Evaluating Agentic Research: Single-Agent vs Autonomous Multi-Agent Decomposition on WebWalker QA Tasks

Benjamin Grayzel*
Dartmouth College
benjamin.a.grayzel.gr@dartmouth.edu

Abstract

Multi-agent systems powered by large language models promise improved performance through task decomposition and specialized coordination, yet empirical comparisons with single-agent baselines remain limited. We conduct a systematic evaluation comparing a single-agent and an autonomous multi-agent decomposition system, where a lead agent dynamically decides whether and how to instantiate subagents. We evaluate on the WebWalker QA benchmark using Claude Haiku 4.5 as our base model, executing 200 runs (100 per architecture) and extracting 26 comprehensive metrics spanning performance, cost, coordination, and behavioral dynamics. Contrary to expectations, the single-agent architecture achieves 50% accuracy compared to 48% for (multi-agent) autonomous decomposition, while requiring only half the cost per correct answer ($0.49 vs. $0.98) and 44% less execution time. We observed an emergent tendency toward minimal task decomposition; 71% of runs of the autonomous multi-agent implementation effectively collapsed into a single-subagent system. We observe a thrashing-like pattern: higher resource usage is associated with lower success rates. These findings challenge assumptions that autonomous multi-agent coordination reliably yields advantages for web-based information retrieval. For the task and architecture evaluated, the single-agent baseline offers superior cost-effectiveness under the evaluated conditions.

1 Introduction

Large language models have become powerful foundations for autonomous agents capable of complex reasoning, tool use, and multi-step task execution. LLM-based agents can now navigate web environments [6, 10, 15], interact with external tools [14], and perform sophisticated information retrieval [17]. Naturally, these capabilities have sparked interest in multi-agent systems where specialized agents collaborate to tackle complex problems [7, 11].

The promise is compelling. Anthropic's multi-agent research system demonstrated 90.2% performance gains over single-agent baselines [1], and such "compound AI systems" that decompose tasks across multiple model calls offer improved performance through parallel exploration and specialized coordination [5, 9], but these benefits are only realized if the system decomposes effectively and if coordination costs do not dominate. Yet other studies document non-monotonic scaling behaviors where additional model calls can paradoxically degrade performance [4, 5]. Most recently,

[*] AI Use Disclosure: Artificial intelligence tools (primarily Anthropic's Claude Code CLI running the Claude 4.5 model series) were used throughout the research process, including pipeline engineering, data analysis, background research, and manuscript drafting. All AI-generated content was verified and validated by the author.

a comprehensive evaluation across 15 datasets found that multi-agent advantages diminish substantially with more capable models—from ∼10% gains to just 3%—while incurring 4-220× higher token costs [8].

Despite this enthusiasm, a fundamental question remains underexplored: Do multi-agent architectures, and specifically *autonomous multi-agent decomposition, outperform well-designed single-agent baselines* on complex information retrieval tasks? Evaluation frameworks like τ^2-Bench [3], AgentBoard [13], and HELM [12] have established methodologies for assessing agent capabilities. But systematic comparisons between single-agent and multi-agent architectures remain surprisingly scarce. We lack comprehensive empirical studies on parallel architectures measuring not just task performance but also resource efficiency, coordination dynamics, and the behavioral patterns that distinguish these approaches.

This paper presents a rigorous comparative evaluation of a single-agent and an autonomous multi-agent deep research architecture on the WebWalker QA benchmark [15]—a recently introduced dataset for multi-hop web navigation and information retrieval. We conduct 200 evaluation runs (100 per architecture) using Claude Haiku 4.5 via the Claude Agent SDK, extracting 26 comprehensive metrics spanning performance, cost, coordination, and behavioral dynamics. The results are counterintuitive. The single-agent architecture achieves 50% accuracy compared to 48% for autonomous decomposition, while requiring only half the cost ($0.49 vs $0.98 per correct answer) and 44% less time. We identify a *thrashing* pattern where higher resource usage is associated with lower success rates consistent with failure modes in the decomposition/coordination policy, though this pattern may also reflect underlying question difficulty.

We emphasize that this work evaluates a specific class of autonomous multi-agent systems: those in which a lead agent dynamically decides whether and how to decompose tasks into subagents rather than hand-designed or fixed-decomposition multi-agent pipelines.

1.1 Research Questions

This work addresses four primary research questions:
RQ1: Performance Comparison. Does autonomous multi-agent decomposition outperform a single-agent baseline on WebWalker QA in accuracy, cost per correct answer, and wall-clock time?
RQ2: Behavioral Predictors. What behavioral patterns and metrics predict system success or failure? Can we build predictive models for accuracy and resource usage?
RQ3: Decomposition & Coordination Dynamics. How do decomposition decisions and coordination dynamics affect outcomes, including when agents are spawned, how many are used, and how successfully they complete tasks?
RQ4: Cost-Benefit Trade-offs. Under what conditions does autonomous decomposition justify coordination overhead, and when should systems default to single-agent (or escalate via cascade)?

2 Related Work

2.1 Multi-Agent Systems with Large Language Models

The integration of LLMs into multi-agent systems has emerged as a transformative approach for modeling collaborative intelligence [7, 11]. Agent orchestration addresses the escalating complexity of multi-agent systems at scale, enabling dynamic introduction of specialized agents [9]. However, significant gaps exist between multi-agent theory and current implementations, with many studies relying on frail architectures, outdated base models, and incompatible single-agent baselines.

2.2 LLM Agent Evaluation and Benchmarks

Evaluating LLM agents requires frameworks capturing interactive, multi-turn behavior [14]. τ^2-Bench [3] provides a controlled testbed for agents calling tools in an evolving environment, while AgentBoard [13] introduces progress rate metrics capturing incremental advancement rather than just final success. HELM [12] measures robustness under prompt transformations. These frameworks assess individual agent capabilities, but systematic comparisons between single-agent and multi-agent architectures on standardized benchmarks remain limited.

