PAY FOR THE FAULT, NOT THE FLOW: LABEL-FREE IN-FLOW MULTI-AGENT WORKFLOW OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) increasingly construct multi-agent workflows that decompose a complex task and assign specialist agents from a pool. However, building such a workflow well remains challenging: how finely to divide the task, which agent to trust with each subtask, and when to create a new specialist are all critical decisions a workflow constructor needs to settle up front. Thus, whether each subtask succeeds remains unknown until the workflow runs. Yet, improving a workflow is costly. Locating a fault usually requires a reference answer, a graded outcome, or a trained assessor, and the fix is applied to the whole workflow through re-execution, re-search, or retraining. We propose INFLOWOP, which prices every decision in one label-free cost that weighs how well an agent's competence meets what a subtask demands against how much that agent takes to run. Before execution, INFLOWOP bidirectionally determines the granularity of task decomposition and agent assignment following from the cost rather than from a fixed template. During execution, INFLOWOP corrects a fault with the cheapest move via the same cost that serves the workflow both as it is built and as it runs. Facing the workflow-level evaluation challenge, we introduce BRAID, a benchmark whose tasks require multi-agent coordination beyond single-agent capability. Across various domains and backbones, INFLOWOP outperforms single agent baselines by up to $+11.97\%$, achieving $+9.64\%$ with in-flow optimization.

1 INTRODUCTION

Large language models have grown from single-turn responders into agents that plan, call external tools, and interact with their environment (Schick et al., 2023; Wang et al., 2023a; Shen et al., 2023). More recently, they begin to organize one another (Hong et al., 2024; Zhuge et al., 2024). Given a task and a pool of available specialist agents, an agent now writes the division of work itself, deciding which agent takes each part and in what order the parts run (Zhang et al., 2025b; Zhou et al., 2026; Yue et al., 2026). What such a flow contributes is structure: *the work is divided explicitly, each part goes to an agent suited to it, and progress accumulates step by step rather than resting on one long attempt.*

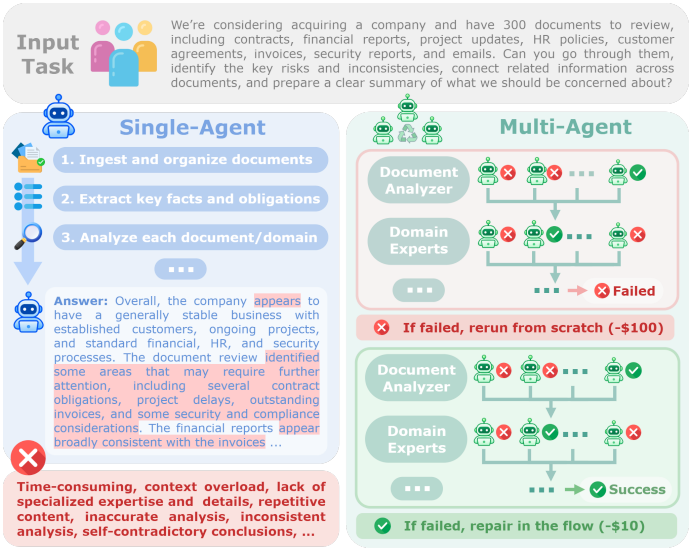

Figure 1: **INFLOWOP for In-Flow Optimization.** Complex tasks challenge single agents with long-context reasoning, while multi-agent systems introduce execution failures and coordination overhead. INFLOWOP optimize agent behaviors in-flow, enabling efficient collaboration at lower cost.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Yet the decisions such an agent makes are narrower than they appear. How finely the task is divided is fixed in advance, such as by a template, a prompt, or a chosen number of steps. Where it does adapt, it adapts to the task alone (Liu et al., 2025; Madhwal et al., 2026; Su et al., 2026). What it would cost the available agents to carry out one division rather than another seldom enters. Where nothing available fits, the work is either forced onto another agent or a new one is created, and neither choice is priced against the other (Zhang et al., 2025c; Wu et al., 2025; Shang et al., 2025). Once the flow runs, the evidence execution produces goes unused: identifying which step goes wrong is treated as needing a reference answer, a graded outcome, or a trained assessor. The first two do not exist while the flow is running, and the third requires supervision the task alone cannot provide. Methods that dispense with all three optimize only once the run has finished, by replaying the completed trace to locate the failure or rerunning the whole flow for a better answer (Chae et al., 2025; Lee et al., 2026a; Lin et al., 2026). Meanwhile, when something goes wrong, the unit of fix is the whole flow, *e.g.*, re-executed, re-searched, or its constructor re-trained. Thus, improvement is priced at the size of the workflow rather than of the fault, and every step that already succeeds is paid for twice (Hwang et al., 2026; Li et al., 2026; Dang et al., 2025). Together, these leave both challenges of §2 open: *building a workflow well under uncertainty and improving one at lower cost*. Worse still, neither is put at risk: *the tasks these systems are measured on are largely ones a single competent agent already handles* (Zhou et al., 2026; Sun et al., 2026; Choi et al., 2025).

INFLOWOP answers these in two stages (§3), pricing every choice in one currency that needs no labels: *how well an agent's competence meets what a part of the work demands, weighed against how much that agent takes to run*. In the first stage, COALESCE decomposes the task to its finest parts and find the proper granularity through coalescing (§3.1). In the second stage, INFLOWOP performs *in-flow dynamic optimization* by turning the same currency on the flow as it runs, identifying and correcting subtasks locally without losing the execution progress (§3.2). One label-free cost thus serves the workflow both as it is built and as it runs. To sum up, our main contributions are:

1. **COALESCE: the decomposition granularity is priced, not presumed.** A cost-driven bidirectional workflow construction approach (Alg. 3) over the agents on hand settles how finely the task is divided, enabling agent creation only when no available agent clears a competence bar (§3.1).
2. **INFLOWOP: optimization in flow, without supervision.** We propose INFLOWOP (Alg. 1), a two-stage training-free framework that leverages COALESCE to build workflows and optimizes them in-flow (Alg. 4), without any reference answer, graded outcome, or trained assessor (§3.2).
3. **BRAID: a benchmark for reasoning over agent-interdependent divisions.** Workflow-level tasks are those a single agent cannot solve effectively (§C). This is the premise that motivates workflows in the first place, yet one that existing benchmarks seldom satisfy (§C.1). We introduce BRAID (§4), a workflow-level benchmark that requires multi-agent coordinations beyond single-agent competence.
4. **Evaluation: gains across domains and backbones.** Our evaluation across eight domains, six backbones, and four workflow constructors shows that INFLOWOP improves over a single agent from $+7.15\%$ to $+11.97\%$ under a matched turn budget, and outperforms workflows baselines by up to $+9.64\%$ (§5).

2 PRELIMINARIES

When tackling complex tasks that cannot be resolved by a single agent, we build multi-agent workflows:

The problem. A task Q is addressed using a pool of specialist agents $\mathcal{A} = \{a_1, \ldots, a_m\}$. An LLM-powered workflow constructor $\mathcal{D}$ produces a workflow $W = \mathcal{D}(Q, \mathcal{A})$ that orchestrates agents from $\mathcal{A}$. The workflow W is then executed to evaluate its performance $\mathrm{Perf}(W) \in [0, 1]$ against the reference answer of Q. The objective is to maximize Perf at minimal cost via the flow $\mathcal{D} \to W \to \mathrm{Perf}$. However, this is affected by (1) $\mathcal{D} \to W$: how W is *constructed*, and (2) $W \to \mathrm{Perf}$: how Perf is *optimized* (§3).

Problem Formulation

Pool-Aware Workflow Construction: Given a task and a pool of specialist agents, automatically assemble a multi-agent workflow that solves the task both *effectively* and *efficiently*.

1. ***Effective to solve* (*reliability*).** A workflow is expected to reliably produce the correct final answer. When composing many agents, its reliability is bounded by each agent's progress.
2. ***Efficient to run* (*execution*).** A more complex task calls for a more complex workflow, accordingly more subtasks, agents, and latency to run. So the complexity of a workflow should be matched to the task, spending no more than the task demands.
3. ***Efficient to improve* (*optimization*).** When the workflow falls short, improving it should be cheap, without the time and compute of full re-execution, re-search, or model training.

Building a Workflow under Uncertainty. Specialist agents in the pool differ markedly in competence, thus binding a subtask to an ill-suited agent is a primary source of failure (Liu et al., 2025; Madhwal et al., 2026; Su et al., 2026). A constructor $\mathcal{D}$ that composes agents without regard to their fit therefore produces unreliable workflows. On the other hand, the agent pool is also finite and may hold no agent well-suited to its assigned work. This induces new agent *creation* rather than forcing a poor fit, while at the added cost of specifying and instantiating it (Zhang et al., 2025c; Wu et al., 2025; Shang et al., 2025). Nevertheless, even a well-matched agent is not guaranteed to succeed: whether it completes its assigned work correctly is *uncertain*, and that uncertainty surfaces only once the workflow runs, which is why fault attribution is studied on completed executions (Chae et al., 2025; Lee et al., 2026a; Lin et al., 2026). Execution alone is not conclusive either, as a step that satisfies what it declares can still mislead the steps that follow (§D.5). Collectively, these raise the first core challenge of workflow construction: *how to build a good workflow under uncertainty?*

The Cost of Workflow Optimization. Once a workflow runs, execution finally reveals which steps falter. However, acting on such evidence is expensive. The naive remedy is to re-execute the workflow or re-search over a set of candidate workflows, while discarding everything already computed and progressed (Hwang et al., 2026; Li et al., 2025b; Yoon et al., 2026). An increment of quality thereby charges additional full execution (Fig. 9). Worse still, it presumes a supervision signal absent at inference: no ground-truth answer is available to say *what* went wrong or *where* to improve. On the other hand, a more targeted way is post-training, guiding $\mathcal{D}$ to learn on success cases via supervised finetuning or improve via reward signals through reinforcement learning (Li et al., 2026; Peng et al., 2026; Nielsen et al., 2026; Dang et al., 2025). Whereas the time and computation cost of data preparation and model training can go far beyond test-time re-execution. These studies reveal the second core challenge on workflow optimization: *how to improve a workflow at lower cost?*

3 INFLOWOP: THE FLOW AS BUILT, THE FLOW AS RUN

How to build a reliable workflow, and optimize it at low cost (§2)? Rather than relying on costly training or sampling, we cast both questions as minimizing a single measurable, annotation-free *surrogate*: the workflow's *cost*, which couples its reliability of success with its latency to run. As such, we minimize this cost in two stages (§D.1): (1) *decomposition-aware workflow construction* (§3.1) constructs a low-cost workflow before execution, and (2) *in-flow dynamic optimization* (§3.2) *tempers* it adaptively during execution.

3.1 BIDIRECTIONAL DECOMPOSITION-AWARE WORKFLOW CONSTRUCTION

Decomposition. A complex task seldom fits any single agent. Instead, its parts demand different competences that no one agent commands (§2). Therefore, we *decompose* the task into subtasks, each narrow enough to hand to a well-fit agent and, being narrower, more reliably completed. A workflow is thus a triple $W = \langle S, A, G \rangle$, where $S = \{s_1, \ldots, s_{n'}\}$ is a set of subtasks, A denotes an assignment binding a subtask s_i to an agent $a(s_i)$, and G represents a dependency graph, which is a directed acyclic graph (DAG) with edges

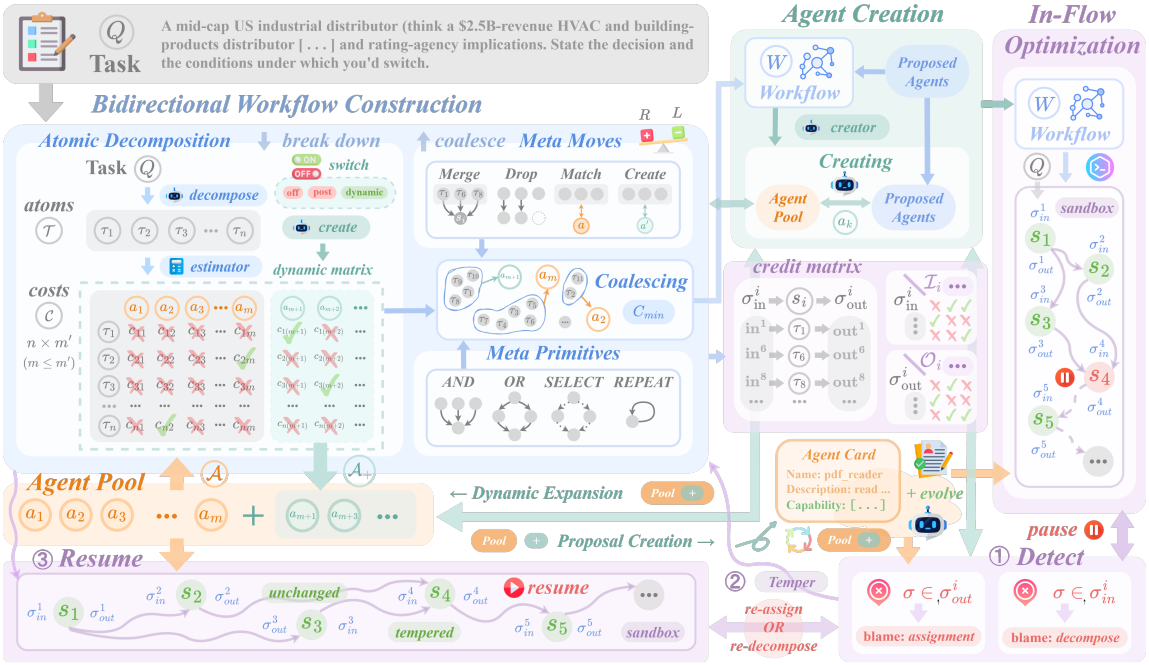

Figure 2: **Bidirectional Workflow Construction and In-Flow Optimization.** Our *bidirectional decomposition-aware workflow construction* (§3.1) builds a workflow W at the proper granularity with minimal cost C_{min} in two directions: *top-down decomposition* and *bottom-up coalescing* (Alg. 3). *In-flow dynamic optimization* (§3.2, Alg. 4) then optimizes W by retrospectively applying the cost matrix to construct the credit matrix, fixing faulty intermediate steps in flow without reference answer. The constructed workflow is executed in the *sandbox*, where INFLOWOP optimizes it as the execution proceeds (Alg. 1).

linking subtasks' nodes. Thus, the *critical path* of G, *i.e.*, its longest chain of dependent subtasks, sets the workflow's latency. Consequently, the right granularity is a question of *cost*: finer decomposition narrows each subtask and raises per-subtask reliability, but multiplies agents to run with increased latency cost and the points at which the workflow may fail with increased reliability cost. This motivates our COALESCE: *bidirectional cost-driven atomic coalescing* (Alg. 3).

