EFFICIENT TREE-STRUCTURED DEEP RESEARCH WITH ADAPTIVE RESOURCE ALLOCATION

Lunyiu Nie[1]*, **Nedim Lipka**[2], **Ryan A. Rossi**[2], **Swarat Chaudhuri**[1]
[1]The University of Texas at Austin [2]Adobe Research
lynie@utexas.edu

ABSTRACT

Deep research agents, which synthesize information across diverse sources, are significantly constrained by the sequential nature of reasoning. This bottleneck results in high latency, poor runtime adaptability, and inefficient resource allocation, making today's deep research systems impractical for interactive applications. To overcome this, we introduce **ParallelResearch**, a novel framework for efficient deep research that transforms sequential processing into parallel, runtime orchestration by dynamically decomposing complex queries into tree-structured sub-tasks. Our core contributions are threefold: **(1)** an **adaptive planner** that dynamically allocates computational resources based on query complexity; **(2)** a **runtime orchestration layer** that prunes redundant paths to reallocate resources and enables speculative execution; and **(3)** a **fully-asynchronous execution infrastructure** that enables concurrency across both research breadth and depth. Experiments on two benchmarks show up to 5× speedups with comparable final report quality, and consistent quality improvements with the same time budgets.

1 INTRODUCTION

Deep research systems, which can generate long-form responses by synthesizing information from diverse resources given an open-ended query, have become increasingly powerful with the advances in LLM-based agents. While the state-of-the-art systems can create high-quality reports for deep research tasks, including literature review (Haman & Školník, 2025) and policy analysis (Gambrell, 2025), their efficiency is still limited: such systems often take tens of minutes to respond. This latency can break users' cognitive flow (Iqbal & Horvitz, 2007), incur high context-switching costs (Mark et al., 2008), and degrade overall experience.

Much of this inefficiency stems from poor runtime orchestration. Existing systems typically rely on sequential planning and reasoning (Xu & Peng, 2025), leading to unnecessary latency when opportunities for parallel branching and speculative exploration are under-utilized. These systems typically plan ahead of research. However, the value of subqueries or deeper investigation often becomes clear only after partial research progress, but current systems rarely prune low-value paths or reallocate resources at runtime.

To address these challenges, we propose **ParallelResearch**, a framework designed for efficient deep research that integrates adaptive planning and runtime orchestration on a fully-asynchronous execution infrastructure. ParallelResearch treats deep research as a dynamic, tree-structured traversal, where a complex query is decomposed into concurrent subqueries that dynamically populate the research tree. The objective is to maximize response quality under a time budget by adjusting the tree structure and reallocating resources across promising research paths.

At the core of ParallelResearch is an adaptive planner that weighs the expected information gains *before* and *after* each research step, expanding breadth and deepening paths only when the information gain justifies the overheads. This allows ParallelResearch to flexibly allocate resources depending on nature, scope, and complexity of each query.

*Work done during internship at Adobe Research.

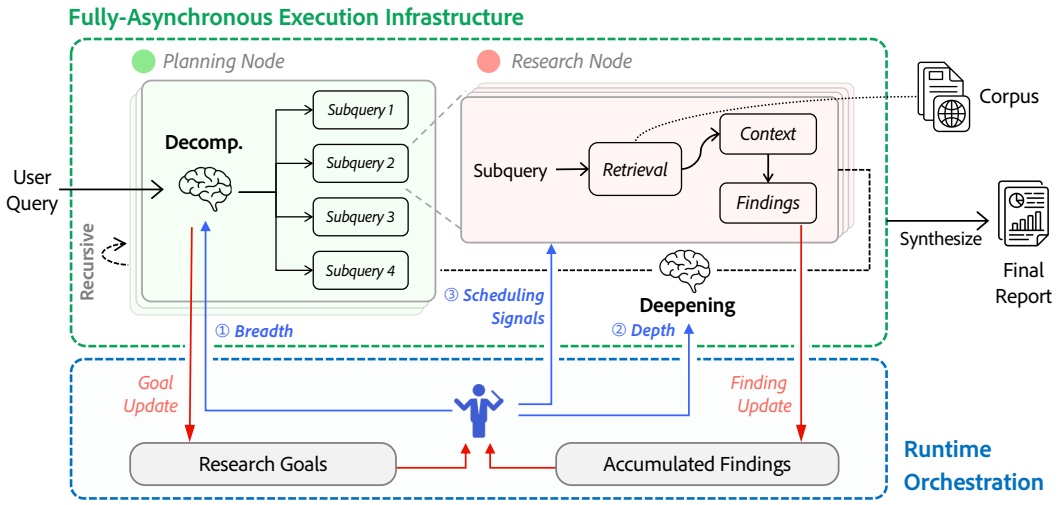


Figure 1: Overview of ParallelResearch: the Planning Nodes adaptively decompose queries into parallel subqueries executed by Research Nodes for findings, which may recursively trigger deeper planning. The **adaptive planner** expands (1) *breadth* to explore prior-research and regulates (2) *depth* to pursue promising paths post-research. The **runtime orchestration layer** monitors progress and reallocates resources through (3) *scheduling signals* mid-research. A **fully-asynchronous execution infrastructure** enables flexible concurrency across both breadth and depth.

However, planning alone is insufficient. Since research is inherently iterative and non-linear, newly emerged evidence *mid-research* may reshape priorities. ParallelResearch incorporates a runtime orchestration layer that monitors ongoing research findings, evaluates them against goal satisfaction and quality rubrics, and makes live adjustments. This mechanism allows the system to terminate low-value or redundant branches early and reallocate computational resources toward more promising paths. More importantly, it allows *speculative execution* by launching child tasks before parent-level planning decisions are finalized, thereby reducing latency and accelerating throughput.

This tight feedback loop between planning and research execution produces a highly dynamic research tree that evolves at runtime. To handle this, ParallelResearch employs a fully-asynchronous execution infrastructure to schedule all the planning, research, and orchestration tasks in a unified pool with thread-safe state management. This enables concurrent exploration of multiple research paths and allows non-blocking orchestration to adapt the tree structure.

To evaluate the effectiveness of ParallelResearch, we conduct experiments on two recent deep research benchmarks, DeepResearchGym (Coelho et al., 2025) and DeepResearch Bench (Alzubi et al., 2025). Compared to the baseline, ParallelResearch can consistently improve research throughput, producing research reports of the same quality with up to 5$\times$ speed-ups, or even higher quality within the same time.

The key contributions of this work are:

- A formal formalization of deep research tasks as a tree-structured optimization problem.
- An adaptive research planner for dynamic, context-aware decisions on task branching and deepening.
- A runtime orchestration layer for task monitoring, speculative execution, and resource reallocation.
- A fully asynchronous execution infrastructure enabling concurrent planning, research, and orchestration across breadth and depth of the evolving tree.
- A comprehensive empirical evaluation on three model families across two benchmarks, demonstrating ParallelResearch's significant improvements in terms of research throughput, quality, and efficiency.

2 Related Works

2.1 Deep Research Agents

Building on earlier tool-use frameworks like WebGPT (Nakano et al., 2021) and ReAct (Yao et al., 2023b), recent deep research agents – including GPT-Researcher (Elovic, 2023), Open Deep Search (Alzubi et al., 2025), and LangChain's Open Deep Research (Langchain, 2025) – decompose complex queries into tool-augmented subtasks. To standardize evaluation, emerging benchmarks like DeepResearchGym (Coelho et al., 2025) and DeepResearch Bench (Du et al., 2025) introduce LLM-as-a-judge protocols tailored for complex research questions.

Despite these advances, current systems typically rely on fixed, pre-specified parameters for controlling the research structure. Their orchestration strategies are dominated by sequential execution or coarse-grained parallelism (Xu & Peng, 2025), which limits adaptability. As a result, when information quality shifts during execution, these systems either waste compute or incur unnecessary latency. ParallelResearch addresses these limitations by introducing a runtime orchestration layer that couples fully-asynchronous execution across both depth and breadth. Unlike prior static approaches, it adaptively expands or prunes subqueries in real time based on intermediate evidence, enabling more efficient and responsive deep research.

2.2 Parallel and Speculative Reasoning

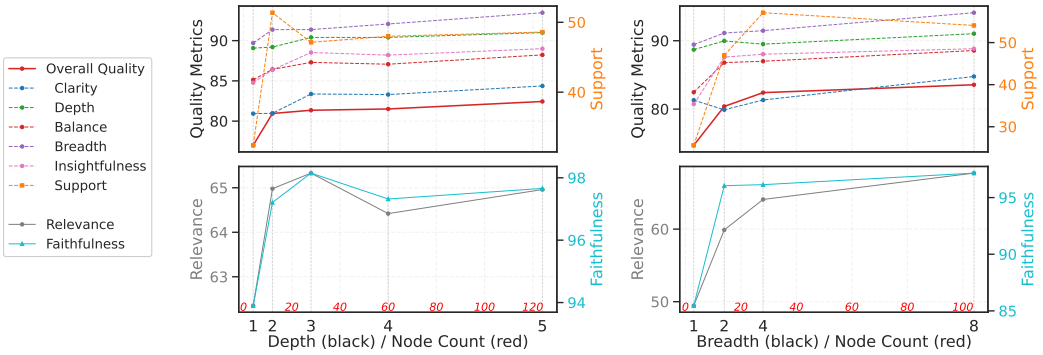
Token- and action-level acceleration methods such as speculative decoding (Leviathan et al., 2023; Miao et al., 2023; Cai et al., 2024) and speculative reasoning for fast inference (Pan et al., 2025; Yang et al., 2025) reduce latency via draft-and-verify or multi-token prediction. At the reasoning level, Dynamic Parallel Tree Search (Ding et al., 2025) accelerates Tree-of-Thoughts by expanding and pruning nodes in parallel, while ParaThinker (Wen et al., 2025) and Parallel-R1 (Zheng et al., 2025) instill native parallel reasoning. These improve efficiency and accuracy but still rely on static branching. Inspired by these works, ParallelResearch advances parallelism to the workflow level: it not only reallocates compute and prunes branches dynamically, but also supports speculative execution—allowing branches to expand without delay and later discarding them if evidence shows they are unnecessary.