2.3 WebWalker QA Benchmark

WebWalker [15], accepted at ACL 2025, evaluates LLMs' ability to perform complex web navigation and information retrieval through multi-hop traversal. Unlike WebVoyager [10], Mind2Web [6], and WebArena [18], which focus on end-to-end task completion or web development, WebWalkerQA emphasizes deep information extraction requiring systematic exploration across interconnected web resources. The benchmark spans four real-world domains with questions crafted to require multi-hop reasoning rather than direct single-source retrieval.

2.4 Single vs. Multi-Agent Architectures

Recent comprehensive evaluations challenge assumptions about multi-agent superiority. Gao et al. [8] evaluate single-agent and multi-agent systems across 15 datasets and 9 frameworks, finding that while MAS historically outperformed SAS, *this advantage diminishes substantially with more capable models*—accuracy gains dropped from ∼10% to just 3% when comparing ChatGPT to Gemini-2.0-Flash. MAS incurs substantial efficiency costs: 4-220× more input tokens and 2-12× more output tokens than SAS equivalents.

The study identifies three categories of MAS failure modes: (1) *node-level* defects where critical agents bottleneck performance, (2) *edge-level* defects where downstream agents become overwhelmed by excessive inputs causing ”overthinking” errors, and (3) *path-level* defects where information loss during agent handoffs creates irreversible downstream errors. These systematic failures suggest that naive multi-agent decomposition often introduces coordination overhead without corresponding benefits.

Importantly, Gao et al. [8] propose hybrid approaches as the path forward: *agent routing* that directs simple queries to SAS and complex ones to MAS, and *agent cascade* that routes requests to SAS first and escalates failures to MAS only when needed—achieving up to 12% accuracy improvements while reducing costs by 20%. This suggests the question is not ”single-agent or multi-agent?” but rather ”when should each be deployed?”

Our work contributes to this emerging literature by providing detailed behavioral analysis of coordination dynamics, identifying thrashing behavior where increased activity correlates with worse outcomes, and characterizing operational modes that explain when multi-agent coordination helps versus hurts.

3 Methods

3.1 System Architectures

Our evaluation employs the Claude Agent SDK, a state-of-the-art framework enabling reproducible agent development with Claude 4.5 models. We selected this platform for three key advantages: (1) production robustness; our evaluation records minimal tool call failures despite executing thousands of tool invocations, (2) frontier model access; Claude 4.5 represents the current state-of-the-art, avoiding evaluation pitfalls where research focuses on dated models, (3) compatible baselines; our multi-agent builds on the single-agent baseline by adding a single tool call and minimally tweaking the system prompt, and (4) design reproducibility—Anthropic's published multi-agent research system [1] provides documented implementation patterns including orchestrator-worker coordination and prompt specifications [2]. Our multi-agent architecture follows Anthropic's lead-subagent pattern, with prompts lightly adapted from these published examples. The primary limitation is vendor lock-in: Claude Agent SDK restricts evaluation to Anthropic models, necessitating Claude Haiku 4.5 for both architectures to maintain cost feasibility.

3.1.1 Single-Agent Baseline

The single-agent architecture consists of a single research agent that independently handles the entire question-answering pipeline. The agent is equipped with four core tools: `WebSearch` (for broad web searches), `WebFetch` (for retrieving full page content), `Read` (for accessing any local files), and `Bash` (for system commands, as a default tool). We employ Claude Haiku 4.5 as the base model, configured with a system prompt emphasizing efficient web research and information synthesis. The agent follows an observe-orient-decide-act (OODA) loop, alternating between tool execution and reasoning steps until arriving at a final answer or exceeding the architectural limit of 50 turns per question.

3.1.2 Autonomous Multi-Agent System

The multi-agent architecture implements the orchestrator-worker pattern documented by Anthropic [1], introducing hierarchical coordination through a lead-subagent structure built on the single-agent baseline.

The *Lead Researcher Agent* receives the original question and orchestrates research by creating some number of specialized subagents (average 1.38, range 1-3 in practice) via the `Task` tool. The lead agent analyzes questions to identify research subtasks, formulates clear instructions for each subagent, monitors subagent progress, and synthesizes results into a final answer. **Outside of minor prompt adjustments and the introduction of the `Task` tool, lead researcher agents are implemented in the exact same architecture as the single-agent baseline.**

Subagent Researchers execute specific research tasks delegated by the lead agent. Each subagent operates independently with access to `WebSearch`, `WebFetch`, `Read`, and `Bash` tools, but *without* the `Task` tool (preventing recursive delegation). Subagents are configured with tool budget constraints of 5-20 tool calls per task to encourage focused research. All agents employ Claude Haiku 4.5 to maintain cost parity with the single-agent baseline, with specialized prompts: the lead agent emphasizes task decomposition and synthesis, while subagents emphasize efficient execution within budget constraints and reporting results back to the lead. **Outside of prompt specialization, subagent researchers are implemented in the exact same architecture as the single-agent baseline**.

We intentionally do not constrain the number of subagents or enforce task decomposition. Instead, the lead agent is granted autonomy to determine whether decomposition is beneficial for a given query. This design choice reflects realistic deployment scenarios, where multi-agent systems must self-regulate coordination rather than rely on manual tuning.