Taxonomy of Cost

With various kinds of costs incurred across each workflow's construction, execution, and optimization stages (§2), we organize them into one systematic cost taxonomy (§D.3):

1. ***Reliability cost*:** R (Eq. 16). How much a workflow risks failing, *i.e.*, the price of entrusting each subtask to an agent that may not realize it correctly.
2. ***Latency cost*:** L (Eq. 17). How much a workflow spends to run, *i.e.*, the time and compute its execution consumes.
3. ***Creation cost*:** $|\mathcal{A}_+|$ (Eq. 18). The price of creating a new agent when the pool holds none that fits, under the uncertainty of its true competence.
4. ***Workflow cost*:** $\mathrm{Cost}(W)$ (Eq. 18). The overall cost of a *constructed* workflow, jointly balancing reliability, latency, and creation, exactly what our construction method aims to minimize (Eq. 2).
5. ***Optimization cost*.** The price of improving a workflow, *e.g.*, the effort of re-execution, re-search, re-sampling, model training, *etc.*, separate from the workflow's own cost.

Bidirectional Cost-Driven Atomic Coalescing

Rather than fixed granularity, we let *cost* steer the appropriate granularity in two directions:

1. ***Top-down atomic decomposition*:** the task Q is decomposed into its finest atoms $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$, each individually reliable.
2. ***Bottom-up cost-driven coalescing*:** guided by *cost matrix* (§3.1), these atoms are *coalesced* into $n' \leq n$ subtasks S, each bound to its lowest-cost agent, either existing in pool or newly created, merging only where a coarser subtask lowers the overall cost.

Granularity is therefore an *outcome* of cost-driven coalescing rather than a fixed choice, adapting to each task's requirements and complexity (COALESCE, Alg. 3).

Cost matrix. Scoring Eq. 18 is based on the reliability cost $C(\tau, a)$ of realizing an atom τ with an agent a. We compute it *without labels* by *matching* each agent's declared competence and self-evolving empirical profile against what that atom demands, via a rubric semantic estimator that scores how well a supplied text satisfies a demanded one: $\hat{p}(\tau, a) = \mathrm{match}(\textit{demand}, \textit{supply}) = \mathrm{match}(\tau, a) \in [0, 1]$. Running over all atoms and agents, it constructs the *cost matrix*, with rows as atoms and columns as agents. An agent *covers* an atom when its cost clears a competence bar $\bar{C}$, so the matrix also exposes where the pool falls short:

$$\rho(\tau) = \mathbb{1}\big[\min_{a \in \mathcal{A}} C(\tau, a) > \bar{C}\big] \tag{1}$$

the *coverage residual* of τ: $\rho(\tau) = 1$ marks an atom no pooled agent can be trusted with, which triggers new agent creation at penalty γ. With *dynamic matrix* enabled, each created agent joins the matrix as a new column that every atom can be matched to. As such, we obtain the *cost matrix* with latencies $\ell(\tau, a)$ to guide COALESCE in finding proper granularity for constructing better workflows (§3.1).

Cost-driven coalescing. Not every atom is needed: we keep only $\nu \subseteq \mathcal{T}$, those from which the goal is reachable over $G_{\mathcal{T}}$, and discard the rest (Alg. 3). COALESCE then proceeds over *partial coalescings* P, each placing a prefix of ν in topological order. Extending P by its next atom τ_i admits two moves: OPEN(τ_i) starts a new subtask, and MERGE(τ_i, s) folds τ_i into an open subtask s, so a workflow is compacted only where merging pays. A partial coalescing is dropped once it can no longer improve on the best complete one found, so the optimum always survives (§D.1). COALESCE returns the least-cost coalescing $P^\star$ within B expansions, and LIFT then carries the atom dependencies $G_{\mathcal{T}}$ to the subtasks of $P^\star$, inducing the subtask graph G.

Objective. Decomposition-aware construction thus reduces to one optimization over decompositions S, assignments A, and the dependency graph G they induce, for the least-cost workflow:

$$W^\star = \arg\min_{\langle S, A, G\rangle} \mathrm{Cost}(W) = \arg\min_{\langle S, A, G\rangle} R(W) + \beta\, L(W) + \gamma\, |\mathcal{A}_+(W)| \tag{2}$$

where G is the dependency graph induced by the decomposition S, and $\mathcal{A}_+(W)$ denotes the agents created for W. COALESCE (Alg. 3) solves it through bidirectional cost-driven atomic coalescing, scoring every candidate pairing via cost matrix and atom coalescing to descend $\mathrm{Cost}(W)$. This cost prices structures COALESCE emits via meta primitives (§D.2).

3.2 IN-FLOW DYNAMIC OPTIMIZATION

A constructed workflow is only a *prediction* of what is expected to work (§2). Execution is therefore the first place a workflow's real weaknesses become visible, while also potentially expensive to act on if acting means re-running or re-searching the workflow in full (§2). Instead, we optimize *in flow*: a fault is *caught* as it arises, execution *pauses* at the subtask it arises in, the workflow is *tempered* there locally, and execution *resumes* over everything already achieved.

In-Flow Dynamic Optimization

Execution proceeds subtask by subtask. Whenever a realized subtask is found faulty, we *temper* the workflow there, and execution resumes, rather than restarting it:

1. ***Detect*:** match each realized subtask against the contract its decomposition declared, using only what execution makes observable *without* reference (§3.2).
2. ***Pause*:** halt at the subtask that falters, not at the workflow it belongs to, so every progress already achieved stays intact (Alg. 1).
3. ***Temper*:** correct the fault along a ladder of moves ordered by *cost matrix* (§3.1), committing a change only once it demonstrably holds (§D.4).
4. ***Resume*:** re-run only the tempered subtask, retaining every result already computed (§D.4).

Detection. COALESCE does not merely coalesce atoms into subtasks. Instead, it establishes each subtask with the *contract* its atoms declare, containing the input conditions σ^i_{in} that s_i requires, and the output conditions σ^i_{out} it is expected to deliver. This contract is what makes a fault observable *without* supervision by stating what *should* hold without knowing the answer. Detection then reuses *cost matrix* (§3.1) and applies retrospectively: where construction matches an atom's demands against an agent's card, in-flow detection matches each declared condition against the work actually realized, the observed input $\mathcal{I}_i$ and output $\mathcal{O}_i$ of s_i:

$$u(\sigma\,, x) = 1 - \text{match}(\sigma\,, x) \in [0, 1], \qquad \sigma \text{ is } \textit{unmet} \text{ iff } u(\sigma\,, x) \geq \bar{u} \tag{3}$$

with $x = \mathcal{I}_i$ for input conditions and $x = \mathcal{O}_i$ for output ones. The resulting *credit matrix* is the retrospective counterpart of *cost matrix*. Which side breaches *blames* the fault, and the blame is what a correction respects:

$$\text{blame}(s_i) = \begin{cases} \textit{decomposition}, & \text{if some } \sigma \in \sigma^i_{\text{in}} \text{ is unmet} \\ \textit{assignment}, & \text{else if some } \sigma \in \sigma^i_{\text{out}} \text{ is unmet} \\ \textit{none}, & \text{otherwise} \end{cases} \tag{4}$$

An unmet *output* condition means the subtask is well posed but its agent underdelivered. An unmet *input* condition means the subtask is not given what it needs, which no choice of agent can undo but COALESCE. Alongside this *semantic* signal, we also retain a *causal* signal: a subtask that yields no usable output is faulty by observation and needs no matching.

Objective. In-flow optimization descends the same cost from the constructed workflow $W^\star$ (Eq. 2) under evidence:

$$W^\dagger = \arg\min_{W' \in \mathcal{N}^\star(W^\star)} \widetilde{\text{Cost}}(W'), \qquad \tilde{C}(s, a) = \begin{cases} \infty, & (s, a) \in \mathcal{X} \\ C(s, a), & \text{otherwise} \end{cases} \tag{5}$$

where $\mathcal{N}^\star(W^\star)$ is the set of workflows reachable from $W^\star$ by ladder moves (Eq. 19), each confined to a closure $I(s_i)$ (Eq. 21); $\mathcal{X}$ collects the pairings observed to breach their contracts (Eq. 20); and $\widetilde{\text{Cost}}$ is Eq. 18 evaluated with $\tilde{C}$ in place of C.

4 BRAID: BENCHMARK REASONING OVER AGENT-INTERDEPENDENT DIVISIONS

A workflow earns its necessity only where a single agent cannot handle the task alone, yet workflows are largely measured on suites one competent agent already solves (§C.1). BRAID closes this gap by composing each task at different *complexity levels* k via two composition methods: *distributed* composition gives every question its own gold source, while *anchored* composition shares one gold source across all provides sources. We enforce four quality-control criteria, and a task failing any of them is never written (§C.2). Built on 10 public datasets, BRAID spans 19 evaluation-only arms across 8 domains (Tab. 3).

Table 1: **Main Results.** Mean accuracy (%) per domain on BRAID, averaged over the arms each domain holds and over both complexity levels (§4). *All* (%) averages over the 16 arms of these five domains.

Method	Doc	Fin	Chart	Math	Phys	All	Δ_{acc}
Baseline: Single LLM (*no tools or skills*)							
Qwen3.5-4B	5.64	0.52	2.15	0.00	0.00	1.74	–
Qwen3.5-9B	5.95	1.64	4.58	3.31	0.54	3.23	–
GPT-5-mini	4.02	3.34	15.75	10.05	11.79	8.85	–
GPT-5.4-mini	9.67	1.97	17.12	12.32	10.82	10.59	–
GPT-5.4	10.43	2.54	14.07	42.73	13.02	18.62	–
GPT-5.6-luna	12.98	3.87	25.20	66.66	18.84	28.25	–
Baseline: Single Agent (*with full pool of tools and skills*)							
Qwen3.5-4B	12.51	4.83	22.93	23.48	4.78	13.66	**+11.92**
Qwen3.5-9B	11.26	20.27	25.88	30.38	4.81	17.38	**+14.15**
GPT-5-mini	23.02	3.54	22.83	19.13	10.00	16.33	**+7.48**
GPT-5.4-mini	25.44	1.38	24.91	21.02	10.38	17.49	**+6.90**
GPT-5.4	33.68	25.67	42.17	35.56	10.52	27.21	**+8.59**
GPT-5.6-luna	41.11	49.03	58.36	58.20	14.94	41.99	**+13.74**
INFLOWOP (*ours*)							
Qwen3.5-4B	14.91	12.87	32.45	38.48	7.17	20.81	**+19.07**
Qwen3.5-9B	14.81	25.20	41.28	42.65	9.80	25.13	**+21.90**
GPT-5-mini	26.71	19.23	37.67	40.42	12.44	27.01	**+18.16**
GPT-5.4-mini	28.96	21.14	40.55	43.09	14.93	29.46	**+18.87**
GPT-5.4	45.24	41.41	55.61	50.52	14.85	38.52	**+19.90**
GPT-5.6-luna	42.22	56.95	70.23	74.33	17.34	49.37	**+21.12**

5 EXPERIMENTS

5.1 EXPERIMENT SETUP

Setup. We compare six systems under the same pool of agents. (1) *Single LLM* answers a task in one model call, without any tools or skills[1]. (2) *Single agent* answers with the full tools and skills provided. The remaining four construct workflows: (3) *greedy-search* and (4) COALESCE build workflows respectively, and our comparison between them isolates what the decomposition alone contributes. (5) COALESCE + *in-flow* adds the tempering of §3.2, and (6) the full method further scores each agent on the self-evolving profile, rather than on its card alone.

Models. We evaluate two open-source models, `Qwen3.5` (`4B` and `9B`) (Team, 2026), together with four close-source models, `GPT-5-mini` (OpenAI, 2025), `GPT-5.4-mini` (OpenAI, 2026b), `GPT-5.4` (OpenAI, 2026a), `GPT-5.6-luna` (OpenAI, 2026c).

Data. We evaluate on BRAID (§4), covering 19 evaluation arms over 14 BRAID datasets adapted from 10 public datasets across 8 domains (Tab. 3). Each is constructed at two complexity levels and two compositions a dataset supports[2].

[1]The single-LLM baseline scores near zero on several domains because every input is provided as a file: without tools or skills, it cannot access them, so only answers from what it already knows.

[2]Due to high API cost, we evaluate `GPT-5.4-mini`, `GPT-5.4`, and `GPT-5.6-luna` on the 5-domain subset of BRAID (Tab. 1). Fig. 6 gives all eight domains for the three models evaluated on the whole benchmark.

Table 2: **What Each Module Contributes.** Mean accuracy (%) of `Qwen3.5-9B` on BRAID. INFLOWOP is assembled one module at a time in (2)-(4): (1) a workflow built greedily, (2) a workflow built with COALESCE, (3) the same COALESCE-built workflow with in-flow tempering, and (4) the full method (§3, §D.1, Alg. 1).