2.3 Agent Workflow Orchestration

Recent work compiles high-level goals into executable agent graphs via MCTS-guided code search (Zhang et al., 2024), evolutionary populations of heterogeneous workflows (Niu et al., 2025), and modular activity-on-vertex graphs (Zhang et al., 2025a). These systems mainly optimize *offline* and then execute largely fixed graphs, and their runtime control over partially executed graphs remains limited. Production frameworks such as AutoGen (Wu et al., 2023), LangGraph (LangChain, 2024), DSPy (Khattab et al., 2024), and OpenAI Swarm (OpenAI, 2024) provide valuable abstractions for multi-agent pipeline control.

More recently, hierarchical frameworks like OWL (Hu et al., 2025) deploy domain-agnostic planners with specialized workers using task decomposition and reactive adaptive, while AgentOrchestra (Zhang et al., 2025c) unifies tools, environments, and agents via TEA Protocol for agent-level coordination. Cognitive Kernel-Pro (Fang et al., 2025) focuses on cognitive architecture and memory for knowledge integration, and OAgents (Zhu et al., 2025) provides a modular infrastructure for agent design and evaluation. For runtime adaptation, Co-Sight (Zhang et al., 2025b) embeds replanning rules in its agentic planner, smolagents (Roucher et al., 2025) employs interval-based replanning up to fixed step limits, and dynamic workflow systems (Gao et al., 2025; Nie et al., 2025) enable intelligent agent resource allocation.

However, these systems usually operate at agent-level granularity without research-specific optimizations. ParallelResearch uniquely addresses this with the tree-structured formulation and node-level orchestration, enabling finer-grained resource reallocation capabilities like parallel branching, speculative execution, and early termination of redundant paths. This allows ParallelResearch to dynamically adapt the research workflow in response to the evolving information with maximized efficiency.



(a) Depth trade-off (b) Breadth trade-off

Figure 2: **Trade-offs between deep research tree structure and response quality.** Left figure (a) varies *depth* (breadth fixed at 4) and right figure (b) varies *breadth* (depth fixed at 3); in each, the *top* plot shows *Quality* metrics with sub-metric *Support* on the right y-axis, while the *bottom* plot shows *Relevance* (left) and *Faithfulness* (right). The red labels along the x-axis give total node counts as a proxy for computational cost. Early increases raise quality, but gains saturate as cost escalates.

3 Background

3.1 Problem Formulation

Deep research tasks involve tackling complex, open-ended queries by iterative reasoning, gathering diverse information, and synthesizing knowledge into comprehensive responses.

We formalize such a task as follows: a user query $q \in \mathcal{Q}$, where $\mathcal{Q}$ is the space of natural language queries. The goal is to produce a response $r \in \mathcal{R}$ by integrating knowledge from retrieved context $C = \{c_1, c_2, \ldots, c_n\}$ sourced from a corpus D (e.g., web searches or local documents). During this process, research findings $F = \{f_1, f_2, \ldots, f_m\}$, comprising reasoning traces and summarized insights, are iteratively derived from the context.

Consequently, the response is generated as

$$r = \sigma(q, C, F), \tag{1}$$

where $\sigma : \mathcal{Q} \times 2^C \times 2^F \to \mathcal{R}$ is a synthesis function that aggregates and processes the inputs to maximize quality metrics like factual accuracy, comprehensiveness, and relevance. Here, 2^C and 2^F denote the power sets of C and F, representing all possible subsets of contexts and findings. In practice, these are subsets collected during the research process, and σ is typically implemented with an LLM.

To solve such tasks scalably, a deep research framework must balance thorough exploration with computational efficiency. Traditional sequential pipelines like linear retrieval-augmented generation (RAG) often struggle with complex queries, incurring high costs due to repetitive traversals or premature convergence to suboptimal research direction. Given the hierarchical and multi-faceted nature of deep research, it is natural to model the process as a tree structure, like in Tree of Thoughts (Yao et al., 2023a).

Formally, we model the process as a directed tree $\mathcal{T} = (N^P \cup N^R, E)$ with disjoint node sets of *planning nodes* N^P and *research nodes* N^R.

A planning node n^P decomposes a query q^n into a finite set of subqueries:

$$n^P(q^n) \to \{q_1^n, \ldots, q_{b_n}^n\}, \quad q_j^n \in \mathcal{Q}. \tag{2}$$

Here, q^n can be either the initial query or a previously decomposed subquery. The tree root is therefore the planning node n_0^P that receives the initial query q. The number of decomposed subqueries $b_n = |n^P(q^n)|$ is defined as the *breadth* at this research level.

Each subquery q_j^n further instantiates a research node $n^R(q_j^n)$, which retrieves information from the corpus D and performs reasoning:

$$n^R(q_j^n) \to (C_{q_j^n}, F_{q_j^n}), \tag{3}$$

producing a set of local context $C_{q_j^n} \subseteq C$ and research findings $F_{q_j^n} \subseteq F$.

Optionally, a research node can be deepened by spawning a child planning node that further decomposes q_j^n, after which the alternating process of planning and research continues recursively. The *depth* d of the tree is defined as the number of research-node layers along the longest root-to-leaf path.

The final response $r_{\mathcal{T}}$ can then be synthesized as

$$r_{\mathcal{T}} = \sigma\left(q, \bigcup_{n_i \in N^R} C_i, \bigcup_{n_i \in N^R} F_i\right), \tag{4}$$

by aggregating each research node's local contexts and findings across the tree. The response quality can be subsequently measured by an evaluation function $E(r)$.

However, fixing the depth and breadth of the tree can lead to suboptimal performance: shallow trees might insufficiently explore complex topics, while overly deep or wide trees can waste substantial resources on simple tasks. Therefore, the core challenge is to orchestrate the tree structure at runtime to maximize the response quality within a time budget t_{max}.

Formally, we aim to solve the optimization problem:

$$\arg\max_{\mathcal{T}} E(r_{\mathcal{T}}) \quad \text{s.t.} \quad t(\mathcal{T}) \leq t_{\text{max}}, \tag{5}$$

where $t(\mathcal{T})$ denotes the end-to-end latency of the research tree including all nodes and edges.

3.2 Motivating Experiments

Following the above formulation, we investigate how the tree structure can affect response quality in deep research tasks. We evaluate an existing deep research framework, GPT-Researcher (Elovic, 2023), on a set of 100 complex queries randomly sampled from DeepResearchGym (Coelho et al., 2025). We varied the research tree's maximum depth and breadth hyperparameters and measured performance across multiple metrics.

Depth In Figure 2(a), increasing depth from 1 to 2 yields the largest gain, with overall quality score rising sharply from 77.00 to 80.95. Beyond depth 3, the curves flatten. Extra depth produces only marginal quality improvements while node number grows exponentially. Notably, *Relevance* and *Faithfulness* peak at depth 3 and then decline, as deeper searches bring in peripheral sources and redundant materials, diluting core evidence and complicating the write-up compression.

Breadth A similar pattern can be observed when varying the breadth. In Figure 2(b), widening the tree from breadth 1 to 2 delivers a substantial quality gain with a moderate increase in nodes. Quality continues to improve up to breadth 4, after which the gains taper off.

To summarize, initial increases in depth or breadth are valuable, but returns diminish as node counts escalate. This highlights that a one-size-fits-all approach is inefficient. Adaptive planning is essential to tailor depth and breadth to each query's complexity, optimizing the quality-cost tradeoff.

4 ParallelResearch

ParallelResearch consists of three core components: (1) an **Adaptive Research Planner**, (2) a **Runtime Orchestration Layer**, and (3) a **Fully-Asynchronous Execution Infrastructure**. Together, these components dynamically expand and prune the research tree $\mathcal{T}$ in real time and execute subtasks concurrently.