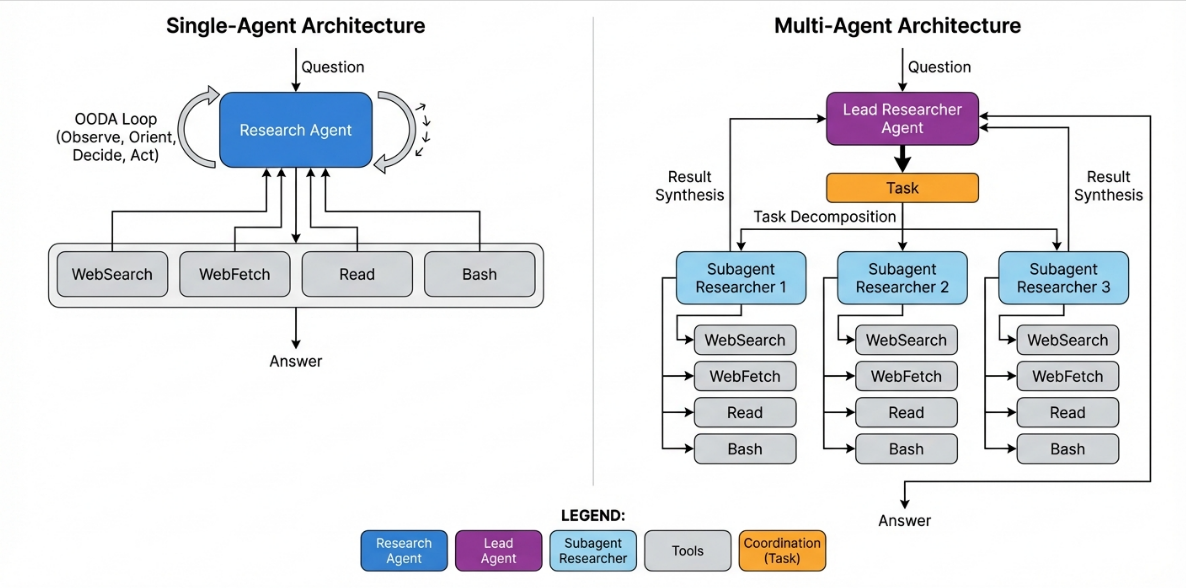

Figure 1: System architecture comparison. (Left) Single-agent system processes queries independently with direct tool access through an iterative OODA loop. (Right) Multi-agent system decomposes tasks through a lead agent that orchestrates specialized subagents, with task decomposition flowing downward and result synthesis flowing back upward.

3.2 Dataset and Evaluation

We evaluate both architectures on WebWalker QA [15], selecting questions from 100 arbitrary indices to ensure diversity across domains and difficulty levels. WebWalker QA was chosen over alternatives such as OpenAI's BrowseComp due to cost feasibility—preliminary testing with BrowseComp indicated per-run costs often exceeding several dollars, rendering large-scale evaluation prohibitively expensive for comparative analysis. WebWalker QA questions require multi-hop reasoning and web navigation, with examples such as: *"What is the population of the capital city of the country that hosted the 2016 Summer Olympics?"* This question necessitates: (1) identifying the 2016 Summer Olympics host country (Brazil), (2) determining its capital (Brasília), and (3) retrieving the current population—requiring sequential web searches and information synthesis.

We conduct 200 total evaluation runs: 100 with the single-agent architecture and 100 with the multi-agent architecture, using identical question sets to enable direct comparison. Runs execute asynchronously with a maximum concurrency of 5 to respect API rate limits while maintaining reasonable throughput. Each run operates in an isolated workspace with a unique temporary directory, ensuring no cross-contamination between evaluations.

Answer grading employs GPT-5-mini as an automated LLM judge, following established practices in agent evaluation [16]. The grader receives the original question, the agent's final answer, and the ground-truth answer, producing two assessments: (1) binary correctness (correct/incorrect), and (2) correctness score (0-10 scale reflecting partial credit). This multi-faceted grading captures nuances beyond binary classification, enabling analysis of partially correct responses and grader agreement patterns.

3.3 Metrics

We extract 26 comprehensive metrics from each evaluation run, organized into five categories:
Performance Metrics: Binary accuracy (correct/incorrect) and correctness score (0-10) capture task success and response quality.
Resource Metrics: Cost (USD, computed from token usage at API pricing rates), execution time (seconds), and total tokens measure computational resource consumption.
Activity Metrics: Number of turns (agent reasoning-action cycles), total messages, total tool calls, and per-tool breakdowns (`WebSearch`, `WebFetch`, `Read`, `Bash`, `Task`) quantify agent behavior and search strategies.
Coordination Metrics (multi-agent only): Number of subagents created, lead agent messages, subagent messages, subagent similarity (cosine similarity of task descriptions using sentence embeddings), individual subagent success rates, subagent completion rates, and temporal clustering patterns (when subagents are created relative to total execution) characterize multi-agent dynamics.
Error Metrics: Total errors encountered during execution (API failures, tool exceptions, timeout events) indicate system robustness.

Metrics are extracted through automated message serialization from the Claude Agent SDK. Each agent conversation is parsed to identify tool use events, extract token counts, compute costs, and analyze coordination patterns. For multi-agent runs, we additionally compute subagent similarity using sentence-transformers (all-MiniLM-L6-v2) to embed task descriptions and calculate pairwise cosine similarities. Subagent success is assessed by submitting individual subagent responses to another GPT-5-mini grader.

3.4 Statistical Analysis

Our analysis employs multiple statistical methods to characterize system behavior and performance. Descriptive statistics (mean, standard deviation, range) summarize distributions across architectures. Pearson correlation analysis identifies relationships between metrics and accuracy, revealing behavioral patterns that predict success or failure.

For predictive modeling, we employ logistic regression to predict binary accuracy from behavioral and coordination metrics, achieving 69% classification accuracy. Linear regression models predict resource consumption (cost and time) from activity metrics, with $R^2 = 0.972$ for cost prediction and $R^2 = 0.758$ for time prediction, demonstrating high predictability of computational resource usage.

To identify distinct operational modes, we apply Principal Component Analysis (PCA) for dimensionality reduction followed by K-means clustering (k=3) on multi-agent runs. PCA retains the first two components explaining 71.1% of variance (PC1: 59.2%, PC2: 11.8%), and clustering reveals three operational patterns differing in coordination intensity and resource usage. Statistical significance testing employs two-sample t-tests with Bonferroni correction for multiple comparisons, and we report effect sizes (Cohen's d) for key differences.