Method	Doc	Fin	Chart	Math	Phys	All	Δ_{acc}
Single Agent	11.26	20.27	25.88	30.38	4.81	17.38	–
greedy-search	8.85	13.46	17.63	33.00	4.55	15.49	**-1.89**
COALESCE	12.19	21.25	38.55	39.15	7.61	22.21	**+4.83**
COALESCE + in-flow	13.47	22.75	39.99	40.86	9.48	23.80	**+6.42**
INFLOWOP	14.81	25.20	41.28	42.65	9.80	25.13	**+7.75**

5.2 WHAT THE FLOW EARNS AND WHAT EARNS THE FLOW

Consistent gains across domains and backbones. INFLOWOP improves every backbone over both single-LLM and single-agent baselines (Tab. 1), achieving gains up to $+21.90\%$ over the single LLM baseline, and up to $+11.97\%$ over the single agent baseline with matched compute budget (§E). What the two baselines earn depends on the backbone, as the single agent gains from $+6.90\%$ to $+14.15\%$ over the single LLM. On the other hand, what INFLOWOP earns varies by $\leq 3.74\%$ across six backbones. The gain also exceeds what a larger backbone delivers: under INFLOWOP, `Qwen3.5-4B` reaches 20.81%, above the single-agent accuracy of `Qwen3.5-9B`, `GPT-5-mini`, and `GPT-5.4-mini`. Meanwhile, the strongest backbone `GPT-5.6-luna` also improves from 41.99% to 49.37%. Across domains, the gain is largest on charts, mathematics, and finance, while smallest on physics, where all methods stay $\leq 17.34\%$ under any backbone. Also, the same improvement trends hold consistently on slides, science, and code, the three additional domains for the three backbones evaluated on the whole benchmark (Fig. 6).

Gains come not from dividing the task, but from how it is divided. Comparing against the single agent baseline evaluated on the same BRAID arms (Tab. 2), a workflow constructed greedily obtains 15.49%, falling below the single-agent's 17.38%. Therefore, dividing a task is not by itself what earns the gain. Deciding the division by cost reverses that: COALESCE reaches 22.21%, $+6.72\%$ above greedy search and $+4.83\%$ above the single-agent baseline. In-flow tempering adds a further $+1.59\%$, and scoring each agent on the evolving profile improves $+1.33\%$ more. Each module therefore earns its place, and the two (3)-(4) optimizing in-flow earn less than the one deciding the workflow before it runs.

5.3 WHAT IMPROVES THE FLOW AND WHAT WEIGHS ON IT

Cost is better computed than judged. INFLOWOP prices the cost matrix with a rubric estimator to match atom demands against agent capabilities (§3.1). This cost matrix can also be priced by an LLM that judges matching scores among agents for each atom in one pass. We ablate our rubric estimator against the LLM counterpart. As shown in Fig. 3 (*left*), the LLM estimator loses -3.27% points on `Qwen3.5-4B` and -3.85% on `Qwen3.5-9B`, leaving `Qwen3.5-9B` below the 22.21% it reaches under COALESCE alone (Tab. 2). Both estimators are given the same agent cards, evolving profiles, and atom demands. Therefore, what separates them is not what they know, but whether their estimated cost can effectively guide atom matching and coalescing to construct better workflows.

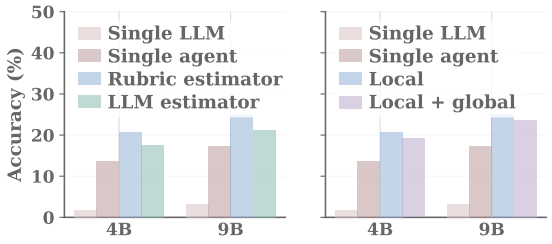

Figure 3: **How Fit Is Scored, and Where a Fault Is Sought.** Mean accuracy (%) on `Qwen3.5-4B` and `Qwen3.5-9B`. *Left*: using rubric estimator against an LLM for cost estimation. *Right*: in-flow optimization looking where a contract breaches, against the same with the global pass enabled beside it (§D.6).

Dynamic cost matrix that updates during COALESCE guides workflow construction better. Different from dynamic cost matrix that adaptively update with newly created agents, a static cost matrix is constructed once and then stays fixed. This leaves every new agent COALESCE creates outside the matching with other atoms in static matrix. As shown in Fig. 4, INFLOWOP with dynamic matrix outperforms the static matrix counterpart by $+3.15\%$ on `Qwen3.5-4B` and $+2.55\%$ on `Qwen3.5-9B`. Therefore, a cost matrix dynamic evolving with the agent pool is able to more effectively guide COALESCE to find more proper granularities.

Pools that dynamically grow build better workflows. The two pools differ where no available agent fits an atom well: a static agent pool settles for the closest agent it happens to hold, while a dynamic agent pool creates the specialist that atom demands. As shown in Fig. 4, INFLOWOP with a dynamic pool outperforms its static counterpart by $+4.49\%$ on `Qwen3.5-4B` and $+3.29\%$ on `Qwen3.5-9B`. Therefore, the pool sets a ceiling that no coalescing can lift: where no available agent covers an atom, dividing the task differently only relocates the shortfall, whereas creating the missing specialist removes it.

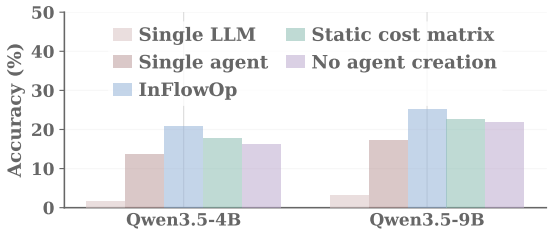

Figure 4: **Construction Earns Its Two Freedoms.** Mean accuracy (%) over `Qwen3.5` (`4B` and `9B`) among static cost matrix, dynamic cost matrix, and static pool.

Evolving profiles help only when they remember success. An agent can be scored on its agent card alone, or on the card together with the evolving profile it accumulates. As shown in Fig. 5, a profile evolving within a single run gains $+2.54\%$, $+1.33\%$, and $+2.23\%$ over the agent card alone on `Qwen3.5-4B`, `Qwen3.5-9B` and `GPT-5.4-mini`, respectively. When continuously evolving it into later runs, INFLOWOP gains a further $+1.21\%$, $+1.19\%$, and $+1.30\%$ when only successful ones are kept. Applying all to profiles instead reaches 17.34%, 21.52%, and 26.53% respectively for three backbones, below the agent card alone on all three backbones. This indicates that what makes a profile worth evolving is evidence of what an agent can do: a success tells the estimator what to trust that agent with, whereas a failure marks only where it once fell short, and prices the agent no better than no profile provided.

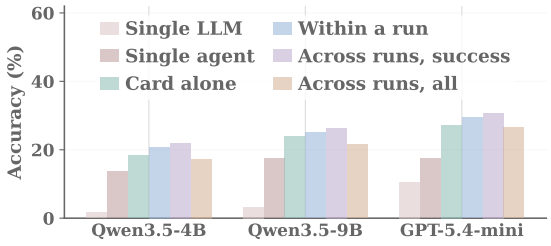

Figure 5: **A Profile Is Worth What It Remembers.** Mean accuracy (%) over an agent scored (1) on its card alone, (2) on a profile evolving within a run, and (3) on a profile evolving across runs, keeping either successful ones only or all of them.

Global localization rarely fires, and loses accuracy when it does. Beyond local in-flow optimization, can a workflow be improved further by a global pass over the same cost matrix (§D.6)? To investigate this, we compare accuracy under *local*-only in-flow optimization against *local*+*global*. As shown in Fig. 3 (*right*), enabling the global pass beside the local one yields -1.63% on `Qwen3.5-4B` and -1.41% on `Qwen3.5-9B`. This reveals that a breached contract already points at what to temper, and that tempering beyond it costs more than it recovers. This finding is also the empirical evidence for our configuration, under which in-flow optimization stays local in every other experiment.

6 CONCLUSIONS

In this work, we present INFLOWOP (§3), a two-stage framework (§D.1) of *bidirectional decomposition-aware construction* (§3.1) and *in-flow dynamic optimization* (§3.2). Given the challenge of workflow-level evaluation, we contribute BRAID, a benchmark whose tasks demand multi-agent coordination beyond single-agent capability (§4). We evaluate INFLOWOP across eight domains and six backbones (§5.1) against single-model and workflow baselines (§5.2) and ablate its design choices (§5.3).

AI USE STATEMENT

In this work, we used generative AI tools to *clean and reformat dataset*. We have **not** used generative AI tools to: *help develop theoretical models or conceptual frameworks, formulate mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, interpret results*. And *generate synthetic data sets, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, assist with translation, support qualitative and thematic data analysis* are not applicable to this work. Additionally, we used generative AI tools for *correct grammar and polish writing*. We have reviewed all AI-assisted work. For example, we have manually go through all the corrected grammar errors to ensure they are not false alarms, and we also manually code and check LLM-assisted quality control during dataset cleaning. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies how LLM-powered agents divide and carry out complex tasks, and raises no concerns involving human subjects, personal data, or sensitive attributes.

REPRODUCIBILITY STATEMENT

The method is specified in full in §3, with both stages given as pseudocode in Alg.3 and Alg.4 and an end-to-end elaboration in §D.1. BRAID's construction, source selection, and quality control are described in §C.2, and every metric used to grade a task is defined in §C.3. The experiment setup, including backbones, baselines, and the matched compute budget each baseline receives, is given in §5.1 and §E, and every parameter our runs configure is listed with its value in Tab. 4. Code for the method, the benchmark construction pipeline, and the evaluation harness, together with the composed BRAID data, will be made public upon acceptance.

REFERENCES

Long and Phan, Alice Gatti, Nathaniel Li, Adam Khoja, Ryan Kim, Richard Ren, Jason Hausenloy, Oliver Zhang, Mantas Mazeika, Dan Hendrycks, Ziwen Han, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, Michael Choi, Anish Agrawal, Arnav Chopra, Aakaash Nattanmai, Gordon McKellips, Anish Cheraku, Asim Suhail, Ethan Luo, Marvin Deng, Jason Luo, Ashley Zhang, Kavin Jindel, Jay Paek, Kasper Halevy, Allen Baranov, Michael Liu, Advaith Avadhanam, David Zhang, Vincent Cheng, Brad Ma, Evan Fu, Liam Do, Joshua Lass, Hubert Yang, Surya Sunkari, Vishruth Bharath, Violet Ai, James Leung, Rishit Agrawal, Alan Zhou, Kevin Chen, Tejas Kalpathi, Ziqi Xu, Gavin Wang, Tyler Xiao, Erik Maung, Sam Lee, Ryan Yang, Roy Yue, Ben Zhao, Julia Yoon, Xiangwan Sun, Aryan Singh, Clark Peng, Tyler Osbey, Taozhi Wang, Daryl Echeazu, Timothy Wu, Spandan Patel, Vidhi Kulkarni, Vijaykaarti Sundarapandiyan, Andrew Le, Zafir Nasim, Srikar Yalam, Ritesh Kasamsetty, Soham Samal, David Sun, Nihar Shah, Abhijeet Saha, Alex Zhang, Leon Nguyen, Laasya Nagumalli, Kaixin Wang, Aidan Wu, Anwith Telluri, Summer Yue, Alexandr Wang, Dmitry Dodonov, Tung Nguyen, Jaeho Lee, Daron Anderson, Mikhail Doroshenko, Alun Cennyth Stokes, Mobeen Mahmood, Oleksandr Pokutnyi, Oleg Iskra, Jessica P. Wang, John-Clark Levin, Mstyslav Kazakov, Fiona Feng, Steven Y. Feng, Haoran Zhao, Michael Yu, Varun Gangal, Chelsea Zou, Zihan Wang, Serguei Popov, Robert Gerbicz, Geoff Galgon, Johannes Schmitt, Will Yeadon, Yongki Lee, Scott Sauers, Alvaro Sanchez, Fabian Giska, Marc Roth, Søren Riis, Saiteja Utpala, Noah Burns, Gashaw M. Goshu, Mohinder Maheshbhai Naiya, Chidozie Agu, Zachary Giboney, Antrell Cheatom, Francesco Fournier-Facio, Sarah-Jane Crowson, Lennart Finke, Zerui Cheng, Jennifer Zampese, Ryan G. Hoerr, Mark Nandor, Hyunwoo Park, Tim Gehrunger, Jiaqi Cai, Ben McCarty, Alexis C. Garretson, Edwin Taylor, Damien Sileo, Qiuyu Ren, Usman Qazi, Lianghui Li, Jungbae Nam, John B. Wydallis, Pavel Arkhipov, Jack Wei Lun Shi, Aras Bacho, Chris G. Willcocks, Hangrui Cao,