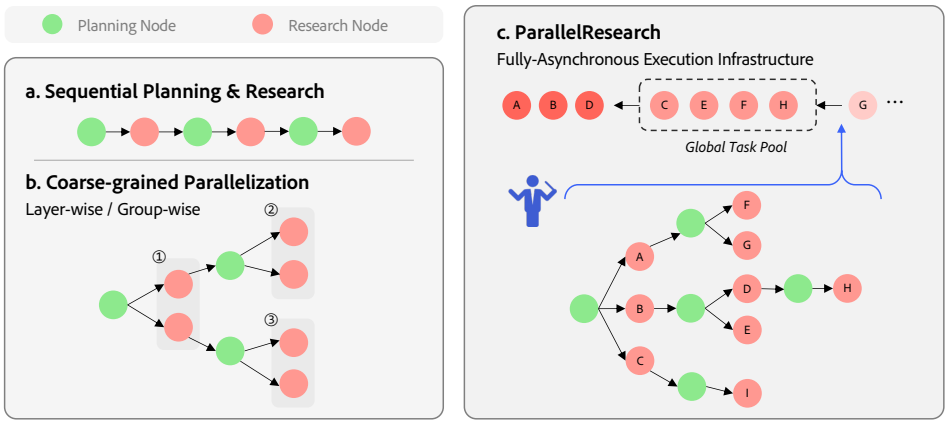


Figure 3: Sequential processing and coarse-grained parallelization at layer or group level force nodes to wait for slow dependencies and introduce unnecessary latency. ParallelResearch enables fully asynchronous execution by submitting task nodes to a global pool; child nodes (e.g., `D`, `E`, `F`) can start as soon as their parents (`A`, `B`) finish, without waiting for unrelated nodes such as `C`.

4.1 Adaptive Research Planning

To efficiently navigate vast information spaces, it is essential to adapt the breadth and depth of research to the scope, nature, and complexity of each query. For example, broad queries like *"What is the impact of climate change?"* can be decomposed into multiple subqueries that address distinct aspects. In contrast, more specific questions like *"What's the process for developing film in a darkroom?"* require less exploration but demand greater focus and precision.

Therefore, we propose an adaptive research planner that decomposes the query q^n into b_n subqueries at each planning node $n^P \in N^P$, conditioned on the current query q^n and the accumulated findings F. Concretely, the planner applies a breadth policy π_b:

$$n^P(q^n) = \pi_b(q^n, F) = \big(b_n, \{q_1^n, \ldots, q_{b_n}^n\}\big), \tag{6}$$

where b_n denotes the branching factor (i.e., the number of subqueries) and $\{q_1^n, \ldots, q_{b_n}^n\}$ are the generated subqueries. The key decision is to choose b_n to balance the expected utility of broader exploration against the additional computation incurred by executing the spawned research nodes:

$$b_n = \arg\max_{b \in [1, b_{\max}]} \mathbb{E}\Big[\Delta U_{\mathrm{IG}}(b \mid q^n, F) \; - \; \lambda \sum_{j=1}^{b} \Delta t\big(n^R(q_j^n)\big)\Big], \tag{7}$$

where $U_{\mathrm{IG}}(b \mid q^n, F)$ estimates the expected *information gain* (Quinlan, 1986) from decomposing into b subqueries, $t\big(n^R(q_j^n)\big)$ denotes the expected latency of running the corresponding research node, and $\lambda > 0$ controls the tradeoff between response quality and computational overhead.

Once a research node finishes, a depth policy π_d decides whether to further deepen the current branch based on the local query q^i and findings F_i at each research node $n_i^R \in N^R$. Specifically, π_d compares the expected information gain from one additional research-node layer to the expected additional latency incurred, and deepens only when the gain-per-latency ratio exceeds a threshold:

$$\pi_d(q^i, F_i) = \mathbb{I}\left\{\frac{\mathbb{E}\big[\Delta U_{\mathrm{IG}}(q^i, F_i)\big]}{\mathbb{E}[\Delta t(q^i, F_i)] + \epsilon} > \tau\right\}, \tag{8}$$

where $\Delta U_{\mathrm{IG}}(q^i, F_i)$ denotes the expected *information gain* from deepening the current branch, $\Delta t(q^i, F_i)$ denotes the corresponding expected incremental latency, $\epsilon > 0$ is a small constant for numerical stability, and τ is a threshold that controls diminishing returns. The output is a binary decision whether to deepen the current research path.

In our work, π_b and π_d are implemented with LLMs for decision-making. However, the proposed formulation is policy-agnostic and can be also instantiated using hand-crafted heuristics or learned models, *e.g.*, via reinforcement learning.

Algorithm 1 Runtime Orchestration

Require: Hyperparameters Φ_{min}, Ψ_{min}
function ORCHESTRATOR(n_i^R, q^i, C_i, F_i)
  n_i^R.terminate $\leftarrow$ `False` ▷ Initialization
  Async Execute node n_i^R, update C_i, F_i and parent
  Async Plan child queries
  for each child query q^j **do**
    Async Speculatively execute n_j^R with C_j, F_j
    Async ORCHESTRATOR(n_j^R, q^j, C_j, F_j)
  end for
  ▷ Continuous monitor at each level
  while not n_i^R.terminate **do**
    Update C_i, F_i from n_i^R and descendants n_j^R
    Pass C_i, F_i to parent node
    Async Evaluate $(\phi_i, \psi_i) \leftarrow \pi_o(q^i, C_i, F_i)$
    if $\phi_i \geq \Phi_{\text{min}}$ and $\psi_i \geq \Psi_{\text{min}}$ **then**
      n_i^R.terminate $\leftarrow$ `True`
      Kill node n_i^R ▷ Early termination
      for each descendant n_j^R **do** ▷ Pruning
        Kill n_j^R and descendants recursively
      end for
    end if
    if n_i^R and all children finished/killed **then**
      n_i^R.terminate $\leftarrow$ `True`
    end if
  end while
end function

4.2 RUNTIME ORCHESTRATION

To address the dynamic and evolving nature of research, where priorities can shift as new evidence emerges, breadth planning conducted *prior-research* and depth planning performed *post-research* can be limiting. Furthermore, depth planning can also delay the exploitation of promising research paths until decisions are finalized.

To mitigate these, we introduce a runtime orchestration layer that dynamically manages the research tree $\mathcal{T}$ using *mid-research* signals for resource reallocation. Each research node $n^R(q^i)$ is continuously monitored by an orchestration policy π_o conditioned on the local query q^i, real-time context C_i, and accumulated findings F_i:

$$\pi_o(q^i, C_i, F_i) = (\phi_i, \psi_i), \tag{9}$$

where $\phi_i \in [0, 1]$ denotes a goal-satisfaction score and $\psi_i \in [0, 1]$ denotes a quality score. The orchestration layer uses these signals to determine whether to continue scheduling work on node i:

$$\text{CONTINUE}(i) = \mathbb{I}\{\phi_i \geq \Phi_{\text{min}} \ \wedge \ \psi_i \geq \Psi_{\text{min}}\}, \tag{10}$$

where Φ_{min} and Ψ_{min} are thresholds for goal satisfaction and quality, respectively. In our implementation, π_o is instantiated with an LLM.

More importantly, this mechanism enables *speculative execution*: child nodes can be spawned and deepen the tree without awaiting the parent's planning decision. Child nodes' findings update the parent's C_i and F_i, even after the parent's research completes, enabling recursive and adaptive task management. As shown in Algorithm 1, upon evaluation at each hierarchy, low-yield nodes and their descendants are terminated early once the research goal is satisfied, pruning the subtree dynamically.

4.3 FULLY-ASYNCHRONOUS EXECUTION INFRASTRUCTURE

To maximize efficiency in traversing the evolving research tree $\mathcal{T}$, ParallelResearch incorporates a fully-asynchronous execution infrastructure that enables concurrent processing across multiple axes:

Table 1: Overall evaluation results on DeepResearch Bench under flexible time budgets, judged by **`Gemini-2.5-flash`** and **`Gemini-2.5-pro`**. ParallelResearch (-AP, -RO) refers to the ablation without adaptive planning and runtime orchestration. Commercial deep research agents' results are directly from the DeepResearch Bench Leaderboard[1] with no latency statistics reported.

Method	Throughput		RACE					FACT	
	# Nodes	Latency	Overall	Comp.	Depth	Inst.	Read.	Cit. Acc.	Eff. Cit.
Grok Deeper Search	-	-	38.22	36.08	30.89	46.59	42.17	73.08	8.58
Perplexity Research	-	-	40.46	39.10	35.65	46.11	43.08	82.63	31.20
OpenAI Deep Research	-	-	46.45	46.46	43.73	49.39	47.22	75.01	39.79
Gemini-2.5-Pro Deep Research	-	-	49.71	49.51	49.45	50.12	50.00	78.30	165.34
GPT-Researcher	23.12	554.41 s	41.15	38.58	37.55	46.03	45.62	65.58	9.40
ParallelResearch (-AP, -RO)	27.88	**207.06** s	41.33	38.61	38.09	46.01	45.80	70.06	17.35
ParallelResearch	**39.30**	367.88 s	**41.92**	**39.55**	**38.61**	**46.36**	**45.83**	58.25	22.94

Table 2: Evaluation of deep research frameworks on DeepResearchGym under fixed time budgets. Scores are assessed by **`gpt-4.1-mini-2025-04-14`** and averaged over 5 runs. All metrics reported with 95% confidence intervals.

