as follows.

- We introduce ARIES, a novel formulation to autonomous topological reasoning, making the whole reasoning task LLM-guided. We frame the topological reasoning task as a collaboration between two agents within a topological thought graph. The LLM policy agent assesses states and determines the actions, while the LLM reasoning agent carries out these actions, executing transformations on the thought graph.

- We show that LLMs exhibit planning capacity and can serve effectively as policy agents on topological reasoning tasks, thus eliminating the requirement for predefined, task-specific scheduling of the reasoning agents, as seen in Tree-of-Thoughts (ToT) and Graph-of-Thoughts (GoT). Additionally, we identify and discuss the limitations and failure modes of their planning abilities.

- We perform carefully controlled experiments against a number of benchmarks, showing that LLM-guided thought graph exploration can lead to up to 29% higher accuracy at 35% lower inference cost, as well as obviating any Bayesian search cost.

2 Related Work

2.1 Topological Reasoning

(Wei et al., 2023) pioneered the elicitation of step-by-step logical reasoning, with subsequent work by (Wang et al., 2023) demonstrating improved performance through the sampling and arbitration along multiple reasoning sequences. (Yao et al., 2023a) formulate concurrent exploration of multiple reasoning paths by scoring reasoning steps, leveraging tree search algorithms (ToT). Finally, (Besta et al., 2024a) generalize problem space exploration by formulating thoughts as a graph, enabling the use of arbitrary transformations such as node refinement and aggregation (GoT).

Several works have explored methods of improving the query efficiency of topological reasoning, which suffers from high computational demand due to iterative LLM prompting (Hu et al., 2023; Sel et al., 2024; Ding et al., 2024). Despite improvements, few works have targeted the generality of this approach by exploring dynamic transformations. While (Yao et al., 2023a) leverage standard tree search algorithms, (Long, 2023) hypothesize that tree search can be enhanced through trained policy networks to guide node backtracking. However, this idea is not explored fully and their evaluation is focused on heuristics-based rules. As such, our work presents the first effort towards generalized topological reasoning through autonomous thought graph exploration.

2.2 LLMs as Action Policy Agents

Significant research has focused on leveraging LLMs for guiding action policies, such as in tasks requiring coordination of heterogeneous model ensembles (Shen et al., 2023). LLMs have also been deployed as action planners in interactive environments where feedback is provided to the action scheduler, such as solving computer tasks (Kim et al., 2023) and online shopping (Yao et al., 2023b). However, some works have outlined the instability in obtaining action plans over long-range horizons, where LLMs have been shown to repeatedly generate invalid action plans (Xie et al., 2023). This limitation has been tackled by works such as (Shinn et al., 2023), which propose an episodic memory buffer of previous trials. However, to our knowledge, no prior work has investigated leveraging LLM planning abilities in the context of topological reasoning.

3 Topological Reasoning with Large Language Models

We consider a reasoning problem to be stated in language as an ordered tuple of tokens $p = (t_1, \ldots, t_m)$, where each token $t \in \mathcal{V}$ belongs to a vocabulary space $\mathbb{V}$. We define a thought $\tau = (t_1, \ldots, t_j)$ as a sequence of tokens sampled autoregressively from an LLM parametrized by θ, 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 to the problem.

A thought sequence can 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 tree T_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. The tree can be parametrized with a depth of d and a width of w, denoting the number of nodes per level. 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. Hence, tree-based thought exploration involves finding a path $\mathcal{P} \subset \mathcal{V}$ that maximizes the cumulative value of thoughts, as follows.

$$P^* = \arg\max_{\mathcal{P}} \sum_{\tau \in \mathcal{P}} \lambda(\tau) \qquad (1)$$

A thought graph G_τ can also be represented via the tuple $(\mathcal{V}, \mathcal{E})$, with no imposed restriction on the arrangement of thoughts and edges. 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. The full set of considered transformations and their formulations are shown in Table 6.

$$G_\tau^* = \phi_m(\ldots(\phi_1(\phi_0(G_\tau^0)))) \qquad (2)$$

Table 1 summarizes the thought graph transformations we consider in the rest of this work. ϕ_{dec} decomposes a reasoning problem into subproblems

Table 1: Thought graph transformations used to solve reasoning problems using a divide-and-conquer strategy. See Appendix B for their complete definitions.