Sumeet Motwani, Emily de Oliveira Santos, Johannes Veith, Edward Vendrow, Doru Cojoc, Kengo Zenitani, Joshua Robinson, Longke Tang, Yuqi Li, Joshua Vendrow, Natanael Wildner Fraga, Vladyslav Kuchkin, Andrey Pupasov Maksimov, Pierre Marion, Denis Efremov, Jayson Lynch, Kaiqu Liang, Aleksandar Mikov, Andrew Gritsevskiy, Julien Guillod, Gözdenur Demir, Dakotah Martinez, Ben Pageler, Kevin Zhou, Saeed Soori, Ori Press, Henry Tang, Paolo Rissone, Sean R. Green, Lina Brüssel, Moon Twayana, Aymeric Dieuleveut, Joseph Marvin Imperial, Ameya Prabhu, Jinzhou Yang, Nick Crispino, Arun Rao, Dimitri Zvonkine, Gabriel Loiseau, Mikhail Kalinin, Marco Lukas, Ciprian Manolescu, Nate Stambaugh, Subrata Mishra, Tad Hogg, Carlo Bosio, Brian P. Coppola, Julian Salazar, Jaehyeok Jin, Rafael Sayous, Stefan Ivanov, Philippe Schwaller, Shaipranesh Senthilkumar, Andres M. Bran, Andres Algaba, Kelsey Van den Houte, Lynn Van Der Sypt, Brecht Verbeken, David Noever, Alexei Kopylov, Benjamin Myklebust, Bikun Li, Lisa Schut, Evgenii Zheltonozhskii, Qiaochu Yuan, Derek Lim, Richard Stanley, Tong Yang, John Maar, Julian Wykowski, Mart Oller, Anmol Sahu, Cesare Giulio Ardito, Yuzheng Hu, Ariel Ghislain Kemogne Kamdoum, Alvin Jin, Tobias Garcia Vilchis, Yuexuan Zu, Martin Lackner, James Koppel, Gongbo Sun, Daniil S. Antonenko, Steffi Chern, Bingchen Zhao, Pierrot Arsene, Joseph M. Cavanagh, Daofeng Li, Jiawei Shen, Donato Crisostomi, Wenjin Zhang, Ali Dehghan, Sergey Ivanov, David Perrella, Nurdin Kaparov, Allen Zang, Ilia Sucholutsky, Arina Kharlamova, Daniil Orel, Vladislav Poritski, Shalev Ben-David, Zachary Berger, Parker Whitfill, Michael Foster, Daniel Munro, Linh Ho, Shankar Sivarajan, Dan Bar Hava, Aleksey Kuchkin, David Holmes, Alexandra Rodriguez-Romero, Frank Sommerhage, Anji Zhang, Richard Moat, Keith Schneider, Zakayo Kazibwe, Don Clarke, Dae Hyun Kim, Felipe Meneguitti Dias, Sara Fish, Veit Elser, Tobias Kreiman, Victor Efren Guadarrama Vilchis, Immo Klose, Ujjwala Anantheswaran, Adam Zweiger, Kaivalya Rawal, Jeffery Li, Jeremy Nguyen, Nicolas Daans, Haline Heidinger, Maksim Radionov, Václav Rozhoň, Vincent Ginis, Christian Stump, Niv Cohen, Rafał Poświata, Josef Tkadlec, Alan Goldfarb, Chenguang Wang, Piotr Padlewski, Stanislaw Barzowski, Kyle Montgomery, Ryan Stendall, Jamie Tucker-Foltz, Jack Stade, T. Ryan Rogers, Tom Goertzen, Declan Grabb, Abhishek Shukla, Alan Givré, John Arnold Ambay, Archan Sen, Muhammad Fayez Aziz, Mark H. Inlow, Hao He, Ling Zhang, Younesse Kaddar, Ivar Ängquist, Yanxu Chen, Harrison K. Wang, Kalyan Ramakrishnan, Elliott Thornley, Antonio Terpin, Hailey Schoelkopf, Eric Zheng, Avishy Carmi, Ethan D. L. Brown, Kelin Zhu, Max Bartolo, Richard Wheeler, Martin Stehberger, Peter Bradshaw, JP Heimonen, Kaustubh Sridhar, Ido Akov, Jennifer Sandlin, Yury Makarychev, Joanna Tam, Hieu Hoang, David M. Cunningham, Vladimir Goryachev, Demosthenes Patramanis, Michael Krause, Andrew Redenti, David Aldous, Jesyin Lai, Shannon Coleman, Jiangnan Xu, Sangwon Lee, Ilias Magoulas, Sandy Zhao, Ning Tang, Michael K. Cohen, Orr Paradise, Jan Hendrik Kirchner, Maksym Ovchynnikov, Jason O. Matos, Adithya Shenoy, Michael Wang, Yuzhou Nie, Anna Sztyber-Betley, Paolo Faraboschi, Robin Riblet, Jonathan Crozier, Shiv Halasyamani, Shreyas Verma, Prashant Joshi, Eli Meril, Ziqiao Ma, Jérémy Andréoletti, Raghav Singhal, Jacob Platnick, Volodymyr Nevirkovets, Luke Basler, Alexander Ivanov, Seri Khoury, Nils Gustafsson, Marco Piccardo, Hamid Mostaghimi, Qijia Chen, Virendra Singh, Tran Quoc Khánh, Paul Rosu, Hannah Szlyk, Zachary Brown, Himanshu Narayan, Aline Menezes, Jonathan Roberts, William Alley, Kunyang Sun, Arkil Patel, Max Lamparth, Anka Reuel, Linwei Xin, Hanmeng Xu, Jacob Loader, Freddie Martin, Zixuan Wang, Andrea Achilleos, Thomas Preu, Tomek Korbak, Ida Bosio, Fereshteh Kazemi, Ziye Chen, Biró Bálint, Eve J. Y. Lo, Jiaqi Wang, Maria Inês S. Nunes, Jeremiah Milbauer, M. Saiful Bari, Zihao Wang, Behzad Ansarinejad, Yewen Sun, Stephane Durand, Hossam Elgnainy, Guillaume Douville, Daniel Tordera, George Balabanian, Hew Wolff, Lynna Kvistad, Hsiaoyun Milliron, Ahmad Sakor, Murat Eron, Andrew Favre, Shailesh Shah, Xiaoxiang Zhou, Firuz Kamalov, Sherwin Abdoli, Tim Santens, Shaul Barkan, Allison Tee, Robin Zhang, Alessandro Tomasiello, G. Bruno De Luca, Shi-Zhuo Looi, Vinh-Kha Le, Noam Kolt, Jiayi Pan, Emma Rodman, Jacob Drori, Carl J. Fossum, Niklas Muennighoff, Milind Jagota, Ronak Pradeep, Honglu Fan, Jonathan Eicher, Michael Chen, Kushal Thaman, William Merrill, Moritz Firsching, Carter Harris, Stefan Ciobâcă, Jason Gross, Rohan Pandey, Ilya Gusev, Adam Jones, Shashank Agnihotri, Pavel Zhelnov, Mohammadreza Mofayezi, Alexander Piperski, David K. Zhang, Kostiantyn Dobarskyi, Roman Leventov, Ignat Soroko, Joshua Duersch, Vage Taamazyan, Andrew Ho, Wenjie Ma, William Held, Ruicheng Xian, Armel Randy Zebaze, Mohanad Mohamed, Julian Noah Leser, Michelle X. Yuan, Laila

Yacar, Johannes Lengler, Katarzyna Olszewska, Claudio Di Fratta, Edson Oliveira, Joseph W. Jackson, Andy Zou, Muthu Chidambaram, Timothy Manik, Hector Haffenden, Dashiell Stander, Ali Dasouqi, Alexander Shen, Bita Golshani, David Stap, Egor Kretov, Mikalai Uzhou, Alina Borisovna Zhidkovskaya, Nick Winter, Miguel Orbegozo Rodriguez, Robert Lauff, Dustin Wehr, Colin Tang, Zaki Hossain, Shaun Phillips, Fortuna Samuele, Fredrik Ekström, Angela Hammon, Oam Patel, Faraz Farhidi, George Medley, Forough Mohammadzadeh, Madellene Peñaflor, Haile Kassahun, Alena Friedrich, Rayner Hernandez Perez, Daniel Pyda, Taom Sakal, Omkar Dhamane, Ali Khajegili Mirabadi, Eric Hallman, Kenchi Okutsu, Mike Battaglia, Mohammad Maghsoudimehrabani, Alon Amit, Dave Hulbert, Roberto Pereira, Simon Weber, Handoko, Anton Peristyy, Stephen Malina, Mustafa Mehkary, Rami Aly, Frank Reidegeld, Anna-Katharina Dick, Cary Friday, Mukhwinder Singh, Hassan Shapourian, Wanyoung Kim, Mariana Costa, Hubeyb Gurdogan, Harsh Kumar, Chiara Ceconello, Chao Zhuang, Haon Park, Micah Carroll, Andrew R. Tawfeek, Stefan Steinerberger, Daattavya Aggarwal, Michael Kirchhof, Linjie Dai, Evan Kim, Johan Ferret, Jainam Shah, Yuzhou Wang, Minghao Yan, Krzysztof Burdzy, Lixin Zhang, Antonio Franca, Diana T. Pham, Kang Yong Loh, Joshua Robinson, Abram Jackson, Paolo Giordano, Philipp Petersen, Adrian Cosma, Jesus Colino, Colin White, Jacob Votava, Vladimir Vinnikov, Ethan Delaney, Petr Spelda, Vit Stritecky, Syed M. Shahid, Jean-Christophe Mourrat, Lavr Vetoshkin, Koen Sponselee, Renas Bacho, Zheng-Xin Yong, Florencia de la Rosa, Nathan Cho, Xiuyu Li, Guillaume Malod, Orion Weller, Guglielmo Albani, Leon Lang, Julien Laurendeau, Dmitry Kazakov, Fatimah Adesanya, Julien Portier, Lawrence Hollom, Victor Souza, Yuchen Anna Zhou, Julien Degorre, Yiğit Yalın, Gbenga Daniel Obikoya, Rai Michael Pokorny, Filippo Bigi, M. C. Boscá, Oleg Shumar, Kaniuar Bacho, Gabriel Recchia, Mara Popescu, Nikita Shulga, Ngefor Mildred Tanwie, Thomas C. H. Lux, Ben Rank, Colin Ni, Matthew Brooks, Alesia Yakimchyk, Huanxu Quinn Liu, Stefano Cavalleri, Olle Häggström, Emil Verkama, Joshua Newbould, Hans Gundlach, Leonor Brito-Santana, Brian Amaro, Vivek Vajipey, Rynaa Grover, Ting Wang, Yosi Kratish, Wen-Ding Li, Sivakanth Gopi, Andrea Caciolai, Christian Schroeder de Witt, Pablo Hernández-Cámara, Emanuele Rodolà, Jules Robins, Dominic Williamson, Brad Raynor, Hao Qi, Ben Segev, Jingxuan Fan, Sarah Martinson, Erik Y. Wang, Kaylie Hausknecht, Michael P. Brenner, Mao Mao, Christoph Demian, Peyman Kassani, Xinyu Zhang, David Avagian, Eshawn Jessica Scipio, Alon Ragoler, Justin Tan, Blake Sims, Rebeka Plecnik, Aaron Kirtland, Omer Faruk Bodur, D. P. Shinde, Yan Carlos Leyva Labrador, Zahra Adoul, Mohamed Zekry, Ali Karakoc, Tania C. B. Santos, Samir Shamseldeen, Loukmane Karim, Anna Liakhovitskaia, Nate Resman, Nicholas Farina, Juan Carlos Gonzalez, Gabe Maayan, Earth Anderson, Rodrigo De Oliveira Pena, Elizabeth Kelley, Hodjat Mariji, Rasoul Pouriamanesh, Wentao Wu, Ross Finocchio, Ismail Alarab, Joshua Cole, Danyelle Ferreira, Bryan Johnson, Mohammad Safdari, Liangti Dai, Siriphan Arthornthurasuk, Isaac C. McAlister, Alejandro José Moyano, Alexey Pronin, Jing Fan, Angel Ramirez-Trinidad, Yana Malysheva, Daphiny Pottmaier, Omid Taheri, Stanley Stepanic, Samuel Perry, Luke Askew, Raúl Adrián Huerta Rodrguez, Ali M. R. Minissi, Ricardo Lorena, Krishnamurthy Iyer, Arshad Anil Fasiludeen, Ronald Clark, Josh Ducey, Matheus Piza, Maja Somrak, Eric Vergo, Juehang Qin, Benjámin Borbás, Eric Chu, Jack Lindsey, Antoine Jallon, I. M. J. McInnis, Evan Chen, Avi Semler, Luk Gloor, Tej Shah, Marc Carauleanu, Pascal Lauer, Tran Duc Huy, Hossein Shahrtash, Emilien Duc, Lukas Lewark, Assaf Brown, Samuel Albanie, Brian Weber, Warren S. Vaz, Pierre Clavier, Yiyang Fan, Gabriel Poesia Reis e Silva, Long Tony Lian, Marcus Abramovitch, Xi Jiang, Sandra Mendoza, Murat Islam, Juan Gonzalez, Vasilios Mavroudis, Justin Xu, Pawan Kumar, Laxman Prasad Goswami, Daniel Bugas, Nasser Heydari, Ferenc Jeanplong, Thorben Jansen, Antonella Pinto, Archimedes Apronti, Abdallah Galal, Ng Ze-An, Ankit Singh, Tong Jiang, Joan of Arc Xavier, Kanu Priya Agarwal, Mohammed Berkani, Gang Zhang, Zhehang Du, Benedito Alves de Oliveira Junior, Dmitry Malishev, Nicolas Remy, Taylor D. Hartman, Tim Tarver, Stephen Mensah, Gautier Abou Loume, Wiktor Morak, Farzad Habibi, Sarah Hoback, Will Cai, Javier Gimenez, Roselynn Grace Montecillo, Jakub Łucki, Russell Campbell, Asankhaya Sharma, Khalida Meer, Shreen Gul, Daniel Espinosa Gonzalez, Xavier Alapont, Alex Hoover, Gunjan Chhablani, Freddie Vargus, Arunim Agarwal, Yibo Jiang, Deepakkumar Patil, David Outevsky, Kevin Joseph Scaria, Rajat Maheshwari, Abdelkader Dendane, Priti Shukla, Ashley Cartwright, Sergei Bogdanov, Niels Mündler, Sören Möller, Luca Arnaboldi, Kunvar Thaman,