3.5 Data and Code Availability

To ensure reproducibility, all materials from this study have been made publicly available. The complete dataset (200 evaluation runs with 26 metrics per run) is released in CSV format (data/ww100.csv), along with analysis scripts for statistical analyses, figure generation, and clustering. System prompts for the lead agent, subagents, and GPT-5-mini grader are provided in full to enable replication of our exact experimental setup. All materials are available at: https://github.com/BGray-wrl/mas-research-polished-current.

Table 1: Performance comparison: single-agent vs multi-agent on WebWalker QA (N=100 each)

Metric	**Single-Agent**	**Multi-Agent**	**Ratio**
Accuracy (%)	50.0	48.0	0.96×
Cost/Question ($)	0.243 ± 0.192	0.473 ± 0.394	1.94×
Time/Question (s)	88.7 ± 74.0	127.8 ± 76.6	1.44×
Tool Calls	12.9 ± 9.2	22.8 ± 15.8	1.76×
Total Messages	36.4 ± 23.5	49.6 ± 31.7	1.36×
Lead Agent Messages	36.4 ± 23.5	6.82 ± 1.4	0.19×
Cost/Correct ($)	0.49	0.98	2.02×

4 Results

4.1 Overall Performance Comparison

Table 1 summarizes the comparative performance across key metrics. The single-agent architecture achieves 50% accuracy (50/100 correct) compared to 48% (48/100 correct) for the multi-agent architecture. Paired analysis reveals this 2 percentage point difference is not statistically significant: McNemar's test on discordant pairs (11 single-only correct vs. 9 multi-only correct out of 20 discordant pairs) yields $p = 0.82$, and bootstrap 95% CI for the accuracy difference is $[-0.07, 0.11]$, containing zero. We therefore conclude the architectures achieve *comparable accuracy* on this benchmark.

Resource consumption reveals significant differences. The multi-agent architecture incurs 1.94× higher cost per question ($0.473 ± $0.394 vs. $0.243 ± $0.192) and 1.44× longer execution time (127.8s ± 76.6s vs. 88.7s ± 74.0s). The efficiency gap widens when normalized by success: the cost per correct answer for multi-agent reaches $0.98 compared to $0.49 for single-agent—a 2.02× disadvantage. The multi-agent architecture achieves no accuracy improvement while consuming substantially more resources.

Activity patterns reveal an architectural distinction. The multi-agent system executes 1.76× more tool calls (22.8 ± 15.8 vs. 12.9 ± 9.2) and generates 1.36× more messages (49.6 ± 31.7 vs. 36.4 ± 23.5)—increased activity without corresponding performance gains. The lead agent contributes only 6.8 ± 1.4 messages—an 81% reduction compared to the single-agent system—indicating minimal deliberation before delegation. Yet despite this delegation, the multi-agent architecture does not achieve faster execution.

Effect sizes (Cohen's d) confirm the practical significance of these differences: cost per question (d = 0.74, medium-large effect), execution time (d = 0.52, medium effect), and tool calls (d = 0.76, medium-large effect) all demonstrate substantial multi-agent decomposition disadvantages beyond statistical significance.

4.2 Thrashing Behavior

A striking pattern emerges: higher resource usage is *negatively* correlated with success rates. Pearson correlations in multi-agent runs reveal moderate negative relationships between accuracy and activity metrics. Total tool calls ($r = -0.355, p < 0.001$), subagent messages ($r = -0.354, p < 0.001$), websearch calls ($r = -0.335, p < 0.001$), webfetch calls ($r = -0.328, p < 0.001$), cost ($r = -0.328, p < 0.001$), and number of subagents ($r = -0.318, p < 0.01$) all correlate negatively with success. All p-values remain significant after Bonferroni correction ($\alpha = 0.0019$ for 26 metrics).

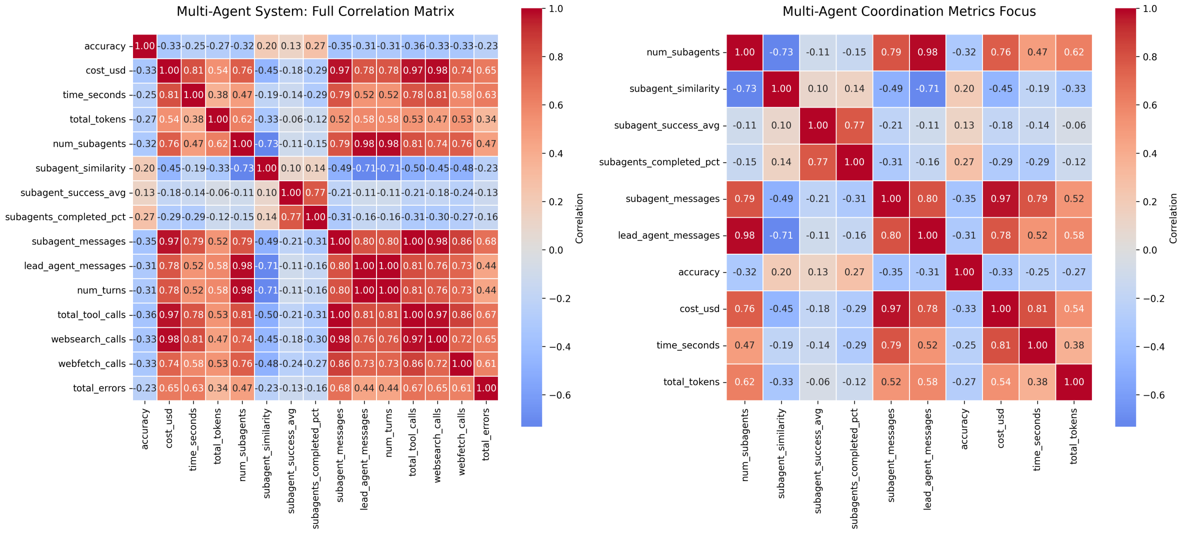

Figure 2: Thrashing behavior revealed through correlation analysis. (Left) Full correlation matrix showing relationships among all metrics. (Right) Focused view of coordination metrics. Notable negative correlations with accuracy include: total tool calls (r=-0.36), cost (r=-0.35), subagent messages (r=-0.35), and websearch calls (r=-0.35). Higher resource usage is negatively correlated with success rates, consistent with thrashing-like behavior or difficulty-driven escalation (or both).

