DynaGraph: Lightweight Multi-Model Interaction Framework via Dynamic Topological Reconfiguration

Anonymous ACL submission

Abstract

Tackling complex reasoning tasks typically relies on massive monolithic LLMs, which suffer from severe computational redundancy. While task decomposition through structured pipelines or multi-agent collaborations offers an alternative, these approaches inevitably fall into a critical dilemma: predefined static topologies are highly vulnerable to cascading errors, whereas unconstrained dynamic agents suffer from trajectory divergence and unpredictable memory bloat. To address this, we present *DynaGraph*, a lightweight multi-model framework driven by dynamic topological reconfiguration. At the execution level, *DynaGraph* multiplexes time-division PEFT adapters over a shared base model, enabling both full system training and inference deployment on a single consumer-grade GPU. At the routing level, the Evaluator continuously monitors execution confidence to trigger hierarchical self-healing: *Fine-grained Patching* for localized data gaps and *Subgraph Reconstruction* for severe logical ruptures. Experiments on StrategyQA, MATH, and FinQA demonstrate our 8B model closely approximates the reasoning capabilities of a 72B monolithic model (e.g., 87.6% on StrategyQA, 82.7% on MATH). Furthermore, it reduces latency by up to 68.1% and token consumption by 68.6% compared to unconstrained dynamic architectures.

1 Introduction

In recent years, LLMs have demonstrated unprecedented capabilities in the field of natural language processing (Dubey et al., 2024). However, as technology advances, the application scenarios of LLMs have expanded to complex tasks requiring deep logical deduction, such as multi-hop question answering, mathematical reasoning, and financial risk analysis (Yang et al., 2018; Hendrycks et al., 2021; Wu et al., 2023). Currently, solving such tasks typically relies on extremely large-scale monolithic models, such as GPT-4 (estimated 1.8T parameters) (OpenAI et al., 2023), Claude 3.5 Sonnet, or Llama 3-70B (Dubey et al., 2024). Unfortunately, relying on a single massive model to handle all segments within a task pipeline not only introduces severe computational redundancy but also limits the system's flexibility for modular expansion of specific capabilities.

To mitigate the limitations of monolithic models on complex tasks, existing solutions primarily focus on augmenting the single-model paradigm. Prompting techniques like CoT (Wei et al., 2022b) and ToT (Yao et al., 2023) guide intermediate reasoning, while RAG (Asai et al., 2024) integrates external knowledge. Alternatively, MoE (Jiang et al., 2024) scales capacity via sparse activation. However, these methods face architectural bottlenecks. Despite heuristic pruning, prompt-based paradigms operate within predefined, rigid topologies lacking reactive runtime self-healing, making them vulnerable to cascading failures from localized anomalies. Meanwhile, MoE remains a highly coupled system demanding steep memory overhead, which prevents independent modular hot updates.

Consequently, decomposing tasks into multi-agent systems has emerged as an alternative (Hong et al., 2024). However, current collaborative frameworks struggle to balance controllability and flexibility, typically falling into a dilemma between two extremes: **1) Static DAG Pipelines:** These offer predictable overheads but lack reactive recovery mechanisms. Localized errors inevitably accumulate along the task chain, leading to severe cascading failures (Zhu et al., 2025). **2) Unconstrained Dynamic Reflection:** Autoregressive agents exhibit high autonomy but are prone to trajectory divergence and meta-cognitive hallucinations (Huang et al., 2024; Lu et al., 2026). Their repeated trial-and-error leads to unpredictable computational costs and severe GPU memory bloat. Thus, developing a collaborative system that achieves

convergent, efficient self-correction under a strictly controlled budget remains a critical challenge.

To address these challenges, we present *DynaGraph*, a lightweight multi-model system with dynamic topological reconfiguration. The framework shifts from static task scheduling to adaptive topological evolution via a real-time feedback loop. At the execution layer, *DynaGraph* multiplexes LoRA adapters over a shared backbone, maintaining a constant $\mathcal{O}(1)$ GPU memory footprint regardless of expert pool size. At the control layer, the Evaluator monitors real-time execution confidence, instantly halting flawed reasoning steps. Based on error severity, it adaptively triggers hierarchical interventions: Fine-grained Patching to suture localized information gaps, or Subgraph Reconstruction to truncate and regenerate severely corrupted branches. This paradigm endows the system with robust self-healing, ensuring trajectory convergence under strict operational budgets.

In summary, our contributions are three-fold:

- **Lightweight Architecture:** *DynaGraph* multiplexes time-division PEFT adapters over a shared base model. This bounds the peak memory footprint to 16.6 GB, enabling complex multi-model inference on a single consumer-grade GPU.
- **Adaptive Topological Self-Healing:** A state-aware reconfiguration mechanism balances execution structure and autonomy. By dynamically deploying *Fine-grained Patching* to suture localized data gaps and *Subgraph Reconstruction* to truncate and regenerate fatally corrupted logic branches, the system guarantees reasoning convergence while halting cascading errors.
- **Efficiency and Performance Breakthrough:** Our 8B-scale *DynaGraph* achieves superior performance (e.g., 87.6% on StrategyQA, 82.7% on MATH), closely approximating 72B massive models. Compared to unconstrained dynamic architectures like Reflexion, it reduces token consumption and latency by up to 68.6% and 68.1%.