Muhammad Rehan Siddiqi, Prajvi Saxena, Himanshu Gupta, Tony Fruhauff, Glen Sherman, Mátyás Vincze, Siranut Usawasutsakorn, Dylan Ler, Anil Radhakrishnan, Innocent Enyekwe, Sk Md Salauddin, Jiang Muzhen, Aleksandr Maksapetyan, Vivien Rossbach, Chris Harjadi, Mohsen Bahaloohoreh, Claire Sparrow, Jasdeep Sidhu, Sam Ali, Song Bian, John Lai, Eric Singer, Justine Leon Uro, Greg Bateman, Mohamed Sayed, Ahmed Menshawy, Darling Duclosel, Dario Bezzi, Yashaswini Jain, Ashley Aaron, Murat Tiryakioglu, Sheeshram Siddh, Keith Krenek, Imad Ali Shah, Jun Jin, Scott Creighton, Denis Peskoff, Zienab EL-Wasif, Ragavendran P, Michael Richmond, Joseph McGowan, Tejal Patwardhan, Hao-Yu Sun, Ting Sun, Nikola Zubić, Samuele Sala, Stephen Ebert, Jean Kaddour, Manuel Schottdorf, Dianzhuo Wang, Gerol Petruzella, Alex Meiburg, Tilen Medved, Ali ElSheikh, S. Ashwin Hebbar, Lorenzo Vaquero, Xianjun Yang, Jason Poulos, Vilém Zouhar, Sergey Bogdanik, Mingfang Zhang, Jorge Sanz-Ros, David Anugraha, Yinwei Dai, Anh N. Nhu, Xue Wang, Ali Anil Demircali, Zhibai Jia, Yuyin Zhou, Juncheng Wu, Mike He, Nitin Chandok, Aarush Sinha, Gaoxiang Luo, Long Le, Mickaël Noyé, Michał Perełkiewicz, Ioannis Pantidis, Tianbo Qi, Soham Sachin Purohit, Letitia Parcalabescu, Thai-Hoa Nguyen, Genta Indra Winata, Edoardo M. Ponti, Hanchen Li, Kaustubh Dhole, Jongee Park, Dario Abbondanza, Yuanli Wang, Anupam Nayak, Diogo M. Caetano, Antonio A. W. L. Wong, Maria del Rio-Chanona, Dániel Kondor, Pieter Francois, Ed Chalstrey, Jakob Zsambok, Dan Hoyer, Jenny Reddish, Jakob Hauser, Francisco-Javier Rodrigo-Ginés, Suchandra Datta, Maxwell Shepherd, Thom Kamphuis, Qizheng Zhang, Hyunjun Kim, Ruiji Sun, Jianzhu Yao, Franck Dernoncourt, Satyapriya Krishna, Sina Rismanchian, Bonan Pu, Francesco Pinto, Yingheng Wang, Kumar Shridhar, Kalon J. Overholt, Glib Briia, Hieu Nguyen, David Quod Soler Bartomeu, Tony CY Pang, Adam Wecker, Yifan Xiong, Fanfei Li, Lukas S. Huber, Joshua Jaeger, Romano De Maddalena, Xing Han Lù, Yuhui Zhang, Claas Beger, Patrick Tser Jern Kon, Sean Li, Vivek Sanker, Ming Yin, Yihao Liang, Xinlu Zhang, Ankit Agrawal, Li S. Yifei, Zechen Zhang, Mu Cai, Yasin Sonmez, Costin Cozianu, Changhao Li, Alex Slen, Shoubin Yu, Hyun Kyu Park, Gabriele Sarti, Marcin Briański, Alessandro Stolfo, Truong An Nguyen, Mike Zhang, Yotam Perlitz, Jose Hernandez-Orallo, Runjia Li, Amin Shabani, Felix Juefei-Xu, Shikhar Dhingra, Orr Zohar, My Chiffon Nguyen, Alexander Pondaven, Abdurrahim Yilmaz, Xuandong Zhao, Chuanyang Jin, Muyan Jiang, Stefan Todoran, Xinyao Han, Jules Kreuer, Brian Rabern, Anna Plassart, Martino Maggetti, Luther Yap, Robert Geirhos, Jonathon Kean, Dingsu Wang, Sina Mollaei, Chenkai Sun, Yifan Yin, Shiqi Wang, Rui Li, Yaowen Chang, Anjiang Wei, Alice Bizeul, Xiaohan Wang, Alexandre Oliveira Arrais, Kushin Mukherjee, Jorge Chamorro-Padial, Jiachen Liu, Xingyu Qu, Junyi Guan, Adam Bouyamourn, Shuyu Wu, Martyna Plomecka, Junda Chen, Mengze Tang, Jiaqi Deng, Shreyas Subramanian, Haocheng Xi, Haoxuan Chen, Weizhi Zhang, Yinuo Ren, Haoqin Tu, Sejong Kim, Yushun Chen, Sara Vera Marjanović, Junwoo Ha, Grzegorz Luczyna, Jeff J. Ma, Zewen Shen, Dawn Song, Cedegao E. Zhang, Zhun Wang, Gaël Gendron, Yunze Xiao, Leo Smucker, Erica Weng, Kwok Hao Lee, Zhe Ye, Stefano Ermon, Ignacio D. Lopez-Miguel, Theo Knights, Anthony Gitter, Namkyu Park, Boyi Wei, Hongzheng Chen, Kunal Pai, Ahmed Elkhanany, Han Lin, Philipp D. Siedler, Jichao Fang, Ritwik Mishra, Károly Zsolnai-Fehér, Xilin Jiang, Shadab Khan, Jun Yuan, Rishab Kumar Jain, Xi Lin, Mike Peterson, Zhe Wang, Aditya Malusare, Maosen Tang, Isha Gupta, Ivan Fosin, Timothy Kang, Barbara Dworakowska, Kazuki Matsumoto, Guangyao Zheng, Gerben Sewuster, Jorge Pretel Villanueva, Ivan Rannev, Igor Chernyavsky, Jiale Chen, Deepayan Banik, Ben Racz, Wenchao Dong, Jianxin Wang, Laila Bashmal, Duarte V. Gonçalves, Wei Hu, Kaushik Bar, Ondrej Bohdal, Atharv Singh Patlan, Shehzaad Dhuliawala, Caroline Geirhos, Julien Wist, Yuval Kansal, Bingsen Chen, Kutay Tire, Atak Talay Yücel, Brandon Christof, Veerupaksh Singla, Zijian Song, Sanxing Chen, Jiaxin Ge, Kaustubh Ponkshe, Isaac Park, Tianneng Shi, Martin Q. Ma, Joshua Mak, Sherwin Lai, Antoine Moulin, Zhuo Cheng, Zhanda Zhu, Ziyi Zhang, Vaidehi Patil, Ketan Jha, Qiutong Men, Jiaxuan Wu, Tianchi Zhang, Bruno Hebling Vieira, Alham Fikri Aji, Jae-Won Chung, Mohammed Mahfoud, Ha Thi Hoang, Marc Sperzel, Wei Hao, Kristof Meding, Sihan Xu, Vassilis Kostakos, Davide Manini, Yueying Liu, Christopher Toukmaji, Eunmi Yu, Arif Engin Demircali, Zhiyi Sun, Ivan Dewerpe, Hongsen Qin, Roman Pflugfelder, James Bailey, Johnathan Morris, Ville Heilala, Sybille Rosset, Zishun Yu, Peter E. Chen, Woongyeong Yeo, Eeshaan Jain, Sreekar Chigurupati, Julia Chernyavsky, Sai Prajwal Reddy, Subhashini Venugopalan, Hunar Batra, Core Francisco Park, Hieu Tran, Guilherme Maximiano, Genghan Zhang, Yizhuo Liang,

Hu Shiyu, Rongwu Xu, Rui Pan, Siddharth Suresh, Ziqi Liu, Samaksh Gulati, Songyang Zhang, Peter Turchin, Christopher W. Bartlett, Christopher R. Scotese, Phuong M. Cao, Ben Wu, Jacek Karwowski, and Davide Scaramuzza. A benchmark of expert-level academic questions to assess ai capabilities, January 2026. ISSN 1476-4687. URL http://dx.doi.org/10.1038/s41586-025-09962-4.

Summer Eunhyung Ann, Haokun Liu, and Chenhao Tan. The interaction tax: When communication erases diversity in multi-agent teams, 2026. URL https://arxiv.org/abs/2608.23541.

Hyungjoo Chae, Sunghwan Kim, Junhee Cho, Seungone Kim, Seungjun Moon, Gyeom Hwangbo, Dongha Lim, Minjin Kim, Yeonjun Hwang, Minju Gwak, Dongwook Choi, Minseok Kang, Gwanhoon Im, ByeongUng Cho, Hyojun Kim, Jun Hee Han, Taeyoon Kwon, Minju Kim, Beong woo Kwak, Dongjin Kang, and Jinyoung Yeo. Web-shepherd: Advancing prms for reinforcing web agents, 2025. URL https://arxiv.org/abs/2505.15277.

Junzhi Chen, Harsh Trivedi, Jane Pan, Michael JQ Zhang, Tejas Srinivasan, Niranjan Balasubramanian, and Ashish Sabharwal. Appworld-ul: Benchmarking diverse agent-user interactions for tool-use, 2026a. URL https://arxiv.org/abs/2607.20536.

Mengzhuo Chen, Junjie Wang, Fangwen Mu, Yawen Wang, Zhe Liu, Huanxiang Feng, and Qing Wang. Seeing the whole elephant: A benchmark for failure attribution in llm-based multi-agent systems, 2026b. URL https://arxiv.org/abs/2604.22708.

Hyeong Kyu Choi, Xiaojin Zhu, and Sharon Li. Debate or vote: Which yields better decisions in multi-agent large language models?, 2025. URL https://arxiv.org/abs/2508.17536.

Zhiqing Cui, Haotong Xie, Jiahao Yuan, Cheng Yang, Hanqing Wang, Yuxin Wu, Yifan Wu, Siru Zhong, Tao Yu, Yifu Guo, Siyu Zhang, Xinlei Yu, Qibing Ren, and Usman Naseem. Uno-orchestra: Parsimonious agent routing via selective delegation, 2026. URL https://arxiv.org/abs/2605.05007.

Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, Xuantang Xiong, Lei Han, Zhiyuan Liu, and Maosong Sun. Multi-agent collaboration via evolving orchestration, 2025. URL https://arxiv.org/abs/2505.19591.

Chao Deng, Jiale Yuan, Pi Bu, Peijie Wang, Zhong-Zhi Li, Jian Xu, Xiao-Hui Li, Yuan Gao, Jun Song, Bo Zheng, and Cheng-Lin Liu. Longdocurl: a comprehensive multimodal long document benchmark integrating understanding, reasoning, and locating, 2025. URL https://arxiv.org/abs/2412.18424.

Jiangwen Dong, Jiayu Li, Tianhang Zheng, and Wanyu Lin. Hybridflow: Resource-adaptive subtask routing for efficient edge-cloud llm inference, 2026. URL https://arxiv.org/abs/2512.22137.

Jinshan Gao, Zhuoran Jin, Tianyi Men, Kang Liu, and Jun Zhao. Swarmbench: Can large language models act as agent swarm orchestrators?, 2026. URL https://arxiv.org/abs/2608.30661.

Xuehang Guo, Zhiyong Lu, Tom Hope, and Qingyun Wang. Anagent for enhancing scientific table & figure analysis, 2026. URL https://arxiv.org/abs/2602.10081.

Advait Gupta, Rishie Raj, Dang Nguyen, and Tianyi Zhou. Fasta*: Fast-slow toolpath agent with subroutine mining for efficient multi-turn image editing, 2026. URL https://arxiv.org/abs/2506.20911.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Eran Hirsch, David Wan, Han Wang, Elias Stengel-Eskin, Mohit Bansal, and Ido Dagan. Who is the agent to blame? localizing faithfulness and citation mistakes in agentic deep research, 2026. URL https://arxiv.org/abs/2608.24306.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework, 2024. URL https://arxiv.org/abs/2308.00352.

Jun Hou, Priya Pitre, Yi Fang, and Xuan Wang. Edge: Error dependency graph-guided multi-error attribution in multi-agent llm systems, 2026. URL https://arxiv.org/abs/2609.01360.

Dae Yon Hwang, Raunaq Suri, Valentin Villecroze, Anthony L. Caterini, Jesse C. Cresswell, Noël Vouitsis, and Brendan Leigh Ross. Agentic monte carlo: Simulating reinforcement learning for black-box agents, 2026. URL https://arxiv.org/abs/2606.05296.

Prathyusha Jwalapuram, Hehai Lin, Chuyuan Li, Fangkai Jiao, Sudong Wang, Yifei Ming, Zixuan Ke, Chengwei Qin, Giuseppe Carenini, and Shafiq Joty. The illusion of multi-agent advantage, 2026. URL https://arxiv.org/abs/2606.13003.

Vansh Kapoor, Aman Gupta, Hao Chen, Anurag Beniwal, Jing Huang, and Aviral Kumar. Trim: Hybrid inference via targeted stepwise routing in multi-step reasoning tasks, 2026. URL https://arxiv.org/abs/2601.10245.

Jaewoo Lee, Archiki Prasad, Justin Chih-Yao Chen, Zaid Khan, Elias Stengel-Eskin, and Mohit Bansal. Prints: Reward modeling for long-horizon information seeking, 2026a. URL https://arxiv.org/abs/2511.19314.

Yoonsang Lee, Howard Yen, Xi Ye, and Danqi Chen. Agentic aggregation for parallel scaling of long-horizon agentic tasks, 2026b. URL https://arxiv.org/abs/2604.11753.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. Taco: Topics in algorithmic code generation dataset, 2023. URL https://arxiv.org/abs/2312.14852.

Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation, 2025a. URL https://arxiv.org/abs/2507.18224.

Zhening Li, Armando Solar-Lezama, Yisong Yue, and Stephan Zheng. Encompass: Enhancing agent programming with search over program execution paths, 2025b. URL https://arxiv.org/abs/2512.03571.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use, 2026. URL https://arxiv.org/abs/2510.05592.