	Throughput	Quality							Relevance		Faithfulness
	# Nodes	**Overall**	Clarity	Depth	Balance	Breadth	Support	Insight	KPR	KPC (↓)	Cit. Recall
2 minutes											
GPT-Researcher	8.00 ± 0.18	74.06 ± 0.88	78.66 ± 1.94	88.88 ± 0.70	84.70 ± 0.80	90.64 ± 0.27	21.22 ± 3.39	80.26 ± 1.88	54.50 ± 2.16	0.76 ± 0.35	85.54 ± 1.77
ParallelResearch	19.42 ± 1.31	**79.78 ± 0.80**	**83.88 ± 1.06**	90.26 ± 0.22	**87.36 ± 0.47**	91.10 ± 0.27	**40.10 ± 3.91**	85.98 ± 1.01	62.29 ± 2.00	0.73 ± 0.26	**94.71 ± 0.87**
ParallelResearch (-AP)	8.94 ± 0.97	79.44 ± 0.76	86.64 ± 0.68	**90.58 ± 0.24**	87.26 ± 0.72	**92.22 ± 0.39**	33.80 ± 3.88	86.16 ± 1.15	62.73 ± 1.99	0.77 ± 0.25	84.88 ± 1.78
ParallelResearch (-AP, -RO)	**21.14 ± 1.01**	78.88 ± 0.96	82.10 ± 1.37	89.04 ± 0.75	85.36 ± 1.02	89.94 ± 0.92	39.62 ± 3.91	**87.22 ± 0.86**	**66.72 ± 2.01**	**0.43 ± 0.21**	94.36 ± 0.86
10 minutes											
GPT-Researcher	23.94 ± 0.83	79.52 ± 0.80	82.78 ± 1.27	89.96 ± 0.41	86.60 ± 0.51	90.64 ± 0.44	39.48 ± 3.91	87.64 ± 0.78	63.88 ± 1.88	0.61 ± 0.27	95.53 ± 0.80
ParallelResearch	**98.43 ± 0.19**	**83.90 ± 0.78**	83.08 ± 1.25	90.52 ± 0.43	**88.08 ± 0.55**	**94.42 ± 0.44**	**58.10 ± 3.81**	89.18 ± 0.46	66.16 ± 1.93	0.55 ± 0.22	95.96 ± 1.03
ParallelResearch (-AP)	62.32 ± 9.84	83.12 ± 0.79	82.80 ± 1.30	**90.60 ± 0.31**	87.86 ± 0.59	93.88 ± 0.43	54.20 ± 3.94	**89.38 ± 0.25**	**68.42 ± 1.91**	**0.56 ± 0.23**	**97.57 ± 0.49**
ParallelResearch (-AP, -RO)	68.00 ± 0.00	82.69 ± 1.04	**83.28 ± 1.32**	89.42 ± 0.95	87.44 ± 0.87	92.46 ± 0.76	56.24 ± 3.85	87.32 ± 1.14	66.44 ± 1.98	0.51 ± 0.21	96.52 ± 0.65

breadth (parallel subqueries at the same level), depth (speculative deepening of paths), and across the runtime orchestrators at different recursive hierarchies in Algorithm 1.

The engine operates by submitting all research nodes n_i^R to a global asynchronous task pool as soon as they are planned and orchestrated. Each node is parameterized by its local query q^i, depth d_i, parent identifier, and a unique status identifier. Dependencies are enforced dynamically: a child node n_j^R becomes eligible for execution only once its parent n_i^R completes its initial research phase, but speculative spawning allows planning and partial execution to begin earlier under the runtime orchestrator's guidance.

ParallelResearch's engine leverages non-blocking asynchronous calls, allowing tasks to progress independently. This approach mitigates bottlenecks inherent in sequential or coarse-grained parallel execution, where nodes must wait for unrelated dependencies to complete. For instance, as illustrated in Figure 3, child nodes (e.g., `D`, `E`, `F`) can initiate immediately upon their respective parents' (`A`, `B`) completion, without delays from slower siblings like `C`.

5 Experiments

5.1 Benchmarks and Evaluation Metrics

We evaluate our method on two deep research benchmarks.

DeepResearch Bench (Du et al., 2025) comprises 100 PhD-level research tasks across 22 distinct fields. These tasks were designed by domain experts based on a statistical analysis of over 96,000 real-world user queries from search-enabled LLM interactions. We use the whole English subset with 50 questions for evaluation. The benchmark proposes two evaluation frameworks: RACE for report quality and FACT for citation trustworthiness.

- **RACE**:
  - **Comprehensiveness**: Evaluates thorough coverage of relevant aspects, including diverse perspectives and key subtopics, ensuring understanding without omissions.

  - **Depth**: Assesses level of detail, analysis, and insights beyond surface-level, including causes, impacts, and trends.
  - **Instruction following**: Checks adherence to query requirements, ensuring alignment with intent by following the topic and answering directly.
  - **Readability**: Assesses clarity through structure, language, and ease of understanding.
- **FACT**:
  - **Effective citation count**: Measures factual abundance by counting the number of unique, relevant citations that effectively support key statements in the report.
  - **Citation accuracy**: Evaluates citation trustworthiness by assessing the proportion of citations that effectively support the referenced claims.

DeepResearchGym (Coelho et al., 2025) is an open-source evaluation sandbox for deep research systems. The benchmark consists of 1,000 complex, high-engagement queries from the Researchy Questions dataset (Rosset et al., 2025). Due to resource constraints, we randomly sampled 100 for evaluation. To ensure representativeness, we performed rigorous statistical validation with Mann-Whitney U, Kolmogorov-Smirnov, and equivalence testing in Appendix B.1, Table 3, showing no significant distributional differences between our sample and the full dataset across all feature dimensions ($p > 0.05$, Cohen's d ranging from -0.234 to 0.070). It employs an LLM-as-a-judge protocol to assess generated reports along three dimensions:

- **Quality:** Measures the report quality with a rubric of fine-grained criteria, including clarity, depth, balance, breadth, insightfulness, and support.
- **Relevance:** Measures how well the generated reports satisfy the information needs.
- **Faithfulness:** Measures the report and citation factuality.

5.2 Evaluation Setup

To ensure a fair and controlled evaluation where performance differences can be attributed solely to our runtime orchestration, we implement ParallelResearch on top of GPT-Researcher system (Elovic, 2023) and treat the original GPT-Researcher as baseline. Although our approach is system-agnostic, adopting a single, stable baseline allows us to isolate the effect of orchestration while keeping all other components fixed. We use the search API from DeepResearchGym across all experiments for consistency.

On DeepResearch Bench, we allow the system to freely execute with a maximum breadth and depth. On DeepResearchGym, we set maximum execution times at 2 and 10 minutes to reflect realistic usage scenarios:

- The 2-minute cutoff reflects human multitasking behavior, where information workers spend an average of $\sim$2–3 minutes on events or tools before task switching (González & Mark, 2004).
- The 10-minute threshold matches the average duration of a "working sphere" (González & Mark, 2004) and is supported by evidence from high-performance computing (Schlagkamp & Renker, 2015) and crowdsourcing tasks (Bernstein et al., 2011), indicating that a 10-minute window preserves task continuity without slowdown.

5.3 Results

DeepResearch Bench. To assess our system's performance relative to other deep research agents, we evaluate it under flexible time budgets on DeepResearch Bench. As shown in Table 1, ParallelResearch achieves substantial efficiency gains, processing 39.3 nodes on average while reducing latency by 1.51$\times$ compared to the baseline.

In terms of quality, our proposed framework consistently improves across all RACE sub-metrics. Compared to the ablation without adaptive planning and runtime orchestration, ParallelResearch incurs higher latency but conducts more research at a comparable throughput, highlighting a favorable trade-off between efficiency and comprehensiveness. Importantly, ParallelResearch also achieves a performance competitive with commercial systems like Grok Deeper Search and Perplexity Research, narrowing the gap with proprietary agents. Beyond quantitative metrics, we also provide a detailed case analysis of ParallelResearch's adaptability across diverse scenarios in Appendix D.

[1] https://huggingface.co/spaces/Ayanami0730/DeepResearch-Leaderboard

DeepResearchGym. Under fixed time budgets, ParallelResearch consistently delivers superior throughput, processing substantially more research than the GPT-Researcher baseline (up to 4.11× in the 10-minute setup), as shown in Table 2. Per-component ablations with 95% confidence intervals isolate the contribution of each mechanism, demonstrating statistically significant gains from adaptive planning and runtime orchestration, which enable speculative execution without compromising quality.

Beyond throughput, ParallelResearch also improves overall response quality, with clear improvements in balance, breadth, and insight metrics. These gains highlight its ability to maintain a robust trade-off between expansive coverage and focused analysis, particularly under tight time constraints. Notably, the overall quality of ParallelResearch with 2-minute execution even surpasses that of the GPT-Researcher baseline with 10 minutes, demonstrating a 5× speed-up while preserving quality.

To validate generalizability, we also experiment on open-source `Qwen3-235B-A22B-Instruct`. Results in Appendix Table 6 indicate that ParallelResearch achieves a consistent 5× quality-preserving speedup, confirming that our efficiency improvements transfer across model families.

Overall, these results underscore ParallelResearch's capacity for efficient deep research across diverse benchmarks, model families, LLM judges, and time constraints.

6 Analysis

To provide deeper insights into ParallelResearch's robustness and practical viability, we conducted comprehensive analyses on evaluation reliability, failure patterns, and economic efficiency.