2 Background

2.1 Large Language Models

Large Language Models (LLMs) (e.g., Gemini (Team et al., 2023) and Qwen (Yang et al., 2025)) exhibit remarkable capabilities across domains like mathematical reasoning (Cobbe et al., 2021) and commonsense question answering (Talmor et al., 2019), propelled by scaling laws (Kaplan et al., 2020) and emergent abilities (Wei et al., 2022a). However, monolithic LLMs often struggle with complex multi-domain tasks. To address this, models are frequently fine-tuned on specialized datasets. For instance, domain-specific fine-tuning in financial (Wu et al., 2023), biomedical (Lee et al., 2020), and legal (Chalkidis et al., 2020) fields yields superior specialized performance without sacrificing general capabilities. These successes inspire the use of parameter-efficient fine-tuning (PEFT) methods, such as prompt tuning (Li et al., 2025) and LoRA (Hu et al., 2022), for efficient domain adaptation.

2.2 Model Interaction

2.2.1 Single-Model-Based Interaction

To enhance reasoning within a single-model paradigm, techniques like Chain-of-Thought (CoT) (Wei et al., 2022b), zero-shot triggers (Kojima et al., 2022), and task-breakdown prompts (Zhou et al., 2023) guide intermediate logical deductions. Furthermore, Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) mitigates hallucinations by grounding outputs in external corpora. Nevertheless, single-model approaches remain vulnerable to biased prompts (Turpin et al., 2023), error accumulation in long inference chains (Lu et al., 2026), and retrieval-generation misalignment (Huang et al., 2025), limiting their efficacy on complex tasks.

2.2.2 Multi-Model-Based Interaction

Multi-model interactions overcome single-model bottlenecks by decomposing tasks across distinct reasoning modules. Frameworks like Tree of Thoughts (Yao et al., 2023) and Graph of Thoughts (Besta et al., 2024) organize reasoning into structured topologies suitable for branching and backtracking. Concurrently, Mixture-of-Experts (MoE) architectures enhance efficiency and accuracy by routing inputs to relevant experts (Zheng et al., 2025), especially when incorporating fine-grained specialization (Dai et al., 2024) and instruction-tuning (Shen et al., 2024).

Despite their effectiveness, existing multi-model frameworks construct static topologies that cannot adapt to real-time execution states and lack autonomous self-repair mechanisms. *DynaGraph* directly addresses these limitations by introducing

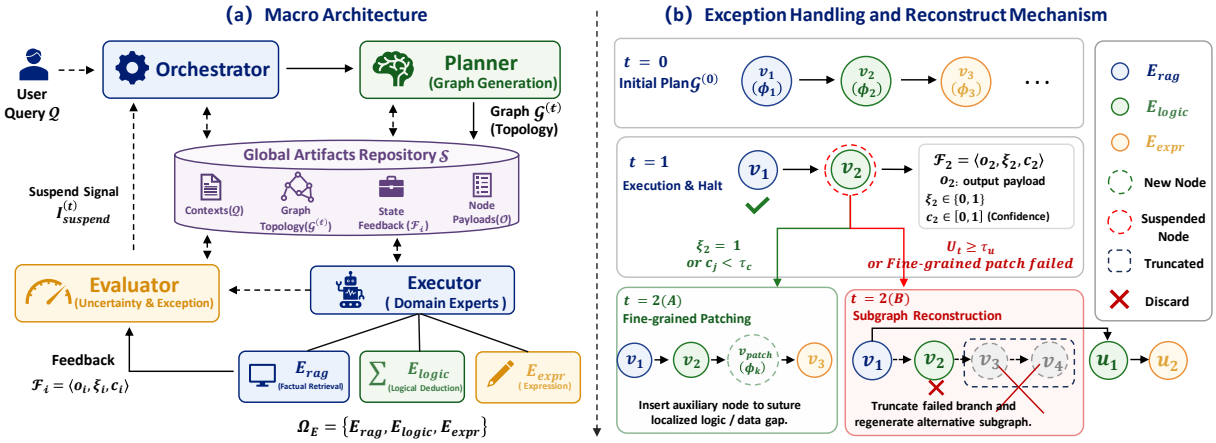

Figure 1: **Macro architecture and dynamic topology evolution of *DynaGraph*.** (a) **Macro Architecture** illustrating component interactions. (b) **Exception Handling and Reconstruct Mechanism** through dynamic node insertion and branch replacement.

dynamic topological reconstruction, pushing multi-model interaction into a resilient, adaptive frontier.

3 Method

3.1 Task Formulation and System Overview

3.1.1 Formalization of Task and DAG

In complex reasoning scenarios, task resolution requires collaborative routing among specialized modules, which we define as the *model interaction topology*. This paradigm decomposes instructions into a graph where nodes represent expert executions and edges denote contextual dependencies.

We propose this to overcome the inherent limitations of existing frameworks: static pipelines are highly vulnerable to cascading errors, while unconstrained dynamic agents suffer from trajectory divergence. To balance structured execution with self-healing capabilities, we introduce a *dynamic* topological reconfiguration mechanism. This allows the system to monitor real-time states and topologically physically restructure the graph to repair logical ruptures and data gaps.

To formalize this dynamic orchestration, given a user query $\mathcal{Q}$, an Orchestrator invokes a Planner to generate an initial task execution graph $\mathcal{G}^{(0)} = (\mathcal{V}^{(0)}, \mathcal{E}^{(0)})$. Unlike static pipelines, our graph topology evolves dynamically with the execution state, defined at time step t as $\mathcal{G}^{(t)} = (\mathcal{V}^{(t)}, \mathcal{E}^{(t)})$.