Single-agent runs show the same pattern, though with weaker correlations.

This pattern is consistent with *thrashing*—a term borrowed from systems engineering where overloaded systems spend resources on overhead rather than productive work. In unsuccessful runs, we often see escalating exploration (more searches/agents/cost) without improved outcomes; this may indicate unproductive search dynamics or reflect higher task difficulty. Successful runs tend to be conversely efficient: they find correct answers with focused, minimal tool use. Figure 2 visualizes these relationships, showing that the most accurate runs tend to coincide with fewer tool calls and lower spend in this setting.

We do not interpret thrashing as evidence that coordination itself causes failure; rather, higher coordination and activity appear to be responses to underlying task difficulty, which the current architecture fails to resolve effectively.

4.3 Multi-Agent Coordination & Decomposition

Table 2 presents multi-agent coordination patterns. The subagent distribution reveals minimal decomposition: 71 runs effectively collapsed into a single-subagent system, 20 runs deployed 2 subagents, and just 9 runs deployed 3 (mean: 1.38). The lead agent contributes only 6.8 ± 1.4 messages compared to 42.8 ± 30.5 subagent messages—86% of conversation occurs at the subagent level with minimal lead deliberation.

The reported subagent similarity (0.923 ± 0.149) must be interpreted cautiously: with 71% of runs deploying only a single subagent, similarity is trivially 1.0 in most cases. Among the 29 runs with 2+ subagents where similarity is meaningful, subagents still receive similar instructions (0.73 ± 0.16), but the more striking finding is that the system overwhelmingly chooses minimal decomposition when given autonomy. This suggests that the evaluated MAS self-prunes toward low-coordination regimes, effectively collapsing into a single-agent-like strategy, consistent with our finding that Light mode (minimal coordination) achieves best accuracy. Individual subagents

Table 2: Multi-agent coordination metrics (N=100 multi-agent runs)

Metric	Mean ± Std	Range / Distribution
Num Subagents	1.38 ± 0.65	71 single, 20 double, 9 triple
Lead Agent Messages	6.8 ± 1.4	4–12
Subagent Messages	42.8 ± 30.5	8–180
Subagent Similarity[†]	0.923 ± 0.149	0.45–1.00
Subagent Quality Score (0-10)	5.25 ± 2.87	0–10
Subagent Completion Rate (%)	32.6	20.5 (fail) – 44.8 (success)
Clustering Mean	0.08 ± 0.06	0.02–0.36

[†] Trivially 1.0 for 71 single-subagent runs; meaningful only for 29 multi-subagent runs.

achieve moderate quality scores (5.25/10 average) and completion rates (32.6%) across all 138 subagents. Critically, successful runs show higher subagent completion rates (44.8%) versus failed runs (20.5%), confirming that subagent performance matters when deployed.

Temporal clustering analysis reveals that subagents are created predominantly early in execution (clustering mean: 0.08 ± 0.06 where 0 = start and 1 = end; for runs with subagent count > 1: 0.09 ± 0.08). This front-loaded and minimal decomposition contrasts with adaptive strategies that would iteratively create agents based on intermediate findings.

4.4 Predictive Modeling

4.4.1 Success Prediction

Logistic regression trained on behavioral and coordination metrics achieves 69% accuracy in predicting binary correctness for multi-agent runs. The lone positive predictor is subagent completion percentage (coefficient: +0.326), indicating that when subagents successfully complete delegated tasks, system-level success improves—an unsurprising result given that 71% of runs deployed only one subagent. Negative predictors include activity metrics: webfetch calls (-0.157), number of subagents (-0.154), and total tool calls (-0.152), reinforcing the thrashing pattern. The volume of tool calls was a stronger negative predictor than the error rate (-0.077), implying thrashing is independent of mechanical robustness.

4.4.2 Resource Prediction

Linear regression models demonstrate high predictability of computational resources. Cost prediction achieves $R^2 = 0.972$ with RMSE = \$0.065. The top predictors are websearch calls (coefficient: 0.154), subagent messages (coefficient: 0.140), and total tool calls (coefficient: 0.136). This high R^2 is unsurprising: cost is nearly mechanically determined by token usage and tool calls, which these metrics directly capture. Time prediction achieves $R^2 = 0.758$, with websearch calls (coefficient: 286.8s) as the primary predictor.

These findings have practical implications for budget management: organizations can estimate costs from expected activity levels, enabling early termination of expensive runs. The high cost predictability reflects the deterministic relationship between API calls and billing rather than any deep insight into system behavior.

4.5 Operational Modes

PCA followed by K-means clustering (k=3) on multi-agent runs reveals three distinct operational modes, visualized in Figure 3. The first two principal components explain 71.1% of variance (PC1: 59.2%, PC2: 11.8%), with PC1 capturing resource intensity (tool calls, cost, messages) and PC2 capturing subagent success metrics.

Light Mode (60 runs): The majority operational mode employs minimal multi-agent coordination and observes mode collapse into a single-agent proxy, with exactly 1.0 subagent and 12.9 tool calls—closely resembling single-agent behavior. This mode achieves the *highest* accuracy at 61.7% while maintaining low cost ($0.25) and fast execution (89s). Light mode's dominance (60% of runs) and superior performance suggest that the multi-agent system naturally converges toward single-agent-like behavior when effective.

Medium Mode (28 runs): An intermediate mode with 1.61 subagents and 29.1 tool calls, achieving 32.1% accuracy at moderate cost ($0.59) and time (157s).