Evaluation Reliability. To validate the reliability of LLM-as-a-judge evaluations, we conducted inter-rater analysis across five independent runs in Appendix B.2, Table 4. The Intraclass Correlation Coefficient (ICC) scores are consistently above 0.85 (mostly $>$0.9) across all metrics and systems, indicating excellent consistency and reliability of the LLM judges. This confirms that our evaluation protocol can produce stable and trustworthy assessments.

Failure Analysis. To understand when ParallelResearch excels or struggles, we analyzed performance patterns across different query types in Appendix C.2, Figure 4. ParallelResearch achieves a 70% win rate overall, with superior performance on knowledge-intensive and reasoning-intensive queries where adaptive planning effectively balances breadth and depth. These query types benefit most from dynamic resource allocation across multiple research branches. However, performance limitations emerge primarily for ambiguous or incomplete queries where the optimal decomposition strategy is unclear, suggesting opportunities for future improvements in intent clarification mechanisms.

Economic Cost Analysis. Cost profiling in Appendix F, Table 7 shows orchestration overhead accounts for only 7% of total LLM API costs, while early pruning saves 3.39× more than orchestration costs. A 2-minute ParallelResearch run costs $0.21 and outperforms a 10-minute baseline costing $0.39. Sensitivity analysis (Appendix C.1, Table 5) confirms ParallelResearch maintains efficiency at 2 minutes with strategic time utilization, achieving strong gains in Support metrics (+16–18, $p<0.001$) versus the baseline's less efficient sensitivity.

7 Conclusion

By formulating efficient deep research as a tree-structured optimization problem and exploiting parallelism across nodes, ParallelResearch achieves significant improvements in research throughput and response quality through adaptive planning, runtime orchestration, and a fully-asynchronous execution infrastructure. Future work includes incorporating richer modalities beyond text and exploring tighter integration with human-in-the-loop feedback to further improve policies' decision-making.

ETHICS STATEMENT

This work does not involve private data or sensitive information. Experiments were conducted using publicly available resources. While our framework aims to improve the efficiency and quality of deep research systems, we acknowledge the broader risks of misuse, including the potential amplification of biased or unreliable information. Responsible deployment requires careful selection of data sources, robust fact-checking, and adherence to ethical standards.

REFERENCES

Salaheddin Alzubi, Creston Brooks, Purva Chiniya, Edoardo Contente, Chiara von Gerlach, Lucas Irwin, Yihan Jiang, Arda Kaz, Windsor Nguyen, Sewoong Oh, et al. Open deep search: Democratizing search with open-source reasoning agents. *arXiv preprint arXiv:2503.20201*, 2025.

Michael S Bernstein, Joel Brandt, Robert C Miller, and David R Karger. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 33–42, 2011.

Tianle Cai, Jiayi Li, Haotian Geng, Yuxin Ge, Shizhe Zhang, et al. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

João Coelho, Jingjie Ning, Jingyuan He, Kangrui Mao, Abhijay Paladugu, Pranav Setlur, Jiahe Jin, Jamie Callan, João Magalhães, Bruno Martins, et al. Deepresearchgym: A free, transparent, and reproducible evaluation sandbox for deep research. *arXiv preprint arXiv:2505.19253*, 2025.

Yifu Ding, Wentao Jiang, Shunyu Liu, Yongcheng Jing, Jinyang Guo, Yingjie Wang, Jing Zhang, Zengmao Wang, Ziwei Liu, Bo Du, et al. Dynamic parallel tree search for efficient llm reasoning. *arXiv preprint arXiv:2502.16235*, 2025.

Mingxuan Du, Benfeng Xu, Chiwei Zhu, Xiaorui Wang, and Zhendong Mao. Deepresearch bench: A comprehensive benchmark for deep research agents. *arXiv preprint arXiv:2506.11763*, 2025.

Assaf Elovic. GPT Researcher: LLM based autonomous agent that conducts deep local and web research on any topic and generates a long report with citations, July 2023. URL `https://github.com/assafelovic/gpt-researcher`.

Tianqing Fang, Zhisong Zhang, Xiaoyang Wang, Rui Wang, Can Qin, Yuxuan Wan, Jun-Yu Ma, Ce Zhang, Jiaqi Chen, Xiyun Li, et al. Cognitive kernel-pro: A framework for deep research agents and agent foundation models training. *arXiv preprint arXiv:2508.00414*, 2025.

Dane Gambrell. Ai for egovernance: Combining artificial intelligence and collective intelligence to develop evidence-based ai policy. In *2025 Eleventh International Conference on eDemocracy & eGovernment (ICEDEG)*, pp. 317–324. IEEE, 2025.

Hongcheng Gao, Yue Liu, Yufei He, Longxu Dou, Chao Du, Zhijie Deng, Bryan Hooi, Min Lin, and Tianyu Pang. Flowreasoner: Reinforcing query-level meta-agents. *arXiv preprint arXiv:2504.15257*, 2025.

Victor M González and Gloria Mark. " constant, constant, multi-tasking craziness" managing multiple working spheres. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 113–120, 2004.

Michael Haman and Milan Školník. Fake no more: the redemption of chatgpt in literature reviews. *Accountability in Research*, pp. 1–3, 2025.

Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Bowei Xia, Tao Sun, Ziyu Ye, Zhaoxuan Jin, Yingru Li, Qiguang Chen, et al. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation. *arXiv preprint arXiv:2505.23885*, 2025.

Shamsi T Iqbal and Eric Horvitz. Disruption and recovery of computing tasks: field study, analysis, and directions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 677–686, 2007.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. Dspy: Compiling declarative language model calls into self-improving pipelines. *The Twelfth International Conference on Learning Representations*, 2024.

LangChain. Langgraph: Build resilient language agents as graphs, 2024. URL https://langchain-ai.github.io/langgraph/.

Langchain. Open deep research, 2025. URL https://github.com/langchain-ai/open_deep_research.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Gloria Mark, Daniela Gudith, and Ulrich Klocke. The cost of interrupted work: more speed and stress. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 107–110, 2008.

Xiaohui Miao, Senzhang Zuo, Ang Li, Jiaqi Guo, Xiaoying Zhang, Yifan Xing, Xiaoliang Qian, Yanzhi Gao, Jingren Zhou, and Fan Zhou. Specinfer: Accelerating large language model serving with speculative inference. *arXiv preprint arXiv:2305.09781*, 2023.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Fan Nie, Lan Feng, Haotian Ye, Weixin Liang, Pan Lu, Huaxiu Yao, Alexandre Alahi, and James Zou. Weak-for-strong: Training weak meta-agent to harness strong executors. *arXiv preprint arXiv:2504.04785*, 2025.

Boye Niu, Yiliao Song, Kai Lian, Yifan Shen, Yu Yao, Kun Zhang, and Tongliang Liu. Flow: A modular approach to automated agentic workflow generation. *arXiv preprint arXiv:2501.07834*, 2025.

OpenAI. Swarm: Educational framework exploring ergonomic, lightweight multi-agent orchestration. https://github.com/openai/swarm, 2024.

Rui Pan, Yinwei Dai, Zhihao Zhang, Gabriele Oliaro, Zhihao Jia, and Ravi Netravali. Specreason: Fast and accurate inference-time compute via speculative reasoning. *arXiv preprint arXiv:2504.07891*, 2025.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

Corbin Rosset, Ho-Lam Chung, Guanghui Qin, Ethan Chau, Zhuo Feng, Ahmed Awadallah, Jennifer Neville, and Nikhil Rao. Researchy questions: A dataset of multi-perspective, decompositional questions for deep research. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3712–3722, 2025.

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

Stephan Schlagkamp and Johanna Renker. Acceptance of waiting times in high performance computing. In *International Conference on Human-Computer Interaction*, pp. 709–714. Springer, 2015.

Hao Wen, Yifan Su, Feifei Zhang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. Parathinker: Native parallel thinking as a new paradigm to scale llm test-time compute. *arXiv preprint arXiv:2509.04475*, 2025.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 3(4), 2023.

Renjun Xu and Jingwen Peng. A comprehensive survey of deep research: Systems, methodologies, and applications. *arXiv preprint arXiv:2506.12594*, 2025.

Wang Yang, Xiang Yue, Vipin Chaudhary, and Xiaotian Han. Speculative thinking: Enhancing small-model reasoning with large model guidance at inference time. *arXiv preprint arXiv:2504.12329*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023b.

Guibin Zhang, Kaijie Chen, Guancheng Wan, Heng Chang, Hong Cheng, Kun Wang, Shuyue Hu, and Lei Bai. Evoflow: Evolving diverse agentic workflows on the fly. *arXiv preprint arXiv:2502.07373*, 2025a.

Hongwei Zhang, Ji Lu, Shiqing Jiang, Chenxiang Zhu, Li Xie, Chen Zhong, Haoran Chen, Yurui Zhu, Yongsheng Du, Yanqin Gao, et al. Co-sight: Enhancing llm-based agents via conflict-aware meta-verification and trustworthy reasoning with structured facts. *arXiv preprint arXiv:2510.21557*, 2025b.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762*, 2024.

Wentao Zhang, Liang Zeng, Yuzhen Xiao, Yongcong Li, Ce Cui, Yilei Zhao, Rui Hu, Yang Liu, Yahui Zhou, and Bo An. Agentorchestra: Orchestrating hierarchical multi-agent intelligence with the tool-environment-agent (tea) protocol. *arXiv preprint arXiv:2506.12508*, 2025c.