Each vertex $v_i \in \mathcal{V}^{(t)}$ represents a sub-task node. To decouple heterogeneous expert capabilities, we formalize the vertex as Eq. (1), where $E_{\phi_i} \in \Omega_E = \{E_{rag}, E_{logic}, E_{expr}\}$ denotes the assigned domain expert, q_i represents the local instruction context, and $\mathcal{P}_i = \{v_j \mid (v_j, v_i) \in \mathcal{E}^{(t)}\}$ is the set of prerequisite parent nodes. Directed edges $e_{j,i} \in \mathcal{E}^{(t)}$ define the logic flow. Node v_i is triggered once all prerequisite nodes in $\mathcal{P}_i$ were execute successfully and passed their contexts.

$$v_i = \langle E_{\phi_i}, q_i, \mathcal{P}_i \rangle. \tag{1}$$

To systematically record all expert execution outputs, we introduce a structured state space $\mathcal{S}$. Upon completing its computation, the output state of the assigned expert E_{ϕ_i} at node v_i is encapsulated into a standardized feedback tuple, as shown in Eq. (2).

$$\mathcal{F}_i = \langle o_i, \xi_i, c_i \rangle. \tag{2}$$

where o_i is the structured output of the corresponding expert. The Exception Flag $\xi_i \in \{0, 1\}$ indicates irrecoverable errors (e.g., format deviations or logical ruptures), and $c_i \in [0, 1]$ represents the expert's normalized confidence score.

3.1.2 System Overview

To address aforementioned problems, we propose *DynaGraph* framework. As illustrated in Fig. 1 (a), our Framework decouples complex task resolution into four canonical phases: Planning, Execution, Evaluation, and Reconstruction. Upon receiving a user query $\mathcal{Q}$, the central Orchestrator prompts the Planner to generate an initial task graph.

Fig. 1 (a) further visualizes the strict module-isolation principle: heterogeneous experts ($E_{\phi_i} \in \Omega_E$) never communicate directly; instead, all intermediate contexts, graph topologies, and node payloads flow through the Global Artifacts Repository $\mathcal{S}$. As experts return state feedback $\mathcal{F}_i$, the Evaluator monitors the execution stream to quantify uncertainty and detect exceptions. If lethal anomalies or high uncertainties exceed thresholds,

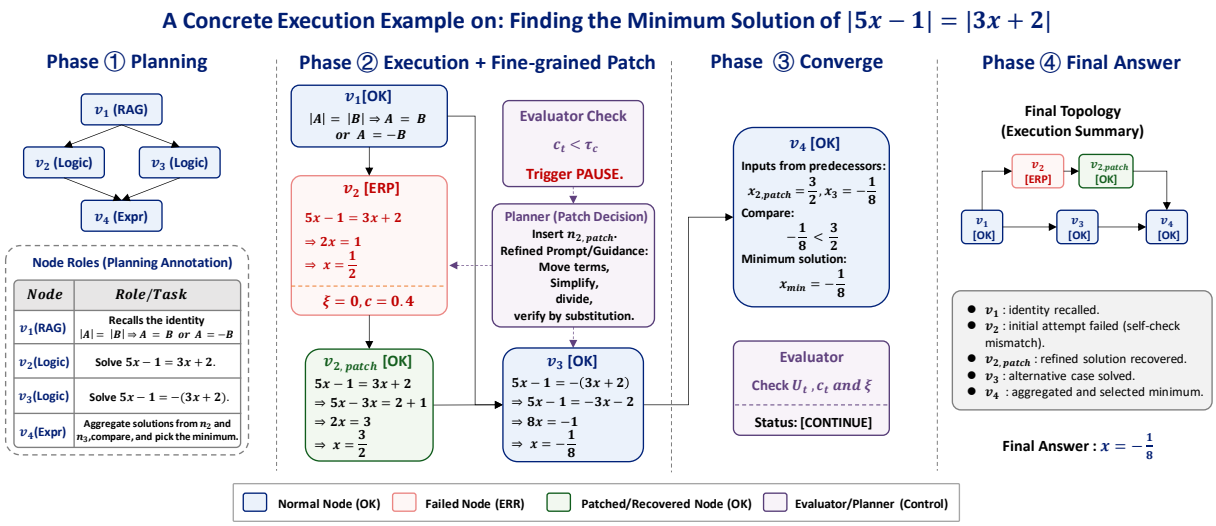

Figure 2: **A concrete execution example of *DynaGraph*.** The Evaluator halts execution upon detecting anomalies at node v_2; a dynamically inserted patch node subsequently recovers the corrupted state to ensure convergence.

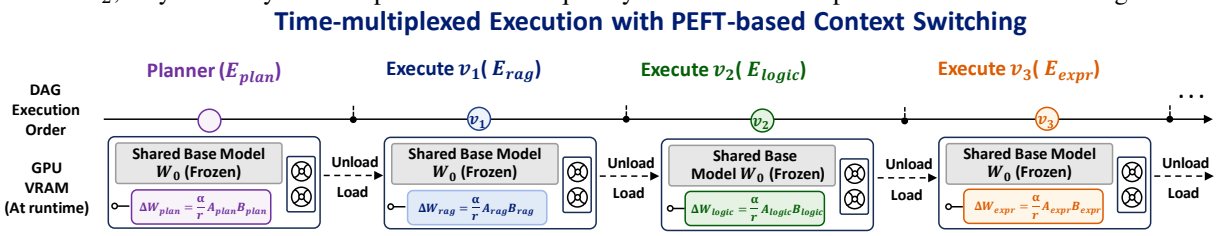

Figure 3: **Time-multiplexed Execution with PEFT-based Context Switching:** The system maintains a frozen shared base model (W_0) and dynamically loads task-specific LoRA adapters (e.g., $A_{rag}B_{rag}$) via PEFT

the Evaluator issues a suspension signal ($\mathbb{I}^{(t)}_{suspend}$), returning control to the Orchestrator. As illustrated in Fig. 1 (b), the Planner then executes hierarchical topological reconfiguration: 1) fine-grained patching for localized data errors or gaps (e.g., erroneous intermediate calculations or unretrieved factual entities), or 2) subgraph reconstruction to truncate and regenerate failed branches.