Heavy Mode (12 runs): Characterized by intensive multi-agent coordination with 2.75 subagents and 57.2 tool calls per run. This mode incurs high costs ($1.34) and long execution times (254s). Accuracy is strikingly low at just 16.7%, demonstrating that heavy coordination is associated with substantially worse performance.

The clustering analysis reveals a stark pattern: accuracy *correlates negatively* with coordination intensity (Light: 61.7% → Medium: 32.1% → Heavy: 16.7%). While this pattern is consistent with the thrashing interpretation, we note that causality may run in both directions: difficult questions may trigger Heavy mode, and Heavy mode may also degrade performance through coordination overhead. Regardless of causal direction, the pattern reveals that increased coordination does not co-occur with improved outcomes in this architecture.

5 Discussion

5.1 Cause of Multi-Agent Inefficiencies & System Collapse

Our results point to multiple inefficiencies in the evaluated architecture that align with the systematic defects identified by Gao et al. [8].

First, *minimal task decomposition*: 71% of runs effectively collapsed into a single-subagent system, with the remaining 29 creating 2-3 subagents. Rather than partitioning problems across agents, the autonomous multi-agent system naturally converges Webwalker QAs toward single-agent-like behavior. This represents a *node-level defect* where the lead agent under-utilizes available capacity—perhaps recognizing, correctly, that decomposition adds overhead without benefit for many queries.

Second, *insufficient lead deliberation.* The lead agent contributes only 6.8 messages on average compared to 36.4 for the single-agent system. The lead performs minimal reasoning before delegation rather than iteratively refining strategies. Front-loaded subagent creation (clustering mean: 0.08) reinforces this.

Third, *coordination overhead fails to yield parallel gains.* Despite deploying 1.76× more tool calls, the evaluated autonomous multi-agent system does not achieve faster execution (1.44× slower) or better accuracy (2% worse). The result that Light operational mode (minimal coordination) achieves best performance (61.7% accuracy) while Heavy mode achieves worst (16.7%) illustrates how heavier coordination co-occurs with worse outcomes in this implementation. Whether coordination causes degradation or is primarily triggered by harder questions remains unclear.

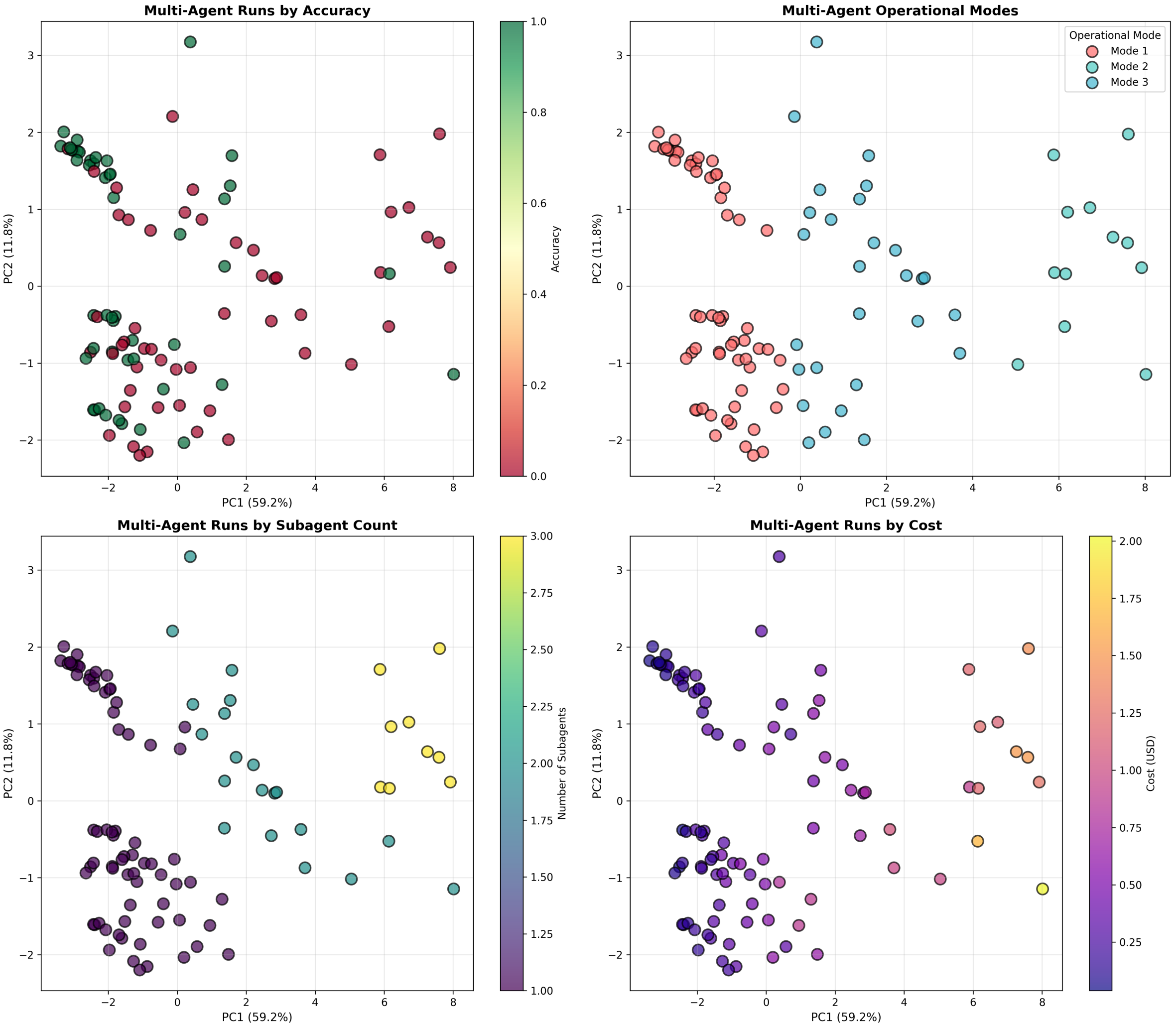

Figure 3: Operational modes identified via PCA and K-means clustering (k=3). PCA components explain 71.1% of variance (PC1: 59.2%, PC2: 11.8%). Three operational modes emerge: Light (60 runs) with minimal coordination achieving best accuracy (61.7%), Medium (28 runs) with moderate coordination (32.1% accuracy), and Heavy (12 runs) with intensive coordination yielding worst performance (16.7% accuracy). Accuracy decreases with coordination intensity, demonstrating that thrashing manifests as distinct operational regimes.