Transformation	**Symbol**
Decompose	ϕ_{dec}
Solve	ϕ_{sol}
Refine	ϕ_{ref}
Reduce	ϕ_{red}
Aggregate	ϕ_{agg}

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.

Static Transformation Schedules: A static transformation schedule can be parametrized by the tuple $(R_{ed}, R_{ef}, S^m, A^m, R^m_{ef})$. S^m, A^m, R^m_{ref} represents the multiplicity (i.e. number of attempts) of the solve, aggregate and refine transformations, respectively. $R_{ed}, R_{ef} \in \{0, 1\}$ indicate whether the ϕ_{red} and ϕ_{ref} transformations are applied after aggregation.

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

In Algorithm 1, each transformation is defined as $\phi(G_\tau, m, S)$, where $G_\tau = (V, E)$ is a thought graph, $S \subset V$ is a subset of nodes and m is the multiplicity (number of attempts). Additionally, 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.

$$\Delta(G^a_\tau, G^b_\tau) = \{v | v \in \mathcal{V}_1 \ \ \& \ \ v \notin \mathcal{V}_2\} \qquad (3)$$

Algorithm 1 represents a standard divide-and-conquer strategy. The ϕ_{dec} transformation decomposes the starting problem into B subproblems, which are solved individually (ϕ_{sol}). The aggregation of the subproblem solutions is attempted A^m times, as the ϕ_{agg} transformation has a non-zero probability of failure. 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 wth ϕ_{ref}, and the highest-scoring attempt is kept as the final solution.

4 Thought Graph Exploration as a Markov Decision Process

Beyond the fixed schedule shown in Algorithm 1, the transformation of a thought graph can be generalized as a Markov decision process $(\mathcal{S}, \mathcal{A}, \mathcal{P}_a)$:

- **State** $s_t \in \mathcal{S}$: represents an arrangement of nodes and edges in the thought graph, with the associated value of each node, i.e. $s_t = (\mathcal{V}, \mathcal{E}, \{\lambda(v) | v \in \mathcal{V}\})$.
- **Action** $a \in \mathcal{A}$: indicates which transformation to perform on the thought graph, and which nodes to perform it on, 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 (Table 6).
- **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.

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

We bound the number of queries by the constant ϵ, as in the limit $|\Phi| \to \infty$, $P(s^+ | s^0, \Phi) \to 1$.

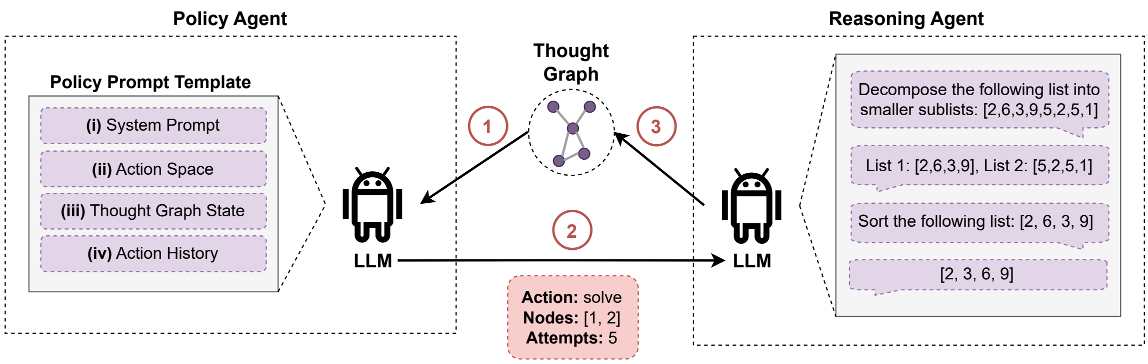

Figure 2: Multi-agent framework for reasoning over thought graphs. First, (1) the policy agent an action and subset of nodes given a prompt including (i-ii) general instructions and (iii-iv) an overview of the exploration state. The sample is then (2) passed to the reasoning agent, which finally (3)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

4.1 Multi-Agent Reasoning

In this work, we hypothesize that LLMs can approximate a solution to the stated optimization problem by acting as policy agents. We develop an 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 6). 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.

4.2 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. The analysis includes the following:

1. Describe the action history and how each action relates to an exploration strategy.
2. Describe the thought graph state, and how each node corresponds to previous actions.
3. Discuss the outlined strategy, stating whether it is successful, unsucessful, or pending.
4. Outline a number of options for the next action, detailing the expected outcome of each.