Fig. 2 illustrates a life cycle on finding the minimum solution of $|5x-1| = |3x+2|$. **Phase** ①: The Planner generates a DAG with nodes v_1 through v_4. **Phase** ②: After v_1 succeeds, the Evaluator detects a calculation error at v_2 and triggers suspension. The Planner inserts a patch node $v_{2,patch}$ to recalculate the faulty nodes. **Phase** ③: The corrected state from $v_{2,patch}$ propagates downstream alongside the valid sibling output to the aggregation node v_3. **Phase** ④: The system emits the final topology and answer, validating that the system can autonomously and dynamically repair faults.

3.2 Domain-Specific Expert Design

3.2.1 Structured State Feedback of Experts

Given that each expert is tailored for a distinct function, their structured outputs vary accordingly. Therefore, we formally define these specific structures below. This framework constructs an expert pool $\Omega_E = \{E_{rag}, E_{logic}, E_{expr}\}$ characterized by functional orthogonality. Upon termination, each node v_i returns a standardized state feedback tuple $\mathcal{F}_i = \langle o_i, \xi_i, c_i \rangle$ to the Orchestrator. The specific variables for each expert are constrained as follows:

① **Factual Retrieval Expert** E_{rag}**:** Dedicated to open-domain factual verification. Its output $o_i = \langle \mathcal{A}, \mathcal{K}, \mathcal{C} \rangle$ comprises the assertion set $\mathcal{A}$, external evidence $\mathcal{K}$, and citation provenance $\mathcal{C}$. Confidence c_i measures evidence reliability. If retrieval fails or cannot support assertions, $\xi_i = 1$.

② **Logical Deduction Expert** E_{logic}**:** Dedicated to closed-domain logical verification. Its output $o_i = \langle \mathcal{H}, \mathcal{V} \rangle$ comprises the reasoning history $\mathcal{H}$ and boolean verification results $\mathcal{V}$. Confidence c_i measures logical self-consistency. Upon localized verification failure, $\xi_i = 1$.

③ **Expression Expert** E_{expr}**:** Dedicated to linguistic fidelity. Its output $o_i = \langle \mathcal{D}, \mathcal{U} \rangle$ contains the generated draft $\mathcal{D}$ and unsupported statements $\mathcal{U}$. Confidence c_i measures semantic fidelity. For severe formatting deviations, $\xi_i = 1$.

In practice, the confidence score $c_i \in [0, 1]$ is derived via verbalized self-calibration, prompting experts to explicitly quantify their certainty. The

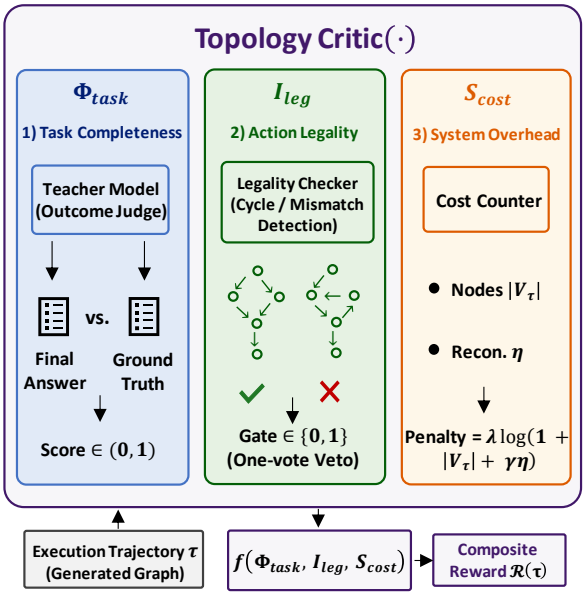

Figure 4: **Topology Critic:** The execution trajectory τ is evaluated by a composite reward function $\mathcal{R}(\tau)$ considering task accuracy/completeness, topological legality, and system overhead.

exception flag $\xi_i \in \{0, 1\}$ triggers deterministically if the expert self-reports anomalies or fails structured parsing.

3.2.2 Context Switching for Low-Memory Usage

To address GPU memory bottlenecks, we implement PEFT based time-division scheduling. For a frozen base model $W_0 \in \mathbb{R}^{d\times k}$, we train low-rank matrices $A_\psi \in \mathbb{R}^{d\times r}$ and $B_\psi \in \mathbb{R}^{r\times k}$ ($r \ll \min(d, k)$) for each system module $\psi \in \{E_{plan}, E_{rag}, E_{logic}, E_{expr}\}$.

As illustrated in Fig. 3, during a specific module ψ's execution, the active weights θ_{active} dynamically incorporate the incremental weights ΔW_ψ:

$$\theta_{active} = W_0 + \Delta W_\psi = W_0 + \frac{\alpha}{r}(A_\psi B_\psi). \quad (3)$$

where α is a scaling coefficient. Fig 3 depicts an example of time-division multiplexing timeline: as the DAG execution order advances from the Planner ($\psi = E_{plan}$) to v_3 ($\psi = E_{expr}$) and onward, the system atomically unloads the previous LoRA adapter and loads the next. So, at any single time step, the GPU memory contains only the frozen backbone W_0 plus one active adapter slice.