Tong Zheng, Hongming Zhang, Wenhao Yu, Xiaoyang Wang, Xinyu Yang, Runpeng Dai, Rui Liu, Huiwen Bao, Chengsong Huang, Heng Huang, et al. Parallel-r1: Towards parallel thinking via reinforcement learning. *arXiv preprint arXiv:2509.07980*, 2025.

He Zhu, Tianrui Qin, King Zhu, Heyuan Huang, Yeyi Guan, Jinxiang Xia, Yi Yao, Hanhao Li, Ningning Wang, Pai Liu, et al. Oagents: An empirical study of building effective agents. *arXiv preprint arXiv:2506.15741*, 2025.

Table 3: Statistical comparison between the sampled set's 100 questions and the whole DeepResearchGym's 1000 questions. MW represents the Mann-Whitney U test. KS represents the Kolmogorov-Smirnov test. TOST stands for the Equivalence Testing with Two One-Sided Tests.

Score Type	**Sampled Set** Mean ± SD	**DeepResearchGym** Mean ± SD	**MW** p-value	**KS** p-value	**Cohen's d**	**TOST p-value (δ=0.5×SD)**	**Equiv?**
Decompositional	0.735 ± 0.088	0.734 ± 0.087	0.9563	0.9348	0.013	0.0000	Yes
Nonfactoid	1.022 ± 0.085	1.017 ± 0.083	0.2800	0.5207	0.070	0.0000	Yes
Ambiguous	0.710 ± 0.697	0.797 ± 0.735	0.2819	0.9898	-0.119	0.0001	Yes
Incompleteness	1.390 ± 1.580	1.527 ± 1.546	0.2330	0.9433	-0.088	0.0000	Yes
Assumptive	0.920 ± 2.226	0.964 ± 2.122	0.1772	0.6794	-0.021	0.0000	Yes
Multi-faceted	7.140 ± 1.149	7.115 ± 1.250	0.9458	1.0000	0.020	0.0000	Yes
Knowledge-intensive	6.510 ± 1.559	6.822 ± 1.307	0.0526	0.5990	-0.234	0.0057	Yes
Subjective	5.060 ± 2.716	4.999 ± 2.666	0.6871	0.9433	0.023	0.0000	Yes
Reasoning-intensive	6.480 ± 1.144	6.600 ± 1.188	0.1535	0.3537	-0.101	0.0001	Yes
Harmful	0.000 ± 0.000	0.000 ± 0.000	1.0000	1.0000	—	—	—

Note: TOST p-value < 0.05 indicates two groups are statistically equivalent within margin δ.

A Implementation Details

A.1 Experimental Setups

For model configuration, we use `gpt-4.1-mini-2025-04-14` for the main research processing, and `o3-mini-2025-01-31` for implementing the policies in Equations 6, 8 and 9 for adaptive research planning and runtime orchestration.

To ensure fair comparisons among deep research systems, we impose a maximum execution time for research trees. Once the time cut-off is reached, the research process terminates immediately, and the system generates a response based on the findings and context gathered up to that point. To allow all systems to fully utilize their time budgets, we set the maximum tree depth to 10 and the maximum breadth to 4 within the GPT-Researcher framework. To provide additional flexibility, the adaptive research planning module may expand the breadth up to 6 when necessary.

Additionally, to control the computational costs introduced by the runtime orchestration layer, we set an interval of 8 seconds between successive evaluations of goal satisfaction and research quality.

B Evaluation Robustness and Reliability

We evaluate the reliability of our evaluation by analyzing whether (1) our evaluation sample represents the broader benchmark population and (2) LLM judges produce consistent measurements.

B.1 Sample Representativeness

Evaluating all 1000 DeepResearchGym questions requires excessive compute and LLM token consumption. Due to the resource constraints, we randomly sampled 100 questions and validated this subset's representativeness by comparing its distributional properties against the full dataset, based on the characterizations data from the original Researchy Questions dataset (Rosset et al., 2025).

Table 3 shows that Mann-Whitney U and Kolmogorov-Smirnov tests yield $p > 0.05$ across all dimensions, indicating no significant distributional differences. TOST equivalence tests ($p < 0.05$) confirm statistical equivalence within a medium effect size, and Cohen's d values are small (-0.234 to 0.070). These results confirm our sample accurately represents the DeepResearchGym benchmark.

B.2 LLM Judge Reliability

Deep research evaluation is challenging due to report length and complexity, making manual assessment costly. We adopt LLM-as-a-judge protocols and verify their consistency via inter-rater reliability analysis. We ran the judge five times on identical inputs and measured agreement using the Intraclass Correlation Coefficient (ICC).

Table 4: Inter-rater reliability analysis using Intraclass Correlation Coefficient (ICC) across three systems under two time constraints. Higher ICC values indicate stronger agreement between multiple runs of the LLM judge.

Time Budget	System	Quality	KPR	KPC	Citation Recall
2 minutes	GPT-Researcher	0.917**	0.995**	0.968**	0.979**
	ParallelResearch (-AP, -RO)	0.909**	0.996**	0.991**	0.932**
	ParallelResearch	0.890*	0.994**	0.966**	0.961**
10 minutes	GPT-Researcher	0.859*	0.995**	0.994**	0.985**
	ParallelResearch (-AP, -RO)	0.942**	0.996**	0.975**	0.969**
	ParallelResearch	0.882*	0.991**	0.962**	0.979**

Note: * Good reliability ($0.75 \leq$ ICC < 0.90), ** Excellent reliability (ICC ≥ 0.90).

Table 5: Statistical significance of performance metric changes from 2-minute to 10-minute time budgets, evaluated via independent samples t-tests (Δ represents mean change).

Metric	GPT-Researcher			ParallelResearch (-AP, -RO)			ParallelResearch		
	Δ	t-value	p-value	Δ	t-value	p-value	Δ	t-value	p-value
Quality									
Overall	+5.46***	9.03	<0.001	+3.81***	5.27	<0.001	+4.12***	7.21	<0.001
Clarity	+4.12***	3.49	<0.001	+1.18	1.22	0.224	-0.80	-0.96	0.339
Depth	+1.08**	2.60	0.010	+0.38	0.62	0.538	+0.26	1.06	0.288
Balance	+1.90***	3.93	<0.001	+2.08**	3.04	0.002	+0.72	1.95	0.052
Breadth	+0.00	0.00	1.000	+2.52***	4.15	<0.001	+3.32***	12.64	<0.001
Support	+18.26***	6.90	<0.001	+16.62***	5.93	<0.001	+18.00***	6.45	<0.001
Insightfulness	+7.38***	7.10	<0.001	+0.10	0.14	0.891	+3.20***	5.67	<0.001
Relevance									
KPR	+9.37***	6.40	<0.001	-0.28	-0.20	0.845	+3.88**	2.73	0.006
KPC (↓)	-0.15	-0.66	0.508	+0.08	0.52	0.604	-0.18	-1.06	0.289
Faithfulness									
Citation Recall	+9.99***	10.09	<0.001	+2.16***	3.92	<0.001	+1.25	1.81	0.070

Note: Significance levels: *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$.

Table 4 shows ICC scores exceeding 0.85 across all metrics and systems, with most surpassing 0.90 (excellent reliability). Quality metric ICCs range from 0.859 to 0.942, while KPR and KPC exceed 0.96. Citation Recall scores are between 0.932 and 0.985. These high ICC values validate the consistency and reproducibility of our LLM-based evaluation.

C Performance Analysis

We analyze system performance across time budgets and query characteristics to understand success and failure modes.

C.1 Time Budget Sensitivity Analysis

We examine performance scaling when increasing the time budget from 2 to 10 minutes (5×). Table 5 shows GPT-Researcher significantly improves across nearly all metrics with more time. ParallelResearch (-AP, -RO) shows selective gains in quality and breadth but minimal improvement in clarity or depth. ParallelResearch displays a distinct pattern: significant overall quality gains (+4.12) are driven by breadth (+3.32), support (+18.00), and insightfulness (+3.20), with minimal changes in clarity and depth. This indicates ParallelResearch's adaptive planning and orchestration prioritize expanding coverage and strengthening evidence over refining already adequate aspects, effectively optimizing the quality-cost tradeoff.

C.2 Success and Failure Pattern Analysis

We analyze question characteristics to identify where ParallelResearch excels or struggles relative to GPT-Researcher.Specifically, we divided the 100 DeepResearchGym questions into two groups based on the 10-minute time budget results: 70 questions where ParallelResearch achieved higher overall quality than GPT-Researcher ("Wins"), and 30 questions where it performed worse

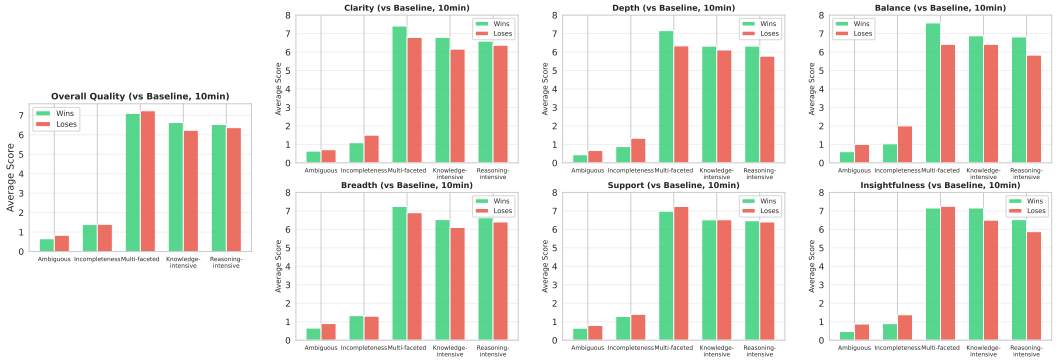


Figure 4: ParallelResearch performance analysis (v.s. GPT-Researcher baseline, 10 min). We compare the average characterization scores for the 70 questions where ParallelResearch *Wins* (outperforms the baseline) against the 30 questions where it *Loses*. Scores are plotted across five characterization dimensions (x-axis) from the Researchy Questions dataset (Rosset et al., 2025) and evaluated with respect to six quality sub-metrics.