4.3 Policy Agent Ensembles

Given the stochastic nature of token prediction in LLMs, we observe high variability in the chosen action over several invocations of a policy agent under the same thought graph state. Given the preconditions and effects of each action are represented via text rather than any rigorous formulation, actions selected by the policy agent can display flawed understanding of the problem constraints, leading to ineffective exploration of the thought graph. To overcome this limitation, 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 Section 6 for ablation studies on the impact of policy agent ensemble size on reasoning performance.

5 Experiments

Through a range of controlled experiments, we evaluate the performance of LLM policy agents on interactive thought graphs. In Appendix D and Section 5.2, we define the benchmarks and baselines.

We present the core results across each benchmark task in Section 5.3. We profile the transition probabilities of each thought graph transformation across tasks in Section 5.4. In Section 5.5, we provide empirical results demonstrating two main failure modes of LLMs as policy agents, namely model size and decomposition depth.

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. Llama-3.1-70B was hosted with 8× A6000 GPUs. Llama-3.1-405B was hosted using 16× H100 GPUs distributed over 4 nodes. The total cost was approximately 3k GPU hours.

5.1 Benchmarks

We run our main evaluation on HumanEval, a widely used benchmark for assessing the functional correctness of code generation models through a set of Python programming problems with corresponding test cases (Chen et al., 2021).

Additionally, we consider two popular tasks for topological reasoning with LLMs, 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), benefitting from a divide-and-conquer strategy (i.e. decomposition, solving subproblems and merging). We evaluate these at various levels of difficulty (quantified by the size of the lists and sets), resulting in six benchmarks: sorting32/64/128 and set-intersection32/64/128.

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 D. Additionally, we report both the search C_s and inference cost C_i. We measure cost by the number of queries since we observe a low standard deviation in the number of generated tokens across all LLM queries during our experiments.

5.2 Baselines

We use static transformation schedules as the baseline, following (Besta et al., 2024a). As previously noted, static schedules require extensive, task-dependent hyperparameter tuning. For each individual task, we carefully tune the hyperparameters using Bayesian optimization resulting in three variants: $\text{GoT}_{25\%}$, $\text{GoT}_{50\%}$ and $\text{GoT}_{100\%}$. Here, the percentage corresponds to the number of trials spent until the hyperparameter search converges.

Table 2: Task accuracy (↑), search and inference costs (↓) on Human Eval. Cost is measured as the number of LLM queries. IO refers to direct prompting. Llama-405b was used for the reasoning and policy agents.

Method	Accuracy [%]	Search Cost (C_s)	Inference Cost (C_i)
IO	77.4	0	1
$\text{GoT}_{25\%}$	66.3	1160	34.8
$\text{GoT}_{50\%}$	67.5	2368	24.3
$\text{GoT}_{100\%}$	60.1	4742	8.17
ARIES	**89.0**	**0**	**5.3**

As such, we compare against baselines with several search compute budgets. See Appendix C for details on the full search methodology. We also consider an Direct IO (Input-Output) baseline, i.e. reasoning via direct LLM prompting.

5.3 Evaluation

Replacing static transformation schedules with LLM policy agents offers generalization to arbitrary tasks at no tuning cost. However, performance may be constrained by the LLM's planning capabilities. As such, we evaluate ARIES against the aforementioned benchmarks, demonstrating its advantages and identifying potential failure modes. We set the policy agent ensemble size to 5 in all experiments, as explained in Section 6.

5.3.1 HumanEval

Our key findings for autonomous policy agents in the context of a coding task are shown in Table 2. 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. We also observe that as further trials are expended in the GoT baseline search, the query efficiency is increased, i.e. hyperparameter configurations are found that achieve similar performance levels at lower query counts. Nevertheless, we achieve 54% lower inference cost on average compared to even the most optimized GoT baseline, and also avoids any search time requirement.