Let $\mathcal{M}(\cdot)$ denote the GPU memory footprint function. The system's peak GPU memory is strictly bounded by Eq. (4).

$$\mathcal{M}_{peak} = \mathcal{M}(W_0) + \max_{\psi}\{\mathcal{M}(A_\psi) + \mathcal{M}(B_\psi)\}. \quad (4)$$

Since r is minimal, $\mathcal{M}(A_\psi)+\mathcal{M}(B_\psi)$ is negligible. This mathematically guarantees an $\mathcal{O}(1)$ redundant GPU memory complexity regardless of the scaling of both the expert pool and the planning policies.

3.3 Exception Handling and Reconstruction Mechanism

To endow the system with topological plasticity, we introduce a continuous monitoring mechanism that detects intermediate anomalies and dynamically restructures the reasoning graph to self-heal.

We define the global uncertainty $\mathcal{U}_t$ across committed nodes $\mathcal{V}_{exec}^{(t)}$ to measure the aggregate execution instability and the cumulative risk of reasoning divergence, as shown in Eq. (5).

$$\mathcal{U}_t = 1 - \frac{1}{|\mathcal{V}_{exec}^{(t)}|} \sum_{v_j \in \mathcal{V}_{exec}^{(t)}} c_j. \quad (5)$$

To prevent error propagation, we define the suspension indicator function $\mathbb{I}_{suspend}^{(t)}$ to govern when the system trigger reconstruction. Once $\mathbb{I}_{suspend}^{(t)} = 1$, the Evaluator halts downstream scheduling and initiates the reconfiguration process, shown in Eq. (6).

$$\mathbb{I}_{suspend}^{(t)} = \begin{cases} 1, & \exists v_j \in \mathcal{V}_{exec}^{(t)},\ \xi_j = 1 \\ 1, & \exists v_j \in \mathcal{V}_{exec}^{(t)},\ c_j < \tau_c \\ 1, & \mathcal{U}_t \geq \tau_u \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

where τ_u defines the global uncertainty tolerance and τ_c denotes the minimum single-node confidence threshold.

As illustrated in Fig. 1 (b), the topological transformation operator $\mathcal{T}$ yields $\mathcal{G}^{(t+1)} = \mathcal{T}(\mathcal{G}^{(t)}, \mathcal{F}_{err})$ via two distinct execution paths:

① **Fine-grained Patching Operator**: Repairs localized deficiencies triggered by irrecoverable node errors or confidence floor breaches. As depicted in Fig. 1 (b), upon detecting a lethal failure ($\xi_i = 1$) or a single-node confidence floor violation ($c_j < \tau_c$) at intermediate node v_i, the Planner dynamically instantiates an auxiliary patch node $v_{i,patch}$ based on the anomalous node's local context, thereby suturing localized breakpoints.

② **Subgraph Reconstruction Operator**: Rectifies macroscopic deviations. If a fine-grained patch fails, or the global uncertainty exceeds the tolerance threshold ($\mathcal{U}_t \geq \tau_u$), the system truncates all downstream branches from the failed node and replaces them with a freshly generated subgraph $\mathcal{G}_{sub}$.

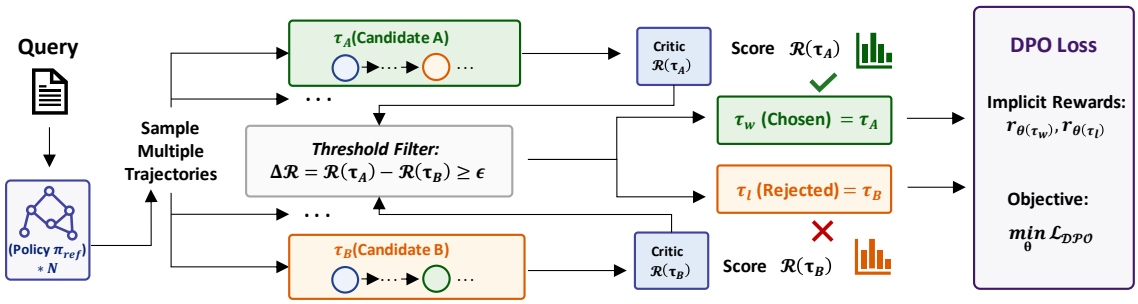

Figure 5: **Preference Pair Construction & DPO Optimization:** Candidate trajectories are sampled and ranked by the Critic to construct preference pairs, which are then used to optimize the central Planner via DPO.

This discards the corrupted execution state rather than allowing error propagation.

To prevent infinite recursive reconstruction, we impose an operational budget constraint. Let η be the cumulative reconstruction count and Ω_{max} the permissible threshold, as shown in Eq. (7).

$$\eta = \sum_{k=1}^{t} \mathbb{I}_{suspend}^{(k)} \leq \Omega_{max}. \tag{7}$$

Once η reaches Ω_{max}, the system terminates dynamic reconstruction and invokes a fallback mechanism, guaranteeing computational convergence in worst-case scenarios.

3.4 Critic Guided Central Model Training

3.4.1 Critic Establishment and Evaluation Dimensions

To train the Planner, we introduce a Topology Critic Crit$(\cdot)$ to map generated trajectories into learnable reward signals. As shown in Fig. 4, Crit$(\cdot)$ evaluates discrete trajectories τ using a composite reward function $\mathcal{R}(\tau)$ across three dimensions:

Task Completion Φ_{task}: Evaluated by a Teacher Model. For deterministic tasks, it strictly weights ground-truth matching. For open-ended scenarios, it assesses the ratio of valid propositions in draft $\mathcal{D}$ supported by evidence $\mathcal{K}$, heavily penalizing hallucinations $\mathcal{U}$.