("Loses"). For each group, we computed the average scores across five key characterization dimensions from the Researchy Questions dataset: decompositional, multi-faceted, knowledge-intensive, reasoning-intensive, ambiguous, and incomplete. We then analyzed these patterns across six quality sub-metrics: overall quality, clarity, depth, balance, breadth, and support.

Figure 4 shows ParallelResearch outperforms the baseline on "knowledge-intensive" and "reasoning-intensive" questions, achieving better clarity, depth, balance, and breadth. This aligns with our design: adaptive planning broadens exploration, while orchestration focuses on high-value paths. Conversely, ParallelResearch underperforms on "ambiguous" and "incomplete" questions. Ambiguity may confuse the adaptive planner or cause the orchestrator to prematurely prune paths. This suggests a need to improve orchestration policies for handling ambiguous queries.

D Case Analysis

We illustrate ParallelResearch's adaptability with three research trees (all 2-minute cutoff). Case 1 (Figure 5) shows wide expansion for a broad topic (non-alcoholic cocktails), capturing diverse perspectives. Case 2 (Figure 6) shows a compact tree for a narrow query (cislunar situational awareness), terminating early to save resources. Case 3 (Figure 7) deepens analysis based on user constraints. Since thinking models shape planning and orchestration, the framework generates explicit reasoning traces that aid decision interpretation and error tracking.

In particular, we examined the execution logs and analyzed the reasoning traces to gain a clearer understanding of the decision-making process during runtime orchestration. The user query explicitly requests "*only cites high-quality, English-language journal articles*". Consequently, when the orchestrator detects that a node does not satisfy this research goal, it terminates the branch early to avoid unnecessary computation. In this specific case, the orchestrator's reasoning trace is shown below.

Reasoning Trace 1

The current research progress mainly comprises news articles, reports, and policy discussions rather than high-quality, peer-reviewed English journal articles specifically focused on AI-driven labor market restructuring. While the materials are relevant to the broader topic, they do not meet the stringent criteria of academic journal quality necessary to fully satisfy the research goal. The findings provide useful context and cover significant aspects of AI's impact on the labor market, but they lack the depth, methodological rigor, and academic sourcing expected from high-quality journal articles. Consequently, although there is moderate progress on the subject, key elements of the goal remain unmet, suggesting that it would be more cost-efficient to redirect or deepen the research efforts rather than continuing on the current path.

Table 6: Evaluation of deep research frameworks on DeepResearchGym using the open-source `Qwen3-235B-A22B-Instruct-2507` model for all the operations. Scores are assessed by `gpt-4.1-mini-2025-04-14` and averaged over 5 runs. All metrics reported with 95% confidence intervals.

	Throughput	**Quality**							**Relevance**		**Faithfulness**
	# Nodes	**Overall**	Clarity	Depth	Balance	Breadth	Support	Insight	KPR	KPC (↓)	Cit. Recall
2 minutes											
Baseline	3.94 ± 0.18	82.26 ± 0.92	87.68 ± 1.26	91.38 ± 0.39	87.54 ± 0.93	93.94 ± 0.69	46.80 ± 4.20	86.20 ± 1.25	65.25 ± 2.09	**0.47 ± 0.21**	**79.83 ± 2.02**
ParallelResearch	**10.90 ± 0.69**	**84.91 ± 0.76**	**89.22 ± 0.62**	**96.72 ± 0.41**	**89.00 ± 0.66**	**96.22 ± 0.43**	**48.16 ± 4.04**	**90.12 ± 0.43**	**69.37 ± 2.00**	0.93 ± 0.29	74.56 ± 1.65
10 minutes											
Baseline	15.14 ± 0.31	83.48 ± 0.79	**85.56 ± 0.97**	90.14 ± 0.31	87.58 ± 0.67	91.04 ± 0.50	58.92 ± 3.82	87.66 ± 0.98	68.59 ± 1.88	**0.47 ± 0.26**	**95.85 ± 0.86**
ParallelResearch	**68.30 ± 10.04**	**87.68 ± 0.60**	83.04 ± 0.99	**93.84 ± 0.43**	**88.38 ± 0.57**	**94.44 ± 0.44**	**76.64 ± 3.04**	**89.76 ± 0.29**	**69.99 ± 2.05**	0.98 ± 0.31	93.78 ± 0.82

E Generalization to Open-Source Models

We evaluate whether ParallelResearch's benefits transfer to open-source models, which offer cost and privacy advantages but may differ in latency and capability. We test if core mechanisms (adaptive planning, runtime orchestration, and parallelization) remain effective.

E.1 Experimental Setup with Open-Source Models

We evaluated ParallelResearch using the open-source Qwen3-235B-A22B-Instruct-2507 (22B activated parameters), deployed locally for all operations. Experimental settings matched our main experiments, with `gpt-4.1-mini` as the judge, averaged over 5 runs (95% CI).

E.2 Results with Open-Source Models

Table 6 presents comprehensive results comparing GPT-Researcher baseline and ParallelResearch when both use the open-source Qwen3 model. Despite the higher latency of locally-served Qwen3, ParallelResearch achieves substantial throughput gains: 2.77× improvement (10.90 vs 3.94 nodes) under 2-minute budget and 4.51× improvement (68.30 vs 15.14 nodes) under 10-minute budget. These gains demonstrate that ParallelResearch's parallel orchestration mechanisms effectively compensate for individual operation latency.

Most remarkably, ParallelResearch's 2-minute execution achieves higher overall quality (84.91 ± 0.76) than the baseline's 10-minute execution (83.48 ± 0.79), representing a 5× speedup while maintaining superior quality. This finding closely mirrors our results with proprietary models (Table 1), confirming that ParallelResearch's architectural benefits are model-agnostic.

Examining individual metrics reveals interesting patterns. The 2-minute executions achieve higher scores in clarity, depth, and breadth compared to 10-minute executions. This counterintuitive result may reflect limitations in the open-source model's ability to effectively manage long contexts during information aggregation. However, 10-minute executions significantly outperform in support (+28.48 points) and citation recall (+19.22 points), indicating that longer execution still substantially benefits evidentiary quality and faithfulness.

Overall, these results validate that ParallelResearch can successfully transfer to open-source models, making the approach practical for diverse deployment scenarios.

F Economic Cost Analysis

While ParallelResearch introduces additional computational overhead through runtime orchestration (periodic quality and goal satisfaction assessments), it also prunes low-value research paths early, potentially reducing overall costs. Understanding the net economic impact requires detailed analysis of token usage across all components. This section quantifies the monetary costs associated with different systems and examines whether orchestration overhead is offset by efficiency gains.

Table 7 provides a detailed breakdown of LLM API usage and costs averaged across 100 DeepResearchGym questions. We report both orchestration-specific costs (only applicable to ParallelRe-

Table 7: LLM token usage and API cost analysis on DeepResearchGym with different time budgets. All values are averaged over 100 questions. Orchestration metrics only apply to ParallelResearch.

	Orchestration Usage			Total Usage		
	Input Tokens	**Output Tokens**	**Cost ($)**	**Input Tokens**	**Output Tokens**	**Total Cost ($)**
2 minutes						
GPT-Researcher	–	–	–	97,170	5,428	0.1258
ParallelResearch (-AP, -RO)	–	–	–	176,147	9,355	0.2218
ParallelResearch	9,632	1,788	0.0185	155,409	10,031	0.2071
10 minutes						
GPT-Researcher	–	–	–	299,093	18,432	0.3941
ParallelResearch (-AP, -RO)	–	–	–	424,488	28,496	0.5629
ParallelResearch	25,646	4,756	0.0491	487,023	36,409	0.6811

search) and total system costs including all operations (planning, web search, retrieval, reasoning, and report generation).

Under the 2-minute budget, ParallelResearch's orchestration layer consumes 9,632 input tokens and 1,788 output tokens per question on average, costing $0.0185. This represents only 8.9% of the total system cost ($0.2071). Under the 10-minute budget, orchestration costs rise to $0.0491 but remain just 7.2% of total costs ($0.6811). These figures demonstrate that runtime orchestration introduces minimal overhead relative to overall system costs.