Xiaofeng Lin, Yingxu Wang, Tung Sum Thomas Kwok, Daniel Guo, Sahil Arun Nale, Charles Fleming, and Guang Cheng. Reflect: Intervention-supported error attribution for silent failures in llm agent traces, 2026. URL https://arxiv.org/abs/2606.09071.

Shuodi Liu, Yingzhuo Liu, Zi Wang, Yusheng Wang, Huijia Wu, Liuyu Xiang, and Zhaofeng He. Select-then-decompose: From empirical analysis to adaptive selection strategy for task decomposition in large language models, 2025. URL https://arxiv.org/abs/2510.17922.

Yubo Ma, Yuhang Zang, Liangyu Chen, Meiqi Chen, Yizhu Jiao, Xinze Li, Xinyuan Lu, Ziyu Liu, Yan Ma, Xiaoyi Dong, Pan Zhang, Liangming Pan, Yu-Gang Jiang, Jiaqi Wang, Yixin Cao, and Aixin Sun. Mmlongbench-doc: Benchmarking long-context document understanding with visualizations, 2024. URL https://arxiv.org/abs/2407.01523.

Dhruv Madhwal, Lyuxin David Zhang, Dan Roth, Tomer Wolfson, and Vivek Gupta. Decomposed prompting does not fix knowledge gaps, but helps models say "I don't know". In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 36688–36710, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1829. URL https://aclanthology.org/2026.findings-acl.1829/.

Stefan Nielsen, Edoardo Cetin, Peter Schwendeman, Qi Sun, Jinglue Xu, and Yujin Tang. Learning to orchestrate agents in natural language with the conductor, 2026. URL https://arxiv.org/abs/2512.04388.

Jingjie Ning, Xueqi Li, and Chengyu Yu. Revision or re-solving? decomposing second-pass gains in multi-llm pipelines, 2026. URL https://arxiv.org/abs/2604.01029.

OpenAI. GPT-5 Mini Model, 2025. URL https://developers.openai.com/api/docs/models/gpt-5-mini.

OpenAI. GPT-5.4 Model, 2026a. URL https://developers.openai.com/api/docs/models/gpt-5.4.

OpenAI. GPT-5.4 Mini Model, 2026b. URL https://developers.openai.com/api/docs/models/gpt-5.4-mini.

OpenAI. GPT-5.6 Luna Model, 2026c. URL https://developers.openai.com/api/docs/models/gpt-5.6-luna.

Jiangweizhi Peng, Yuanxin Liu, Ruida Zhou, Charles Fleming, Zhaoran Wang, Alfredo Garcia, and Mingyi Hong. Hiper: Hierarchical reinforcement learning with explicit credit assignment for large language model agents, 2026. URL https://arxiv.org/abs/2602.16165.

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. Adapt: As-needed decomposition and planning with language models, 2024. URL https://arxiv.org/abs/2311.05772.

Varshini Reddy, Rik Koncel-Kedziorski, Viet Dac Lai, Michael Krumdick, Charles Lovering, and Chris Tanner. Docfinqa: A long-context financial reasoning dataset, 2025. URL https://arxiv.org/abs/2401.06915.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023. URL https://arxiv.org/abs/2302.04761.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. Agentsquare: Automatic llm agent search in modular design space, 2025. URL https://arxiv.org/abs/2410.06153.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face, 2023. URL https://arxiv.org/abs/2303.17580.

Chufan Shi, Cheng Yang, Yaokang Wu, Linghao Jin, Bo Shui, Taylor Berg-Kirkpatrick, and Xuezhe Ma. Are vlms seeing or just saying? uncovering the illusion of visual re-examination, 2026. URL https://arxiv.org/abs/2605.15864.

Jinwei Su, Qizhen Lan, Yinghui Xia, Lifan Sun, Weiyou Tian, Tianyu Shi, Xinyuan Song, Lewei He, and Yang Jingsong. Difficulty-aware agentic orchestration for query-specific multi-agent workflows, 2026. URL https://arxiv.org/abs/2509.11079.

Weixiang Sun, Zehong Wang, Hong Huang, Colby Nelson, and Yanfang Ye. The collaboration tax: How much llm multi-agent systems pay to coordinate, 2026. URL https://arxiv.org/abs/2608.22152.

Ryota Tanaka, Kyosuke Nishida, Kosuke Nishida, Taku Hasegawa, Itsumi Saito, and Kuniko Saito. Slidevqa: A dataset for document visual question answering on multiple images, 2023. URL https://arxiv.org/abs/2301.04883.

Liyan Tang, Grace Kim, Xinyu Zhao, Thom Lake, Wenxuan Ding, Fangcong Yin, Prasann Singhal, Manya Wadhwa, Zeyu Leo Liu, Zayne Sprague, Ramya Namuduri, Bodun Hu, Juan Diego Rodriguez, Puyuan Peng, and Greg Durrett. Chartmuseum: Testing visual reasoning capabilities of large vision-language models, 2026. URL https://arxiv.org/abs/2505.13444.

Qwen Team. Qwen3.5-omni technical report, 2026. URL https://arxiv.org/abs/2604.15804.

Yuxuan Wan, Tianqing Fang, Zaitang Li, Yintong Huo, Wenxuan Wang, Haitao Mi, Dong Yu, and Michael R. Lyu. Inference-time scaling of verification: Self-evolving deep research agents via test-time rubric-guided verification, 2026. URL https://arxiv.org/abs/2601.15808.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023a. URL https://arxiv.org/abs/2305.16291.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models, 2023b. URL https://arxiv.org/abs/2305.04091.

Yaoxiang Wang, Zhiyong Wu, Junfeng Yao, and Jinsong Su. Tdag: A multi-agent framework based on dynamic task decomposition and agent generation, 2025. URL https://arxiv.org/abs/2402.10178.

Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, Alexis Chevalier, Sanjeev Arora, and Danqi Chen. Charxiv: Charting gaps in realistic chart understanding in multimodal llms, 2024. URL https://arxiv.org/abs/2406.18521.

Xiao Wu, Ting-Zhu Huang, Liang-Jian Deng, Yanyuan Qiao, Imran Razzak, and Yutong Xie. A knowledge-driven adaptive collaboration of llms for enhancing medical decision-making, 2025. URL https://arxiv.org/abs/2509.14998.

Zihan Xu, Haolin Tian, and Hai Jiang. A two-tier perspective on inference-time parallelism in multi-agent llm systems, 2026. URL https://arxiv.org/abs/2608.05791.

Liming Yang, Junyu Luo, Xuanzhe Liu, Yiling Lou, and Zhenpeng Chen. Bamas: Structuring budget-aware multi-agent systems, 2025a. URL https://arxiv.org/abs/2511.21572.

Zhe Yang, Yichang Zhang, Yudong Wang, Ziyao Xu, Junyang Lin, and Zhifang Sui. A probabilistic inference scaling theory for llm self-correction, 2025b. URL https://arxiv.org/abs/2508.16456.

Youngsik Yoon, Sungjae Lee, Seockbean Song, Siwei Wang, Wei Chen, and Jungseul Ok. Pat: Planning-after-trial for efficient test-time code generation, 2026. URL https://arxiv.org/abs/2605.07248.

Michelle Yuan, Khushbu Pahwa, Shuaichen Chang, Mustafa Kaba, Jiarong Jiang, Xiaofei Ma, Yi Zhang, and Monica Sunkara. Automated composition of agents: A knapsack approach for agentic component selection, 2025. URL https://arxiv.org/abs/2510.16499.

Yang Yue, Xuancheng Zhu, Yuyang Ma, Guoshun Nan, Zihan Dou, Jingru Shan, Congyu Guo, Ji Zhang, Hua Wang, and Jingfeng Zhang. Autoras: Learning robust agentic systems with primitive representations, 2026. URL https://arxiv.org/abs/2606.21445.

Sukwon Yun, Jie Peng, Pingzhi Li, Wendong Fan, Jie Chen, James Zou, Guohao Li, and Tianlong Chen. Graph-of-agents: A graph-based framework for multi-agent llm collaboration, 2026. URL https://arxiv.org/abs/2604.17148.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet, 2025a. URL https://arxiv.org/abs/2502.04180.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. Aflow: Automating agentic workflow generation, 2025b. URL https://arxiv.org/abs/2410.10762.

Qizheng Zhang, Michael Wornow, Gerry Wan, and Kunle Olukotun. Agentic plan caching: Test-time memory for fast and cost-efficient llm agents, 2026a. URL https://arxiv.org/abs/2506.14852.

Yao Zhang, Chenyang Lin, Shijie Tang, Haokun Chen, Shijie Zhou, Yunpu Ma, and Volker Tresp. Swarmagentic: Towards fully automated agentic system generation via swarm intelligence, 2025c. URL https://arxiv.org/abs/2506.15672.

Yao Zhang, Shijie Tang, Zeyu Li, Zhen Han, and Volker Tresp. Webarbiter: A principle-guided reasoning process reward model for web agents, 2026b. URL https://arxiv.org/abs/2601.21872.

Yuzhe Zhang, Feiran Liu, Yi Shan, Xinyi Huang, Xin Yang, Yueqi Zhu, Xuxin Cheng, Cao Liu, Ke Zeng, Terry Jingchen Zhang, and Wenyuan Jiang. Silo-bench: A scalable environment for evaluating distributed coordination in multi-agent llm systems, 2026c. URL https://arxiv.org/abs/2603.01045.

Zhen Zhang, Wanjing Zhou, Juncheng Li, Hao Fei, Jun Wen, and Wei Ji. Radar: Redundancy-aware diffusion for multi-agent communication structure generation, 2026d. URL https://arxiv.org/abs/2605.09907.

Mingming Zhao, Xiaokang Wei, Yuanqi Shao, Kaiwen Zhou, Lin Yang, Siwei Rao, Junhui Zhan, and Zhitang Chen. a^2flow : automating agentic workflow generation via self-adaptive abstraction operators, 2025. URL https://arxiv.org/abs/2511.20693.

Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Ö. Arık. Multi-agent design: Optimizing agents with better prompts and topologies, 2026. URL https://arxiv.org/abs/2502.02533.

Rongzhi Zhu, Xiangyu Liu, Yi Liu, Shuo Zhang, Ruirui Zhang, Rui Wu, Tao Jiang, Zequn Sun, Wenhao Xu, and Wei Hu. Dense process supervision for search agents via fact utility estimation, 2026. URL https://arxiv.org/abs/2609.00833.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Language agents as optimizable graphs, 2024. URL https://arxiv.org/abs/2402.16823.

Algorithm 1 INFLOWOP $(Q, \mathcal{A})$: The Flow as Built, the Flow as Run (§3, §D.1)

Require: task Q; agent pool $\mathcal{A}$

One estimator serves both stages: $\mathrm{match}(\textit{demand}, \textit{supply})$ *prices pairings into* C*, and scores declared conditions against realized states in flow. Stage II inherits* C *the workflow with contracts from Stage I.*

Ensure: the final answer y and the workflow $W^\dagger$ that produced it

Stage I: Workflow Construction (§3.1; Alg. 3)

$C(s,a) \leftarrow -\log \mathrm{match}(s,a)$; $\quad \rho(s) \leftarrow \mathbb{1}\big[\min_a C(s,a) > \bar{C}\big]$ ▷ *cost matrix*; where the pool falls short (Eq. 16, 1)

$W^\star, \{(\sigma^i_{\text{in}}, \sigma^i_{\text{out}})\} \leftarrow \textsc{Coalesce}\big(Q, \mathcal{A} \mid C\big)$ ▷ C settles granularity and agent assignment (Eq. 2)

Stage II: In-Flow Optimization (§3.2; Alg. 4)

for all $s_i \in W^\star$ **do** ▷ in topological order

  $(\mathcal{I}_i, \mathcal{O}_i) \leftarrow \textsc{Execute}\big(s_i, a(s_i)\big)$

  $u(\sigma, x) \leftarrow 1 - \mathrm{match}(\sigma, x)$ ▷ *credit matrix*: $x = \mathcal{I}_i$ for σ^i_{in} and $x = \mathcal{O}_i$ for σ^i_{out} (Eq. 3)

  $b \leftarrow \mathrm{blame}(s_i)$ ▷ which side of the contract is breached (Eq. 4)

  if $b \neq$ *none* **then**

    $\textsc{Pause}(s_i)$ ▷ halt at the fault, not at the flow

    $W' \leftarrow \textsc{Temper}\big(W, s_i, b \mid C, \sigma^i\big)$ ▷ $b =$ *assignment* (Eq. 20) or $b =$ *decomposition* (Eq. 19)

    $W \leftarrow W'$ **iff** $u(\sigma, \cdot) < \bar{u} \ \ \forall \sigma$ ▷ else roll back: never worse than built

    $\textsc{Resume}\big(I(s_i)\big)$ ▷ only the fault is paid for (Eq. 21)

  end if

end for

return $y,\ W^\dagger$ ▷ Eq. 5

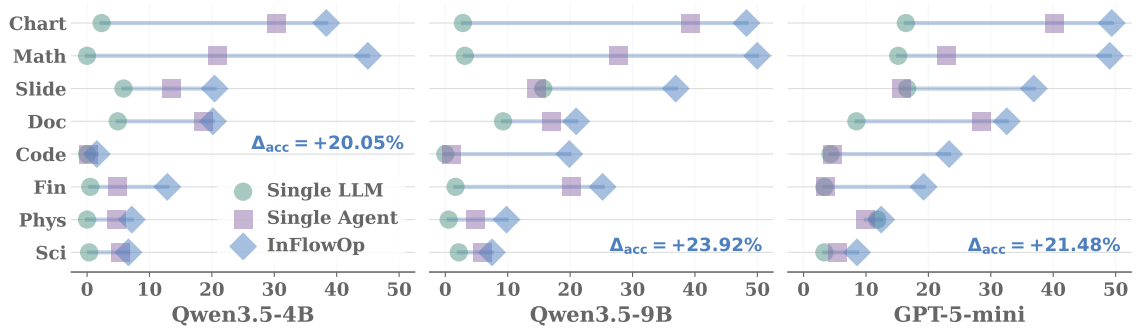

Figure 6: **Evaluation on All Eight Domains of BRAID.** Mean accuracy (%) per domain for the three models evaluated on the whole benchmark. Each row shows a domain under the two baselines and INFLOWOP (§5.1). The horizontal axis is accuracy (%). Tab. 1 covers the five domains every model is evaluated on.