These patterns suggest that this multi-agent architecture may benefit from: (1) more sophisticated task decomposition, (2) increased lead agent deliberation, and (3) coordination protocols that minimize idle overhead. Whether alternative multi-agent designs or evaluation tasks would result in the same failure modes remains an open question.

5.2 Thrashing Explained

Thrashing-like patterns appear when runs do not converge quickly and escalate tool use without accuracy improvements; this escalation may be both a response to difficulty and a contributor to inefficiency. Easy questions get answered quickly with minimal resources, while difficult or ambiguous questions trigger extensive but futile exploration. This suggests optimization techniques may include early termination strategies, budget-aware planning, difficulty-based routing, and a focus on search quality rather than quantity.

5.3 Cost-Benefit Trade-offs and Hybrid Approaches

The cost-benefit analysis is stark. The multi-agent implementation cost \$0.98 per correct answer versus \$0.49 for single-agent—a 2.02× efficiency disadvantage consistent with the 4-220× token overhead documented by Gao et al. [8], with a 44% longer execution time, consuming significant additional computational resources without delivering accuracy improvements.

These findings suggest the answer is not ”single-agent or multi-agent?” but rather *adaptive deployment*. Our operational modes analysis provides empirical support: Light mode (minimal coordination) achieves 61.7% accuracy—dramatically outperforming both Medium mode (32.1%) and Heavy mode (16.7%). Accuracy inversely correlates with coordination intensity. This pattern mirrors the *agent cascade* approach proposed by Gao et al. [8], where systems route requests to single-agent first and escalate to multi-agent only when needed.

For information retrieval tasks like WebWalker QA, single-agent architectures are more cost-effective. But hybrid approaches—routing or decomposing complex queries to multi-agent systems while handling simpler ones via single-agent—may offer the best of both worlds. The predictive models we develop (69% success prediction, R^2=0.972 for cost) could enable such adaptive routing by identifying queries likely to benefit from multi-agent coordination versus those better served by focused single-agent exploration.

5.4 Implications for System Design

The high predictability of cost ($R^2 = 0.972$) reflects that API costs are mechanically determined by token usage and tool calls: useful for budget management but not a meaningful finding about agent behavior. The predictability of accuracy (69%) may enable early identification of struggling runs for adaptive intervention: early termination, human-in-the-loop assistance, or fallback to simpler strategies.

The thrashing pattern suggests that prompt engineering targeting reduced tool calls could yield both cost savings and accuracy improvements. Designing agents with constrained search budgets may be more effective than allowing unlimited exploration.

5.5 Interpretation Boundaries

Our results should be interpreted as a behavioral analysis of one autonomous orchestrator–worker multi-agent design under realistic cost constraints. We do not claim that multi-agent systems are inherently inferior, nor that enforced or manually designed task decomposition cannot yield improvements. Rather, we show that when given autonomy to decide whether and how to decompose WebWalker QA tasks, this MAS overwhelmingly selects minimal coordination (and incurs overhead when it does not)."

5.6 Limitations

Several limitations constrain our study. First, we evaluate on a *single benchmark* (WebWalker QA), limiting conclusions to multi-hop web navigation tasks. Patterns may differ on benchmarks emphasizing different capabilities (e.g., GAIA for general assistance, τ^2-Bench for tool use, BrowseComp for more extensive web interaction).

Second, we employ a *single model family* (Claude Haiku 4.5), chosen for cost-efficiency but potentially masking dynamics observable with more capable models like Claude Opus 4.5 or Gemini 3.0.

Third, our evaluation encompasses *100 questions per condition*—sufficient for statistical significance but expandable for finer-grained analysis.

Fourth, *grader validation* relies solely on GPT-5-mini as an automated judge without robust human validation or inter-rater reliability assessment. While automated grading follows established practices in agent evaluation, the lack of systematic comparison with human annotations limits confidence in absolute accuracy measurements.

Fifth, *prompt design* significantly influences behavior, and our specific prompts for lead and subagents represent one instantiation among many possible approaches. Alternative prompting strategies emphasizing diverse task decomposition or iterative refinement might yield different coordination patterns.

Finally, *cost constraints* prevented comprehensive testing across model tiers, leaving open whether more capable (and expensive) models exhibit different trade-offs.

Questions remain: Would the thrashing pattern persist across domains? Would stronger models demonstrate genuine multi-agent advantages? Are our findings specific to information retrieval or applicable to broader agent tasks? Future work should address these questions through cross-dataset evaluation and model ablation studies.

5.7 Future Work

The most promising direction is *hybrid architectures* that adaptively select between single-agent and multi-agent approaches. Following Gao et al. [8], agent routing and cascade strategies could leverage our predictive models to identify queries likely to benefit from multi-agent coordination. The correlation between early behavioral signals and outcomes (69% prediction accuracy) suggests that adaptive escalation—starting with single-agent and escalating to multi-agent only when struggling—could capture benefits of both approaches.

Within autonomous multi-agent systems, addressing identified deficiencies requires: (1) *improved task decomposition* generating diverse subtasks rather than single-agent collapse, (2) *increased lead deliberation* enabling monitoring and replanning, and (3) *coordination protocols that minimize idle overhead.*

Cross-dataset evaluation should test generalizability across diverse benchmarks (GAIA, τ^2-Bench, BrowseComp) and domains. *Model comparisons* with more capable models would clarify whether autonomous multi-agent advantages emerge with increased base capability.