Action Legality $\mathbb{I}_{leg}$: Acts as a strict gating constraint. Cyclic dependencies or mismatched expert assignments forcibly nullify the trajectory score.

System Overhead S_{cost}: Imposes a sub-linear penalty on total node count $|\mathcal{V}_\tau|$ and reconstruction occurrences η to suppress redundant paths.

The composite reward is formally defined as Eq. (8), where λ scales the overhead penalty and γ balances the reconstruction contribution weight. This equation utilizes a one-vote veto gating mechanism and a logarithmic penalty to permit reasonable expansion while restricting redundancy.

$$\begin{aligned}\mathcal{R}(\tau) = &\left(\prod_{v_i \in \tau} \mathbb{I}_{leg}(v_i)\right) \cdot \Phi_{task}(\tau) \\ &- \lambda \cdot \log\left(1 + |\mathcal{V}_\tau| + \gamma\eta\right).\end{aligned} \tag{8}$$

3.4.2 Critic-Guided Direct Preference Optimization

We employ Direct Preference Optimization (DPO) to fine-tune the low-rank adapter matrices of the central Planner π_θ, bypassing unstable reward model training.

As illustrated in Fig. 5, given a query x, we sample candidate trajectories using a reference policy π_{ref} and score them via $\mathcal{R}(\tau)$. Fig. 5 details the pipeline for constructing preference pairs. For a given query x, the reference policy π_{ref} generates several candidate trajectories, which the Crit$(\cdot)$ then scores. For each valid pair, the higher-scoring trajectory is designated as Chosen (τ_w), and the lower-scoring one as Rejected (τ_l). To filter evaluation noise, a sample pair is incorporated into the preference dataset $\mathbb{D} = \{(x^{(i)}, \tau_w^{(i)}, \tau_l^{(i)})\}_{i=1}^N$ only if the reward differential satisfies a margin threshold $\Delta\mathcal{R} \geq \epsilon$. Based on the Bradley-Terry model, the implicit reward function is defined as Eq. (9),

$$r_\theta(\tau|x) = \beta \log \frac{\pi_\theta(\tau|x)}{\pi_{ref}(\tau|x)}, \tag{9}$$

where β controls the KL divergence penalty. The final DPO loss minimizes the negative log-likelihood like Eq. (10) and $\sigma(\cdot)$ means the logistic function.

$$\begin{aligned}&\mathcal{L}_{DPO}(\pi_\theta; \pi_{ref}) = \\ &\quad - \mathbb{E}_{(x,\tau_w,\tau_l)\sim\mathbb{D}}\Big[\log \sigma\big(r_\theta(\tau_w|x) - r_\theta(\tau_l|x)\big)\Big].\end{aligned} \tag{10}$$

Table 1: **Main Experimental Results on Heterogeneous Cognitive Tasks.** We report task accuracy (Acc, %), token consumption alongside estimated compute (#Tok / TFLOPs), and average end-to-end latency (Lat., seconds). **Bold** indicates the best performance among models in the 8B parameter class, and underline denotes the second best (the 72B large model is excluded from this ranking).

Method / Architecture	StrategyQA (Open-domain)			MATH (Deduction)			FinQA (Heterogeneous)			GPU DRAM
	Acc (↑)	#Tok / TFLOPs (↓)	Lat. (↓)	Acc (↑)	#Tok / TFLOPs (↓)	Lat. (↓)	Acc (↑)	#Tok / TFLOPs (↓)	Lat. (↓)	Usage (GB)
Static Monolithic Baselines (8B Class)										
Standard Prompting	65.2	**210 / 3.4**	**2.5**	45.5	**350 / 5.6**	**4.2**	55.0	**420 / 6.7**	**4.8**	16.5
Standard CoT	78.4	650 / 10.4	8.0	65.4	1,020 / 16.3	11.8	70.5	1,260 / 20.2	14.9	16.5
Standard ToT	85.8	3,400 / 54.4	39.5	80.0	5,430 / 86.9	67.6	80.7	6,930 / 110.9	76.3	16.5
Self-Consistency (k=5)	83.3	1,700 / 27.2	19.8	78.2	2,730 / 43.7	34.4	76.4	3,490 / 55.8	38.4	16.5
Unconstrained Dynamic Agents (8B Class)										
ReAct	84.5	2,480 / 39.7	30.5	77.6	4,310 / 69.0	50.0	76.2	5,080 / 81.3	59.5	16.5
Reflexion	86.2	3,890 / 62.2	44.5	80.5	6,160 / 98.6	76.4	78.8	7,690 / 123.0	86.9	16.5
Multi-Agent (3 × 8B)	84.0	1,520 / 24.3	17.5	79.4	2,470 / 39.5	29.5	77.6	2,780 / 44.5	34.3	> 49.5
High-Resource Large Model Reference										
Qwen-2-72B-Instruct	*92.5*	*800 / 115.2*	*10.5*	*89.2*	*1,440 / 207.4*	*16.4*	*86.1*	*1,640 / 236.2*	*20.6*	*> 145.0*
DynaGraph (Ours, 8B)	**87.6**	1,220 / 19.5	15.3	**82.7**	2,170 / 34.7	24.4	**82.5**	2,480 / 39.7	30.2	16.6 ($\mathcal{O}(1)$)

4 Experiments

4.1 Experimental Setup

We evaluate *DynaGraph* on three datasets: (1) StrategyQA (Geva et al., 2021) for open-domain multi-hop fact retrieval; (2) MATH (Hendrycks et al., 2021) for long-horizon logical deduction and topological resilience; and (3) FinQA (Chen et al., 2021) for cross-modal multi-expert orchestration. Using DeepSeek-8B as the shared base model with LoRA-parameterized experts, all experiments run on a single RTX 5090 (32GB) GPU to validate the $\mathcal{O}(1)$ spatial complexity.