5.3.2 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

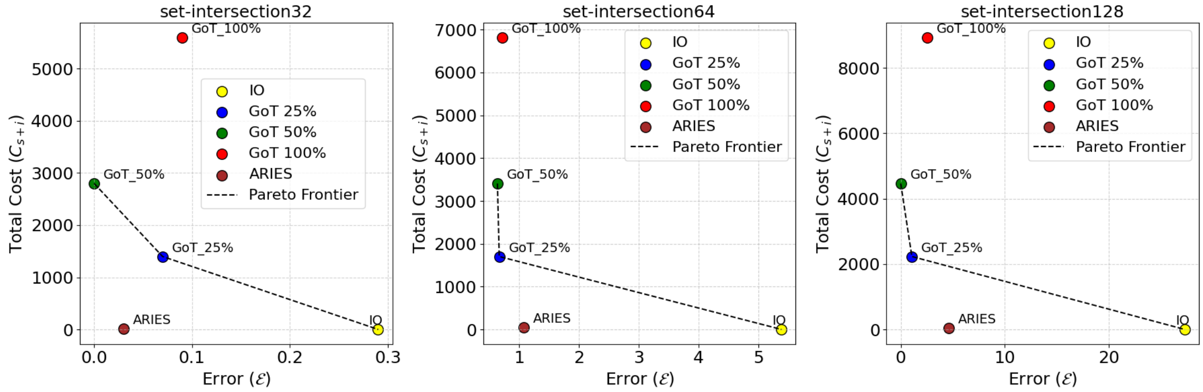

Figure 3: Pareto frontiers in total query cost (C_{s+i}) and task error ($\mathcal{E}$) for set intersection tasks at various difficulty levels. The total cost is the number of queries expended at search and inference time. Llama-3.1-405B was used for the reasoning and policy agents. Our results (ARIES)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

Table 3: 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

static schedule baselines and direct prompting. In the set-intersection32 task, we achieve a 2.3× error reduction relative to GoT_{25} while also achieving 116× lower overall cost.

5.4 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 3. 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.

5.5 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 4, 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 5. 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 Section 5.4, 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,

Table 4: 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	$\text{GoT}_{25\%}$	$\text{GoT}_{50\%}$	$\text{GoT}_{100\%}$	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 5: 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		$\text{GoT}_{25\%}$		$\text{GoT}_{50\%}$		$\text{GoT}_{100\%}$		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

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.

6 Ablation Studies

As discussed in Section 4, 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 prompting, 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.

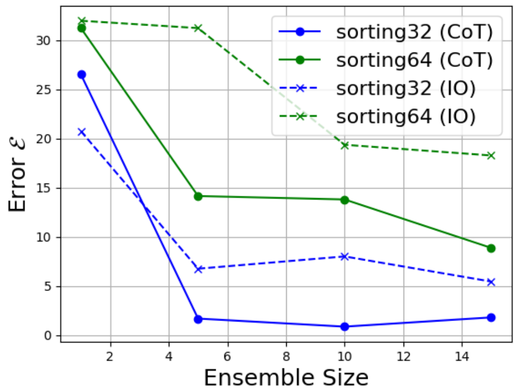

Figure 4: Mean error (y-axis) obtained in the sorting32 task over a sweep of ensemble sizes (x-axis). Llama-3.1-70B was used as the policy agent.

7 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. We show up to 29% higher accuracy on HumanEval while reducing inference costs by 35% compared to static schedules. We identified two key limitations: insufficient model size and excessive decomposition depth on the task at hand. These constraints indicate that while LLMs show promise as reasoning agents, their effectiveness depends on parameter count and task complexity.

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.

Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Wei Zhang, Si Qin, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2024. Everything of thoughts: Defying the law of penrose triangle for thought generation. *Preprint*, arXiv:2311.04254.

Jonathan Evans and Keith Frankish. 2009. *In two minds: Dual processes and beyond*. Oxford University Press.

Pengbo Hu, Ji Qi, Xingyu Li, Hong Li, Xinqi Wang, Bing Quan, Ruiyu Wang, and Yi Zhou. 2023. Tree-of-mixed-thought: Combining fast and slow thinking for multi-hop visual reasoning. *Preprint*, arXiv:2308.09658.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks. *Preprint*, arXiv:2303.17491.

Jieyi Long. 2023. Large language model guided tree-of-thought. *Preprint*, arXiv:2305.08291.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

Bilgehan Sel, Ahmad Al-Tawaha, Vanshaj Khattar, Ruoxi Jia, and Ming Jin. 2024. Algorithm of thoughts: Enhancing exploration of ideas in large language models. *Preprint*, arXiv:2308.10379.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Preprint*, arXiv:2303.17580.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Preprint*, arXiv:2303.11366.