Comparing total costs across systems reveals important tradeoffs. GPT-Researcher baseline costs $0.1258 (2-min) and $0.3941 (10-min). The ParallelResearch (-AP, -RO) system (without adaptive planning and orchestration) costs $0.2218 (2-min) and $0.5629 (10-min), showing that adding parallelization alone increases costs by 76% and 43% respectively. ParallelResearch costs $0.2071 (2-min) and $0.6811 (10-min). While ParallelResearch incurs 65% and 73% higher costs than the baseline, it explores 2.1× and 4.5× more research nodes while achieving superior quality (Table 1). Importantly, ParallelResearch's 2-minute execution ($0.2071, Quality: 83.82) achieves comparable quality to the baseline's 10-minute execution ($0.3941, Quality: 83.74) at roughly half the cost, demonstrating substantial economic efficiency.

The key insight is that ParallelResearch achieves better cost-quality tradeoffs by intelligently allocating computational resources. While absolute costs increase moderately, the system delivers disproportionate quality improvements by focusing resources on high-value research paths and pruning redundant exploration. The orchestration overhead (7-9% of costs) is far outweighed by the gains from improved resource allocation.

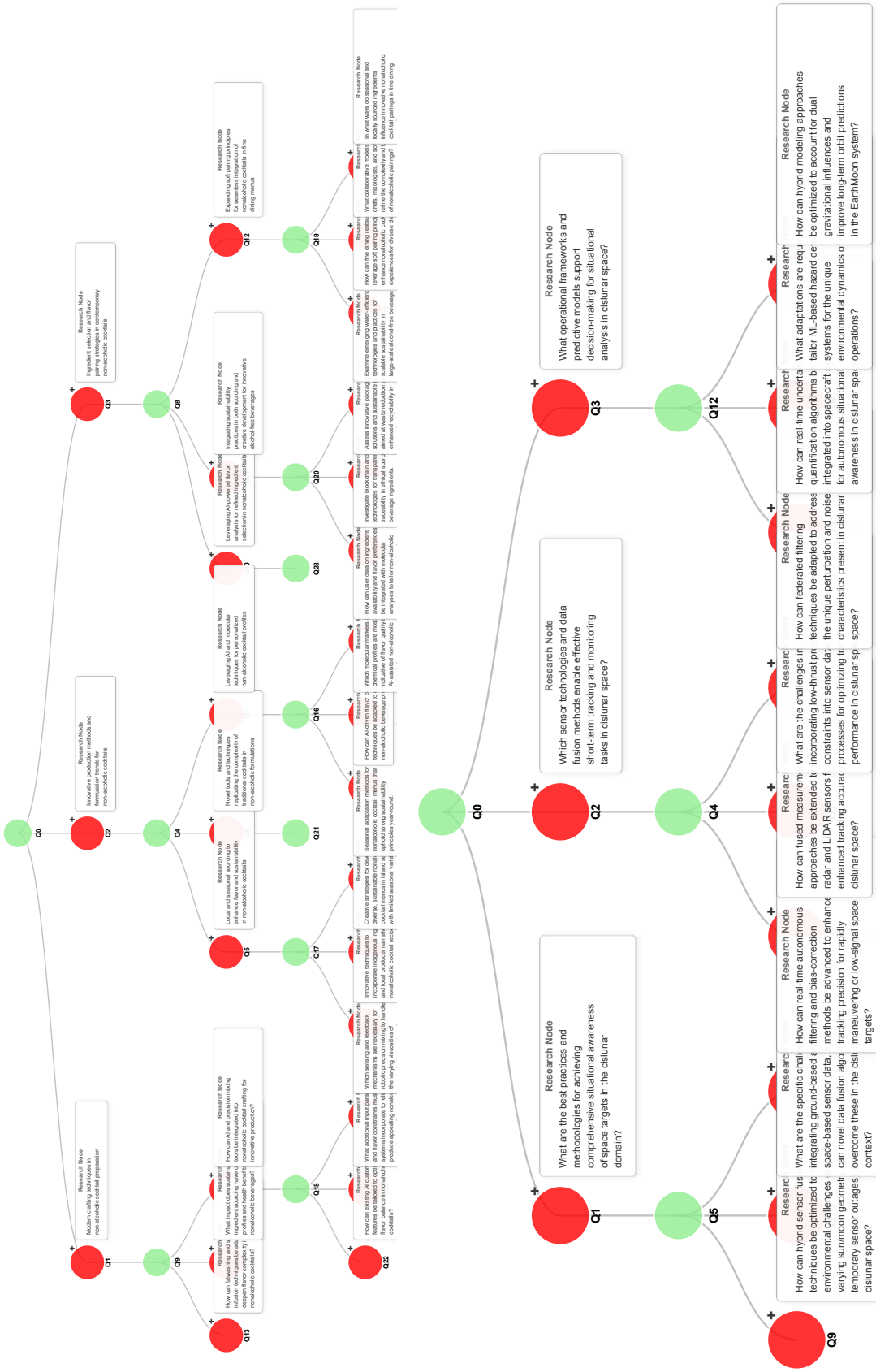


Figure 6: Research tree generated by ParallelResearch for a narrow, domain-specific query: "*How to conduct comprehensive and accurate situational awareness of space targets in the cislunar space, and support the effectiveness of short-term cislunar space tracking and monitoring tasks?*"

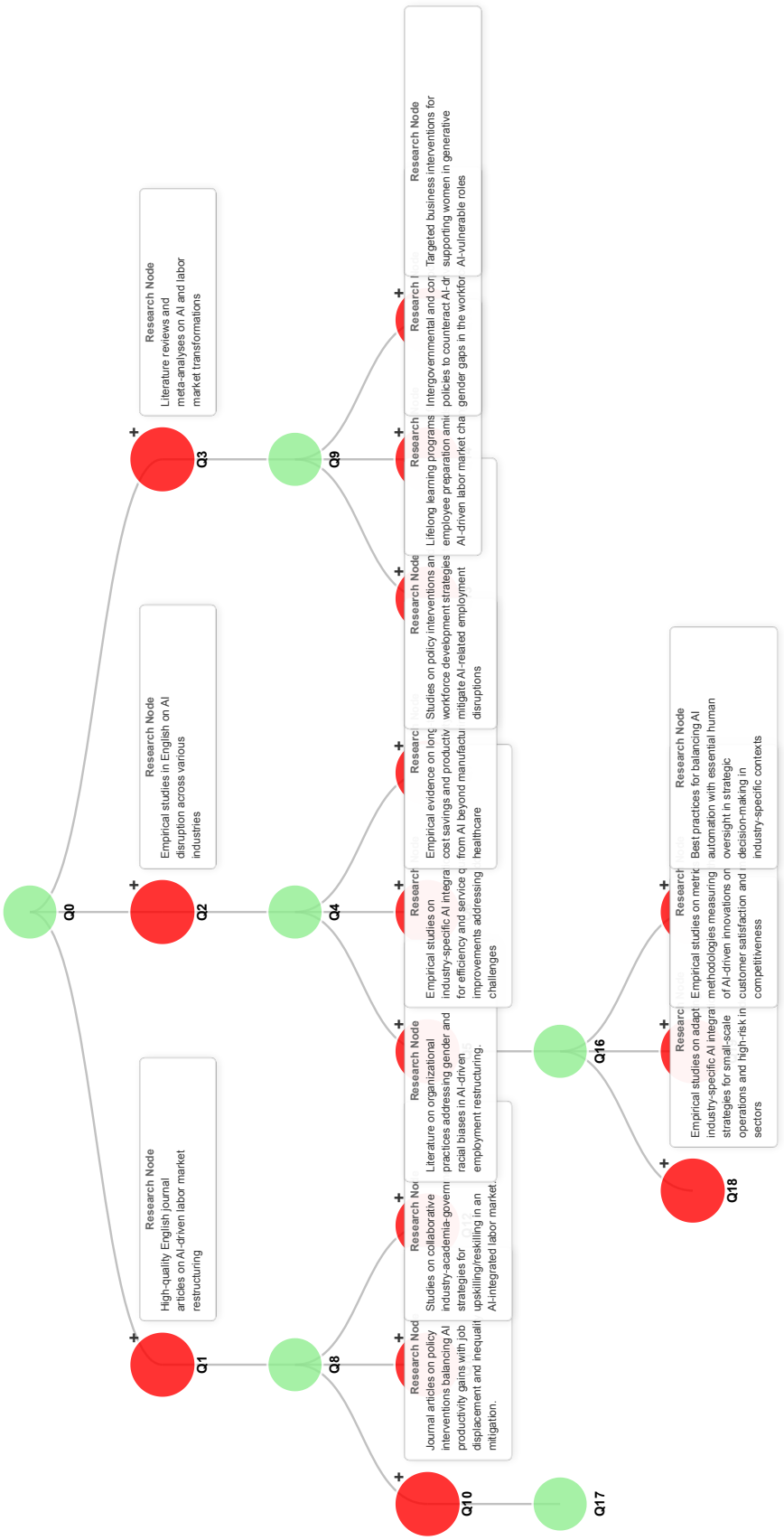


Figure 7: Research tree generated by ParallelResearch for a user-focused query with explicit demands: "*Please write a literature review on the restructuring impact of Artificial Intelligence (AI) on the labor market. Focus on how AI, as a key driver of the Fourth Industrial Revolution, is causing significant disruptions and affecting various industries. Ensure the review only cites high-quality, English-language journal articles.*"