We compare against representative 8B architectures. Static Monolithic Baselines include Standard Prompting (zero-shot), CoT (Wei et al., 2022b) (intermediate reasoning), Self-Consistency ($k = 5$) (Wang et al., 2022) (majority voting over CoT paths), and ToT (Yao et al., 2023) (tree search over reasoning branches). Unconstrained Dynamic Agents comprise ReAct (Yao et al., 2022) (heuristic trial-and-error), Reflexion (Shinn et al., 2023) (ReAct with textual error-reflection), and Multi-Agent ($3 \times 8B$) (collaboration lacking dynamic memory optimization). Finally, Qwen-2-72B-Instruct (Yang et al., 2024) serves as the 72B High-Resource Reference to evaluate how closely our 8B system approximates massive-parameter capabilities.

Hyperparameters & Metrics. All 8B models share the same DeepSeek-8B bf16 checkpoint (temperature=0.7, top-p=0.9) on a single RTX 5090 GPU. Experts utilize LoRA (rank=8, $\alpha = 16$, dropout=0.05) with thresholds $\tau_c = 0.35$ and $\tau_u = 0.45$. GPT-5 serves strictly as the offline DPO training teacher. #Tok sums all prompt and completion tokens (including failed/reconstructed trajectories and planning calls). TFLOPs are calculated as $2 \times$ parameters $\times$ tokens. Latency reflects single-sample, unbatched wall-clock time, factoring in a ∼0.8 s adapter hot-loading overhead.

4.2 Main Results: Task Efficacy and Economy

4.2.1 Efficacy and High-Resource Approximation

We evaluate *DynaGraph* across cognitive efficacy and resource constraints, utilizing Tab. 1 and Fig. 6 (a). As shown in Tab. 1, *DynaGraph* achieves 87.6%, 82.7%, and 82.5% on StrategyQA, MATH, and FinQA, respectively, outperforming all 8B baselines. Compared to Self-Consistency ($k = 5$), it improves average accuracy by approximately 5.0%. Notably, on MATH, the 8B *DynaGraph* (82.7%) closely approaches the 72B reference (89.2%), demonstrating that dynamic topological intervention can partially mitigate inherent capability limitations imposed by parameter scale.

4.2.2 Compute and Time Economy

Unconstrained dynamic agents and complex tree-search methods suffer severe context redundancy. For instance, Reflexion consumes 7,690 tokens and 86.9 s on FinQA due to lengthy textual self-reflections, while Standard ToT requires up to 6,930 tokens and 76.3 s. *DynaGraph* replaces linguistic reflection with physical-level graph pruning. While achieving higher accuracy than Reflexion (82.5% vs. 78.8% on FinQA), it reduces token consumption by 64.8%–68.6% and latency by 65.2%–68.1% across benchmarks (e.g., FinQA latency drops from 86.9 s to 30.2 s). Furthermore,

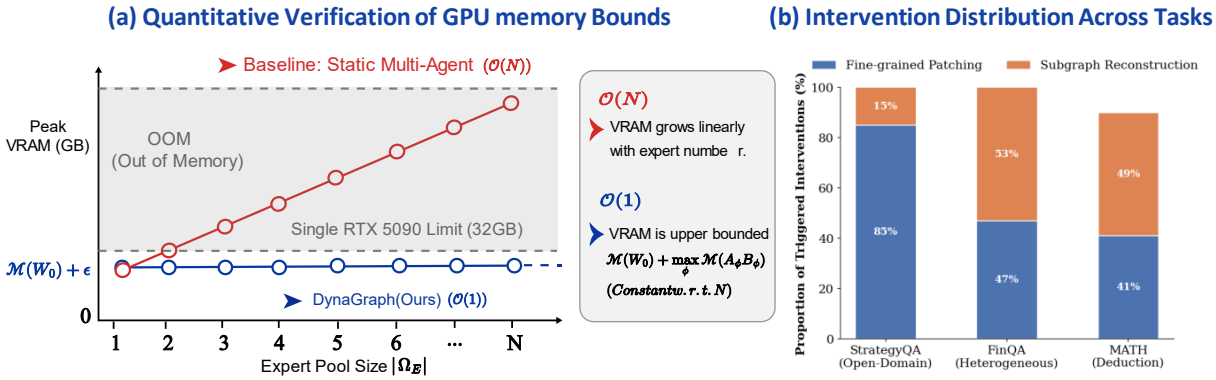

Figure 6: **System efficiency and adaptive routing of *DynaGraph*.** (a) Quantitative Verification of GPU Memory Bounds: PEFT multiplexing yields a constant $\mathcal{O}(1)$ peak GPU memory footprint. (b) Intervention Distribution Across Tasks: The Controller deploys task-adaptive corrections: StrategyQA predominantly uses *Fine-grained patching* (85%), while MATH triggers *Subgraph Reconstruction* (49%) to halt cascading errors.

the TFLOPs metric reveals absolute cross-scale efficiency: despite processing comparable or more tokens than the 72B model, 8B *DynaGraph* incurs only a fraction of its compute .