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.

Yaqi Xie, Chen Yu, Tongyao Zhu, Jinbin Bai, Ze Gong, and Harold Soh. 2023. Translating natural language to planning goals with large-language models. *Preprint*, arXiv:2302.05128.

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.

A Limitations

A.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, list sorting, 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 models, and results may not be directly transferable to other architectures.

A.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.

A.3 Failure Modes

Our empirical findings 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.

B Thought Graph Transformations

The full set of considered transformations is shown in Table 6.

C Static Schedule Parameter Search

As described in Section 3, a static transformation can be characterized using a set of discrete parameters. We ran bayesian search using using Tree-structured Parzen Estimator (TPE) sampling to determine each parameter, establishing strong baselines for each task.

The search space is shown in Table 7. We run multi-objective search to concurrently minimize the task-specific error function $\mathcal{E}$ (Section D) 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 4, 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. $\alpha E\left[\mathcal{E}\right] = (1-\alpha) E\left[|\Phi(\omega)|\right]$, or equivalently $\alpha = \frac{E[|\Phi(\omega)|]}{E[\mathcal{E}+|\Phi(\omega)|]}$ where E denotes the expected value. The expectations are obtained

Table 6: 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}	$\{\mathcal{E}(v) \vert v \in S\}$	$\emptyset$	$\{(u,v) \vert u \in S, v \in V^+\}$	$\emptyset$
Solve	ϕ_{sol}	$\{\mathcal{S}(v) \vert v \in S\}$	$\emptyset$	$\{(u,v) \vert u \in S, v \in V^+\}$	$\emptyset$
Refine	ϕ_{ref}	$\{\mathcal{R}(t) \vert t \in S\}$	$\emptyset$	$\{(u,v) \vert u \in S, v \in V^+\}$	$\emptyset$
Reduce	ϕ_{red}	$\emptyset$	S	$\emptyset$	$\{(u,v) \vert u \in S \vee v \in S\}$
Aggregate	ϕ_{agg}	$\mathcal{A}(S)$	$\emptyset$	$\{(u,v) \vert u \in S, v \in V^+\}$	$\emptyset$

Table 7: 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	$\{1, 5, 10, 15, 20\}$

with random sampling.

$$\min_{\omega} \left[\alpha \mathcal{E} + (1 - \alpha)|\Phi(\omega)|\right] \tag{4}$$

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 8: 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 8. 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.

D Benchmarks

We choose 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).

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 5, 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.

$$\begin{aligned} X &= \sum_{i=1}^{m-1} \text{sign}(\max(b_i - b_{i+1}, 0)) \\ Y &= \sum_{i=0}^{9} \left|\left|\{b_p : b_p = i\}\right| - \left|\{a_q : a_q = i\}\right|\right| \end{aligned} \tag{5}$$

Set Intersection: involves finding the intersection of sets A and B. The error function is defined in Equation 6, where C is the candidate solution.

Table 9: Core results for topological reasoning across all tasks and models. We show the mean value of the score function $\mathcal{E}$ ($\downarrow$), which is defined for each task in Section 5. GoT_{100}, GoT_{50}, GoT_{25} represent the obtained values from static schedule parameters obtained at convergeance, 50% and 25% of convergeance trials, respectively.

Task	Llama-70b				Llama-405b			
	GoT_{25}	GoT_{50}	GoT_{100}	GoT_{LLM}	GoT_{25}	GoT_{50}	GoT_{100}	GoT_{LLM}
sorting32	0.82	0.95	**0.73**	1.29	0.74	0.82	0.28	**0.22**
sorting64	4.73	4.73	**4.64**	10.04	**2.22**	2.74	3.46	9.15
sorting128	16.18	**13.86**	16.07	31.79	13.96	**12.65**	18.65	32.74
set-intersection32	0.41	**0.0**	0.37	1.22	0.07	0.0	0.09	**0.03**
set-intersection64	3.40	2.66	**1.27**	7.34	0.67	**0.64**	0.72	1.08
set-intersection128	13.23	12.92	**12.73**	22.98	**1.07**	0	2.54	4.62

The first and second terms correspond to missing and extra elements, respectively.

$$\mathcal{E} = |(A \cap B) \setminus C| + |C \setminus (A \cap B)| \qquad (6)$$