6 Conclusion

This work presents a systematic empirical comparison of a single-agent and an autonomous multi-agent decomposition system for deep research on the WebWalker QA benchmark using Claude Haiku 4.5. Through 200 evaluation runs extracting 26 comprehensive metrics, we find that the single-agent architecture achieves comparable accuracy (50% vs. 48%) while achieving 2.02× better cost efficiency ($0.49 vs. $0.98 per correct answer) and 44% faster execution. These results challenge assumptions that multi-agent coordination inherently improves performance on complex information retrieval tasks, though we note these conclusions apply to autonomous decomposition under the evaluated architecture and task and should not be generalized to manually structured or out-of-domain multi-agent systems.

Our analysis identifies a thrashing pattern where higher resource usage correlates with lower success rates (r = -0.335 to -0.355 for key metrics), suggesting that increased activity often accompanies unsuccessful runs; this may reflect unproductive exploration, higher difficulty, or both. The evaluated autonomous multi-agent system exhibits several inefficiencies: minimal task decomposition (71% of runs effectively collapsed into a single-subagent system), minimal lead agent deliberation (6.8 vs. 36.4 messages), front-loaded coordination, and coordination overhead that does not result in execution benefits. Clustering analysis reveals that that intensive coordination in this autonomous MAS is associated with substantially worse outcomes on WebWalker QA: Light mode (minimal coordination) achieves 61.7% accuracy, Medium mode 32.1%, and Heavy mode just 16.7%.

Cost is nearly perfectly determined by activity metrics ($R^2 = 0.972$), enabling accurate budget forecasting, while success prediction achieves 69% accuracy. These findings have practical implications: organizations deploying LLM-based agents should carefully evaluate whether multi-agent complexity is justified by measurable performance gains rather than assuming architectural sophistication translates to better outcomes.

Our findings align with concurrent work showing that multi-agent advantages diminish with capable models and incur substantial overhead [8], finding no meaningful advantages in the autonomous multi-agent vs single agent system. Our operational modes analysis—where Light mode (61.7% accuracy) dramatically outperforms Heavy mode (16.7%)—provides empirical support for hybrid approaches like agent cascade [8] that route to single-agent first and escalate to multi-agent only when needed. The predictive models we develop could enable such adaptive routing.

For the task and architecture evaluated, the single-agent baseline offers superior cost-effectiveness. But the path forward likely involves hybrid systems that selectively apply multi-agent coordination where genuinely beneficial, and further research evaluating alternative multi-agent designs across diverse benchmarks.

References

[1] Anthropic. How we built our multi-agent research system. https://www.anthropic.com/engineering/multi-agent-research-system, June 2025. Technical blog post on orchestrator-worker pattern for multi-agent research.

[2] Anthropic. Claude cookbooks: Agent prompts. https://github.com/anthropics/claude-cookbooks/tree/7431b7e98d64a9ee34844f59d15fe6902f8fcbe9/patterns/agents/prompts, 2025. Published prompt templates for lead-subagent coordination patterns.

[3] Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench:

Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982. Benchmark for dual-control conversational AI agent evaluation.

[4] Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail?, 2025. URL https://arxiv.org/abs/2503.13657. Empirical analysis of multi-agent system failure modes.

[5] Lingjiao Chen, Jared Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. Are more llm calls all you need? towards scaling laws of compound inference systems. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2403.02419. Analysis of how multiple LM calls affect compound system performance.

[6] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*, 2023. Generalist web agent development.

[7] Chen Gao, Xiaochong Lan, Nian Li, Yuan Yuan, Jingtao Ding, Zhilun Zhou, Fengli Xu, and Yong Li. Large language models empowered agent-based modeling and simulation. *Nature Humanities and Social Sciences Communications*, 2024. doi: 10.1038/s41599-024-03611-3. Survey of LLM integration into agent-based modeling.

[8] Mingyan Gao, Yanzi Li, Banruo Liu, Yifan Yu, Phillip Wang, Ching-Yu Lin, and Fan Lai. Single-agent or multi-agent systems? why not both? *arXiv preprint arXiv:2505.18286*, 2025. URL https://arxiv.org/abs/2505.18286. Empirical comparison of single-agent and multi-agent LLM systems showing performance tradeoffs as model capabilities improve.

[9] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI)*, 2024. doi: 10.24963/ijcai.2024/890. Agent orchestration for managing complexity.

[10] Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models, 2024. End-to-end web agent evaluation.

[11] Cristian Jimenez-Romero, Alper Yegenoglu, and Christian Blum. Multi-agent systems powered by large language models. *Frontiers in Artificial Intelligence*, 2025. doi: 10.3389/frai.2025.1593017. Toolchain integrating LLMs with NetLogo for swarm intelligence.

[12] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic evaluation of language models, 2023. Holistic evaluation with robustness testing.

[13] Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn llm agents. In *OpenReview.net*, 2024. Progress rate metric for multi-turn agent evaluation.

[14] Mahmoud Mohammadi, Yipeng Li, Jane Lo, and Wendy Yip. Evaluation and benchmarking of llm agents: A survey. https://arxiv.org/html/2507.21504v1, 2024. Comprehensive survey of LLM agent evaluation.

[15] Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and Fei Huang. Webwalker: Benchmarking llms in web traversal, 2025. URL https://arxiv.org/abs/2501.07572. WebWalker QA benchmark for web navigation and multi-hop information retrieval.

[16] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

[17] Junting Zhou, Wang Li, Yiyan Liao, Nengyuan Zhang, Tingjia Miao, Zhihui Qi, Yuhan Wu, and Tong Yang. Scholarsearch: Benchmarking scholar searching ability of llms, 2025. Browsing capabilities enhance LLM performance on academic searches.

[18] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *Proceedings of the International Conference on Learning Representations*, 2024. Realistic web environment for agent evaluation.