4.2.3 Space Economy

Multi-expert deployment is typically hindered by GPU memory bottlenecks. The static Multi-Agent ($3 \times 8B$) baseline requires > 49.5 GB, posing severe OOM risks. In contrast, *DynaGraph* maintains a stable peak GPU memory of $\approx$16.6 GB (Fig. 6 (a)) regardless of expert pool scale, empirically validating the $\mathcal{O}(1)$ bound derived in Sec. 3 and confirming consumer-grade deployability.

4.3 Ablation Study on Adaptive Interventions

4.3.1 Dynamic Routing Distribution

We conduct ablation studies to investigate the dynamic routing behavior and contextual decision-making capacity of the system. We first evaluate whether anomalies are routed adaptively rather than relying on static heuristics. Fig. 6 (b) shows the proportional distribution of fine-grained patching and subgraph reconstruction across tasks. The results show a clear alignment between intervention strategy and task-specific error tolerance. On StrategyQA, 85% of anomalies are resolved via lightweight ine-grained Patching, leveraging E_{rag} to append missing facts without disrupting execution. Conversely, on MATH, 59% of anomalies trigger Subgraph Reconstruction to prune erroneous branches and halt cascading errors. This confirms that *DynaGraph* evaluates error costs rather than relying on hard-coded thresholds.

Table 2: **Ablation Results of Dynamic Intervention Mechanisms.** We report task accuracy (Acc, %) and average end-to-end latency (Lat., seconds).

Configuration	StrategyQA		MATH	
	Acc (↑)	Lat. (↓)	Acc (↑)	Lat. (↓)
w/o Fine-grained Patching	86.9	29.4	81.2	35.4
w/o Subgraph Reconstruction	78.4	13.2	41.5	16.9
Full *DynaGraph (Ours)*	**87.6**	**15.3**	**82.7**	**24.4**

4.3.2 Mechanism Ablation

To quantify each mechanism's contribution, we isolate them in control experiments. Tab. 2 confirms their complementary efficiency-precision boundaries. Without fine-grained patching, StrategyQA latency escalates from 15.3 s to 29.4 s (+92.2%) as localized gaps force global re-computation, though accuracy remains intact. This highlights patching's pivotal role in execution economy. Conversely, removing subgraph reconstruction collapses MATH accuracy from 82.7% to 41.5%, as early hallucinations corrupt the execution state beyond local repair. This underscores structural reconstruction's indispensability for topological resilience and deductive coherence.

5 Conclusion

We propose *DynaGraph*, a lightweight framework that overcomes the fragility and memory bottlenecks of multi-agent systems. By multiplexing PEFT adapters and employing dynamic topological reconfiguration, it enables complex inference on a single consumer GPU. Empirically, our 8B model rivals 72B monoliths while drastically reducing latency and computational overhead.

Limitations

While *DynaGraph* enables efficient self-healing, its time-division multiplexing inherently enforces sequential expert execution. This bounds VRAM to prevent out-of-memory errors but precludes simultaneous execution on parallelizable sub-tasks, introducing switching overhead. Furthermore, topological reconfiguration relies heavily on local expert self-calibration and predefined confidence thresholds. Empirically, evaluations are limited to specific English-centric datasets; generalizing to specialized domains (e.g., biomedical or legal) or morphologically rich languages requires further exploration. Finally, investigating *DynaGraph*'s scaling laws on massive models (e.g., 70B+) to determine if topological interventions yield diminishing returns remains a critical avenue for future work.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

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

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. Legal-bert: The muppets straight out of law school. *Preprint*, arXiv:2010.02559.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan R Routledge, et al. 2021. Finqa: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *Preprint*, arXiv:2401.06066.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Dan Hendrycks, Collin Burns, Saurav Kadavath, et al. 2021. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS*.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven K. S. Yau, Zijian Lin, et al. 2024. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Hanna, Florian Bensch, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Zongqian Li, Yixuan Su, and Nigel Collier. 2025. A survey on prompt tuning. *Preprint*, arXiv:2507.06085.

Haolang Lu, Yilian Liu, Jingxin Xu, Guoshun Nan, Yuanlong Yu, Zhican Chen, and Kun Wang. 2026. Auditing meta-cognitive hallucinations in reasoning large language models. *Advances in Neural Information Processing Systems*, 38:162500–162543.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, et al. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Sheng Shen, Le Hou, Yanqi Zhou, Nan Du, Shayne Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen, et al. 2024. Mixture-of-experts meets instruction tuning: A winning combination for large language models. In *International Conference on Learning Representations*, volume 2024, pages 18858–18884.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 8634–8652.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *Preprint*, arXiv:1811.00937.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, et al. 2023. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Preprint*, arXiv:2305.04388.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv 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. 2022a. 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. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance. *Preprint*, arXiv:2303.17564.

An Yang, Yang Baosong, Hui Binyuan, Zheng Bo, Yu Bowen, Gao Chang, Li Chengpeng, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, et al. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. 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. 2022. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations (ICLR)*.

Zihao Zheng, Xiuping Cui, Size Zheng, Maoliang Li, Jiayu Chen, Yun Liang, and Xiang Chen. 2025. Dynamo: Runtime switchable quantization for moe with cross-dataset adaptation. *arXiv preprint arXiv:2503.21135*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. *Preprint*, arXiv:2205.10625.

Kunlun Zhu, Zijia Liu, et al. 2025. Where llm agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*.