Agent Card

Every agent is represented to the estimator by a card, the sole evidence an atom is scored against (1) *declared competence*: a written brief of what the agent is designed for, including its role, domain, and the skills and tools it commands; and (2) *self-evolving empirical profile*: the record an agent accumulates through execution, profiling which kinds of work it has realized successfully and where it has faltered. So a declared competence is progressively complemented by one that is observed.

A Related Work

A.1 Workflow Construction over an Agent Pool

Compared to hand-designed multi-agent systems that fix the roles and their order in advance (Hong et al., 2024; Li et al., 2026; Guo et al., 2026), automatic construction searches for the effective structure rather than authoring it, within a design space of operators, connections, or collaboration structures enumerated in advance (Zhang et al., 2025b; Zhao et al., 2025; Yue et al., 2026; Li et al., 2025a). That design space, rather than the agent pool, settles how finely the task is divided: prompted decomposition takes its grain from exemplars or instructions (Liu et al., 2025; Madhwal et al., 2026; Wang et al., 2023b), while plan-then-assign pipelines fix the parts before choosing which agent runs each one (Shen et al., 2023; Dong et al., 2026; Kapoor et al., 2026). Where the grain adapts, it adapts to the query alone (Su et al., 2026), or revises a division already proposed (Prasad et al., 2024; Wang et al., 2025). Cost enters some of these decisions, in recent work alongside how far the task is divided (Zhang et al., 2025a; Cui et al., 2026; Xu et al., 2026), but *the quantity that prices a choice is fitted before the task and refitted whenever the set of available operators changes*.

Which agent carries out each part divides along a second axis. Some methods search over a pool they leave fixed, so a part that no agent fits well still goes to the closest one available (Zhuge et al., 2024; Yuan et al., 2025). Another line of work adds agents that the pool does not hold, so the same part instead yields an agent created for it (Zhang et al., 2025c; Wu et al., 2025). Where a system offers both, it alternates between them or settles the choice by a test of similarity, rather than weighing one against the other (Shang et al., 2025; Gupta et al., 2026; Zhang et al., 2026a). As for pricing, where it appears, it ranges over backbone models rather than over reuse and creation (Yang et al., 2025a). *What is missing is a single price over the pool as it stands, charged per part before execution, under which creating an agent is one option inside the same search that sets the grain*.

A.2 Optimizing a Workflow after Execution

Recent work optimizes workflow performance in different ways. One route improves a workflow by training: the constructor or planner is fine-tuned against scored outcomes (Li et al., 2026; Peng et al., 2026; Nielsen et al., 2026; Dang et al., 2025). Step-level signals are obtained the same way, with process reward models and learned verifiers fitted on annotated steps or on rollouts graded against known answers (Zhang et al., 2026b; Chae et al., 2025; Zhu et al., 2026; Lee et al., 2026a). *Improvement is therefore paid for before the task arrives, in data and compute*. A second route requires no training but re-runs instead, sampling the workflow repeatedly, searching for a better one, or re-executing after a failure (Hwang et al., 2026; Li et al., 2025b; Yoon et al., 2026; Lee et al., 2026b). *Each increment of quality costs another execution of the entire workflow, while what needs correcting is only a small part of it*. A third route is both training-free and label-free: the system criticizes its own output and revises it (Shi et al., 2026; Yang et al., 2025b; Ning et al., 2026; Wan et al., 2026). *Yet what it revises is the flow output, without identifying which part inside the flow is responsible*. When blame is assigned within a flow, it is scored on a completed trace against annotated ground truth (Chen et al., 2026b), recovered by replaying a trace that has already finished (Hou et al., 2026; Lin et al., 2026), or derived from a relation the domain itself makes checkable, such as whether a claim follows from the sources cited for it (Hirsch et al., 2026). *What is missing is a fault located from a single execution as it runs, with the correction confined to the part at fault, under no training, no reference answer, and no graded outcome*.

A.3 Evaluating Multi-Agent Workflows

Workflow systems are measured on the suites built for single models: mathematical reasoning, code generation, and question answering (Zhang et al., 2025b; 2026d; Yun et al., 2026; Zhou et al., 2026). On these, repeated sampling from one strong agent captures much of what coordination adds, and under matched budgets a coordinating team can fall behind the same models working alone (Ann et al., 2026; Sun et al., 2026; Choi

et al., 2025; Jwalapuram et al., 2026). *A benchmark that one agent already solves cannot show what dividing the work contributes.*

Benchmarks that do stress coordination exist, built around long-horizon tool use, siloed state, or distributed evidence (Gao et al., 2026; Zhang et al., 2026c; Chen et al., 2026a). In these, the need to divide is imposed by the environment rather than arising from what the task demands: state is partitioned, tools are split between agents, or knowledge is withheld from one of them. *What is missing is a benchmark whose tasks require the division on their own terms, with parts that call for different competences and a result that depends on how those parts combine.*

B LIMITATIONS

INFLOWOP is evaluated on BRAID, whose tasks span eight domains and are graded against a reference answer for each question (§4). Tasks settled by no checkable answer, such as open-ended generation or long-horizon interaction with a stateful environment, lie outside this study. In future work, we plan to extend INFLOWOP to such settings and different domains. We also plan to investigate workflows whose agents act on an environment that changes as they run.

C WHERE A WORKFLOW BECOMES NECESSARY: BRAID FOR WORKFLOW-LEVEL EVALUATION

What a workflow contributes is coordination, especially how the work is divided among agents and how those divisions depend on one another. Therefore, measuring the contribution of a workflow requires complex tasks whose demands exceed what one agent can meet. However, existing benchmarks rarely supply such task complexity: their tasks are simple enough for one competent agent to solve alone. Consequently, a workflow evaluated on them is measured on tasks that require neither task decomposition nor multi-agent coordination beyond single-agent competence (§1, A.3, C.1). To close this gap, we propose BRAID, a benchmark for workflow-level agent-interdependent reasoning (**[Xuehang]** cite section4 data section after adding in main paper). Concretely, BRAID composes every task through two composition methods under quality control (§C.2), and measures the effectiveness of a workflow by the mean accuracy over the questions each task comprises (§C.3).

C.1 WHY WORKFLOW-LEVEL EVALUATION IS NECESSARY

When a workflow earns its necessity. A workflow is worth building only where a single agent cannot effectively carry the task alone, yet workflows are seldom evaluated on tasks that reach such complexity (§1, A.3). As shown in Fig. 7, we conduct a preliminary test on this gap, comparing a single agent solving the task in one pass with a workflow an LLM constructs for the same task: when evaluated on the original data, the single agent is more accurate than the workflow (-12.02%) with notably lower cost (-6.91). Therefore, coordination is not a necessity for simple tasks, and running multi-agent workflows on them can further incur several times the cost. On the other hand, with BRAID composition (§C.2), the ordering reverses: the single agent degrades significantly on such complex tasks, while the workflow, running under a matched compute budget, showcases the strength of task decomposition and multi-agent coordination with better performance. Workflow-level task complexity is therefore a necessity for workflow evaluation, and the objective BRAID targets.

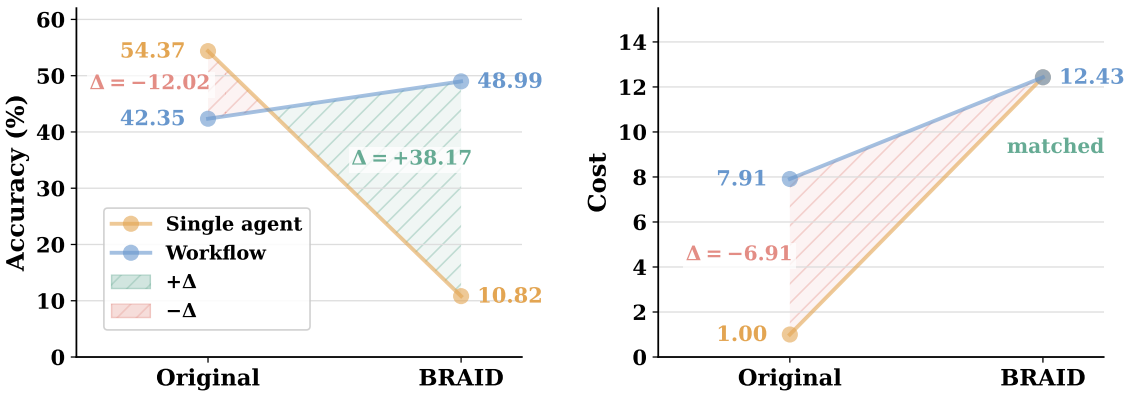

Figure 7: **A Workflow Earns Its Necessity Only Where One Agent Falls Short.** The shaded band marks the gap between the two solvers in accuracy (left) and in cost (right). Cost is counted in units of one single-agent run on the original data, and on BRAID the two solvers run under matched budgets. Both use `gpt-5-mini` over 150 samples of *SlideVQA* (*doc* variant of BRAID), *Distributed* at complexity level $k = 5$.

Why workflow-level evaluation is necessary. The benchmarks a multi-agent workflow is usually measured on sit on the left side of each subplot (Fig. 7: left *original* half of both subplots): a single competent agent is already able to solve their tasks, which therefore demand neither task decomposition nor coordination among agents. As a result, evaluating a workflow on them cannot effectively measure how a workflow performs and what it actually contributes. Instead, reliable workflow-level evaluation requires tasks whose complexity demands multi-agent coordination beyond single-agent capability.

C.2 BENCHMARKING WORKFLOW-LEVEL EVALUATION

BRAID constructs workflow-level tasks by composing each task with k questions and providing extra sources beyond those its questions require, without stating which source serves which question. We use k to denote the *complexity level* of a task. Answering one task therefore demands both halves of what a workflow is for: *dividing* the task, and reaching its answers through coordinated *routing*. Here, we define *routing* as locating the source a question is answered from and reasoning over it to that question's answer. We use *solver* to denote an evaluated system in general: a single LLM, an agent, or a multi-agent workflow that receives the task and produces its answers.

Composition methods. A task is composed in one of two methods: *distributed* or *anchored*. A *source* here is one document or image provided with the task, however many files it takes to provide it, and the source a question is answered from is its *gold source*. They differ in where the gold sources of a task's questions sit and therefore in what the *division* and the *routing* each demand:

1. ***Distributed*: every question has its own source.** Each *distributed* task takes k questions that are each answered from a different source. BRAID provides those k sources, and additionally inputs $(k-1)$ more sources that answer none of the questions, which we call *distractors*. As a result, a *distributed* task inputs $(2k-1)$ sources in total.

② ***Anchored*: all questions share one source.** An *anchored* task takes k questions that are all answered from one source. BRAID provides that source, and additionally inputs $(k-1)$ distractors alongside it. As a result, an *anchored* task inputs k sources in total.

Source selection. Distractors come from inside BRAID, and the kind of input decides which extra sources supply them. A dataset whose inputs are documents draws its distractors from its own split, so a distractor is the same kind of document as the gold source and differs only in what it contains. A dataset whose inputs are images draws from a different dataset of the same kind, which keeps routing real via strict quality control while ensuring that a gold source's own family never supplies a distractor (Alg. 2). A dataset whose items provide no input of their own draws its distractors from charts. The draw is randomly seeded and prefers the least-used sources of the split, so no source dominates the benchmark and the same composition is reproduced on every rebuild.

Quality control. A source counts as a *distractor* only if it answers no question of the task: when any other provided source also answers a question, that question has more than one gold source, and thus the task can no longer isolate the *routing* it is built to evaluate. BRAID therefore rejects such tasks while composing them (Alg. 2) rather than filtering them after construction, and a task that fails any of the four criteria below is never written. (1) *Separation* keeps no provided source standing in for another: two questions of one task do not rest on near-duplicate gold sources, and a distractor resembling any of the task's gold sources is redrawn. (2) *Routing* requires each question to mention at least one word or phrase that appears in its own gold source and in no other provided source, so that the question can identify that source among those provided. (3) *Leakage* forbids any source other than a question's gold source from containing that question's answer. (4) *Answerability* covers what text matching cannot decide: where a task supplies images or documents, an LLM judges every pairing of a question with a provided source, and the task is discarded if any question's answer holds against a source other than its gold source. Beyond these four criteria, file names are also controlled for the same reason: the provided sources are listed in random order with file names revealing nothing, so a solver can identify the gold source only from its content, never from its file name.

Dataset coverage. BRAID is constructed upon ten public datasets spanning eight domains (Ma et al., 2024; Deng et al., 2025; Tanaka et al., 2023; Reddy et al., 2025; Wang et al., 2024; Tang et al., 2026; Hendrycks et al., 2021; He et al., 2024; and Phan et al., 2026; Li et al., 2023), including documents, slides, finance, charts, mathematics, physics, science, and code. A dataset enters BRAID with whichever *composition methods* and *complexity levels* it supports, and each pairing of a BRAID dataset with a composition method forms one *arm*, evaluated at both complexity levels. *Distributed* demands only that each question have its own gold source, which every original dataset supplies, while *anchored* requires a source that carries at least k questions, which the document and slide datasets provide and the others cannot: a dataset whose items are one question per figure, one problem per statement, or a prompt with no provided input has no gold source to share. A complexity level is kept only where it yields at least 80 questions. For example, DocFinQA at $k = 5$ is dropped for size: each of its filings carries too few questions to compose tasks at that complexity level in sufficient number, giving 5 questions under *distributed* and 40 under *anchored*, both far below 80. Every other absence is a matter of constructibility, since the chart, mathematics, physics, science, and code datasets hold at most a handful of questions per source, and so cannot anchor k of them. What remains spreads difficulty across two axes. Across datasets, the provided sources range from documents and slides to charts and text-only samples, and the answer types from numeric through open-ended to code generation. Within a dataset, raising the complexity level from $k = 3$ to $k = 5$ carries the same source data into longer tasks, and under *distributed* from five provided sources to nine.

Data statistics. As summarized in Tab. 3, BRAID constructs nineteen arms over fourteen BRAID datasets, adapted from ten public datasets, across eight domains. BRAID builds workflow-level samples at two complexity levels, counted in questions rather than tasks because accuracy is measured over questions. Every

Algorithm 2 COMPOSE$(\mathcal{S}, k)$: BRAID Task Construction (§C)

Require: the questions and sources $\mathcal{S}$ of one dataset split; complexity level k; composition *distributed* or *anchored*
Notation: $\mathrm{gold}(q_i)$ *is the source* q_i *is answered from,* y_i *is its answer,* $w \in q_i$ *is a word or phrase of its text,* $\mathrm{sim}(\cdot,\cdot)$ *is source similarity with bar* θ*, and* $\mathrm{judge}(q_i, z) \in \{0,1\}$ *is an LLM judge on whether* y_i *holds against* z

Ensure: task $Q = \langle \{q_1, \ldots, q_k\},\ E \cup N \rangle$, or DISCARD

$\{q_1, \ldots, q_k\} \leftarrow k$ questions of the split **s.t.** $|\{\mathrm{gold}(q_i)\}| = k$ (*distributed*) **or** 1 (*anchored*)
$E \leftarrow \{\mathrm{gold}(q_i)\}$; $N \leftarrow \textsc{Draw}\big(\mathcal{S} \setminus E,\ k-1\big)$ ▷ *gold sources*; *distractors*, drawn seeded and least-used first
(1) ***Separation****: no provided source stands in for another*
if $\exists\, e \neq e' \in E :\ \mathrm{sim}(e, e') \geq \theta$ **then**
  return DISCARD ▷ one question's gold source would answer another
end if
for all $d \in N$ **with** $\max_{e \in E} \mathrm{sim}(d, e) \geq \theta$ **do**
  $d \leftarrow \textsc{Draw}\big(\mathcal{S} \setminus (E \cup N),\ 1\big)$, **or return** DISCARD ▷ redraw until clean, else drop
end for
(2)-(4) ***Answerability****: exactly one provided source answers each question*
for all q_i **with** $Z_i \leftarrow (E \cup N) \setminus \{\mathrm{gold}(q_i)\}$ **do**
  if $\nexists\, w \in q_i :\ w \in \mathrm{gold}(q_i) \ \wedge\ w \notin z\ \ \forall z \in Z_i$ **then**
    return DISCARD ▷ (2) *routing*: no word identifies $\mathrm{gold}(q_i)$ among $E \cup N$
  end if
  if $\exists\, z \in Z_i :\ y_i \subseteq z$ **then**
    redraw z **if** $z \in N$, **else return** DISCARD ▷ (3) *leakage*
  end if
  if $E \cup N$ carries images or documents **and** $\exists\, z \in Z_i :\ \mathrm{judge}(q_i, z) = 1$ **then**
    return DISCARD ▷ (4) *answerability*, where $y_i \subseteq z$ cannot be decided by text
  end if
end for
return $Q \leftarrow \langle \{q_1, \ldots, q_k\},\ \textsc{Shuffle}(E \cup N) \rangle$ ▷ random order, file names carrying no signal

arm is evaluation-only and inherits the test side of the dataset it is composed from, so no BRAID task shares a row with any training split.

C.3 EVALUATION METRICS

Task Accuracy. The accuracy of a task Q is the average score of its k questions, and the accuracy of a dataset $\mathcal{Q}$ is the average over its tasks. Let $\hat{y}_i$ denote the predicted answer for q_i, y_i denote its gold answer, and t_i denote its answer type ($t_i \in \{$*numeric*, *math*, *text*, *choice*, *list*, *code*$\}$):

$$\mathrm{Acc}(Q) = \frac{1}{k} \sum_{i=1}^{k} \mathrm{score}_{t_i}\big(\hat{y}_i,\ y_i\big), \qquad \mathrm{Acc}(\mathcal{Q}) = \frac{1}{|\mathcal{Q}|} \sum_{Q \in \mathcal{Q}} \mathrm{Acc}(Q) \tag{6}$$

Numeric Answers. A numeric answer is scored on its value rather than on its string, and a dataset whose golds are measured quantities may admit a relative tolerance ε:

$$\mathrm{score}_{\textit{numeric}}\big(\hat{y}, y\big) = \mathbb{1}\big[|\hat{y} - y| \leq \varepsilon |y|\big] \tag{7}$$

Mathematical Answers. A mathematical answer is scored under symbolic equivalence $\equiv$, so an answer differing from the gold in notation, ordering, or simplification still counts:

$$\mathrm{score}_{\textit{math}}(\hat{y}, y) = \mathbb{1}\big[\hat{y} \equiv y\big] \tag{8}$$

Textual Answers. A textual answer counts in full when it matches the gold after normalization $\mathrm{norm}(\cdot)$, and otherwise earns the token-level F_1 between the two. Thus, an answer carrying part of the gold is credited in part:

$$\mathrm{score}_{\textit{text}}(\hat{y}, y) = \begin{cases} 1 & \text{if } \mathrm{norm}(\hat{y}) = \mathrm{norm}(y) \\ \mathrm{F}_1(\hat{y}, y) & \text{otherwise} \end{cases} \tag{9}$$

A dataset whose golds are short spans of a long source may admit the relaxed form, in which a gold appearing inside the answer also counts in full, and where a dataset accepts several golds the answer takes its best score among them.

Multiple-Choice Answers. A multiple-choice answer is scored on the option it selects, which is extracted from the answer before comparison:

$$\mathrm{score}_{\textit{choice}}(\hat{y}, y) = \mathbb{1}\big[\hat{y} = y\big] \tag{10}$$

List Answers. A list answer is scored position by position against the gold sequence, so an answer right in part is credited in part:

$$\mathrm{score}_{\textit{list}}(\hat{y}, y) = \frac{1}{|y|} \sum_{j=1}^{|y|} \mathbb{1}\big[\hat{y}_j = y_j\big] \tag{11}$$

Coding Answers. A coding answer is scored by execution against the tests $\mathcal{V}$, and counts only when every test passes:

$$\mathrm{score}_{\textit{code}}(\hat{y}, y) = \mathbb{1}\big[\mathrm{pass}(\hat{y}, v) \;\; \forall v \in \mathcal{V}\big] \tag{12}$$

Per-Question Grading. Every question of a task is graded separately according to its answer type, so the accuracy of a task rises with each question whose routing succeeds. Two answer types are graded by degree within a single question as well: a textual answer that does not match its gold falls back to token overlap (Eq. 9), and a list answer is credited position by position (Eq. 11). Particularly, we grade per question rather than scoring a task as a single unit. This is because a single-unit score counts only when all k of its questions are correct, which therefore assigns the same value to a solver that answers most questions of a task and to one that answers none. Such a score fails to effectively measure the actual performance and competence of a solver in workflow-level problem-solving.

D Beneath the Flow: Complementary Details and Discussions

D.1 InFlowOp Overview: From the Flow as Built to the Flow as Run

InFlowOp performs a task in two stages (Alg. 1): *build* the workflow and *run* the workflow. The first *workflow construction stage* turns the task into a workflow: it prices every atom against every agent into the cost matrix, leveraging the cost matrix to determine how finely the task is divided and who carries each divided subtask, and hands every subtask the contract its atoms declare. The *in-flow optimization stage* runs that workflow and corrects it as it runs: it matches each realized subtask against its contract by reusing the estimator in the first stage, blames whichever side breaches, and tempers the workflow there locally. The two stages share one estimator and one matrix, which is what lets a correction be priced in the same currency the workflow was built with.

Stage I: Construction (Alg. 3). The task is first decomposed to its finest atoms, and the atoms the goal never needs are discarded. COALESCE proceeds over partial groupings, extending each by its next atom, which either opens a new subtask or joins one already open. A grouping is pruned once its lower bound reaches the best cost found so far. Each subtask of the returned atom grouping is assigned with its lowest-cost agent. When no agent meets the competence bar, a new agent is created at the creation penalty. The atom dependencies then carry over to the subtask graph, and every subtask leaves this stage with its contract. This then supports Stage II for label-free in-flow optimization.

Stage II: In-Flow Optimization (Alg. 4). Execution proceeds subtask by subtask, and every realized input and output is retained. A subtask that yields no usable output is faulty by observation. Otherwise, its realized work is matched against its contract, and which side breaches blames the fault. A fault blamed on the assignment is corrected by the next agent the cost matrix ranks, and one blamed on the decomposition by re-decomposing the subtask under the same contract (Eq. 19). A correction is kept only if the subtask then executes and its unmet conditions are satisfied, and is undone otherwise. Tempering therefore never leaves a workflow worse than the one Stage I produced. Tempter is confined to the the local subtask (Eq. 21) with every progress achieved so far intact.

Cost-Driven Pruning. COALESCE expands a frontier $\mathcal{F}$ of partial coalescings and retains the best complete one found, the incumbent $P^\star$. Since Cost (Eq. 18) only accumulates as atoms are placed, any completion of a partial coalescing P' is bounded from below by the lower bound $\mathrm{lb} = \mathrm{Cost}(P') + \textsc{OptimisticRemainder}(P')$, where the remainder is the cheapest cost the unplaced atoms could possibly incur. As that remainder never over-estimates, a partial coalescing whose lb already reaches $\mathrm{Cost}(P^\star)$ can complete no better and is pruned, without risk of discarding the optimum.

D.2 META PRIMITIVES

COALESCE may emit four structures. Throughout, $\hat{p}(s,a)$ is the probability that agent a realizes subtask s (§3.1), $C(s,a) = -\log \hat{p}(s,a)$, and $\ell(s,a)$ its latency.

AND for *conjunction*: atoms are grouped into one subtask, and subtasks with no dependency path between them are left unordered. Its reliability and latency are Eq. 16 and Eq. 17 unchanged.

REPEAT for *bounded repetition*: a subtask is re-run in place until its output is usable, at most B_{rep} times.

$$R_{\text{REP}} = -\log\left(1-(1-\hat{p})^{B_{\mathrm{rep}}}\right), \qquad L_{\text{REP}} = \ell \sum_{r=1}^{B_{\mathrm{rep}}} (1-\hat{p})^{\,r-1} \tag{13}$$

OR for *conditional alternatives*: one required output is pursued by J independent methods and consolidated.

$$R_{\text{OR}} = -\log\left(1-\prod_{j=1}^{J}\left(1-\hat{p}_j\right)\right), \qquad L_{\text{OR}} = \max_{j\le J} L_j \tag{14}$$

SELECT for *branched selection*: exactly one of K mutually exclusive continuations is taken under a branch test.

$$R_{\text{SEL}} = \sum_{j=1}^{K} \omega_j R_j, \qquad L_{\text{SEL}} = \max_{j\le K} L_j \tag{15}$$

where R_j and L_j are the reliability and latency of continuation j, and ω_j its selection weight.

Algorithm 3 COALESCE$(Q, \mathcal{A})$: Bidirectional Cost-Driven Atomic Coalescing (§3.1)

Require: task Q, agent pool $\mathcal{A}$, trade-off β, creation penalty γ, competence bar $\bar{C}$, expansion budget B
Ensure: workflow $W = \langle S, A, G \rangle$ of least cost (Eq. 2)

Direction 1: *top-down atomic decomposition*

$\mathcal{T}, G_{\mathcal{T}} \leftarrow$ ATOMIZE(Q) ▷ atoms $\{\tau_1, \ldots, \tau_n\}$ and their data-flow DAG
$\nu \leftarrow \{\tau \in \mathcal{T} : \tau$ back-reachable from the goal over $G_{\mathcal{T}}\}$ ▷ drop atoms the goal never needs

Direction 2: *bottom-up cost-driven coalescing*

$C, \ell, \rho \leftarrow$ COSTMATRIX$(\nu, \mathcal{A})$ ▷ label-free estimator on singleton atoms
$P^\star \leftarrow \{\{\tau\} : \tau \in \nu\}$; $\mathcal{F} \leftarrow \{\varnothing\}$ ▷ incumbent is the finest cut, one subtask per atom; frontier to extend
while $\mathcal{F} \neq \varnothing$ **and** explored nodes $< B$ **do**
  $P \leftarrow$ POP$(\mathcal{F})$ ▷ a partial coalescing; τ_i is its next atom in topological order
  for all $\mu \in \{$OPEN$(\tau_i)\} \cup \{$MERGE$(\tau_i, s) : s \in P$ open$\}$ **do** ▷ open or merge (§3.1)
    $P' \leftarrow \mu(P)$ ▷ the resulting coalescing; s is the subtask changed by μ
    $C(s, \cdot), \ell(s, \cdot), \rho(s) \leftarrow$ COSTMATRIX$(\{s\}, \mathcal{A})$ ▷ re-score only the changed subtask
    if $\rho(s) = 1$ **then** ▷ no pooled agent covers s (Eq. 1)
      $a_+ \leftarrow$ CREATEAGENT(s); $\mathcal{A}_+ \leftarrow \mathcal{A}_+ \cup \{a_+\}$ ▷ charged γ in Cost
    end if
    $a(s) \leftarrow \arg\min_{a \in \mathcal{A} \cup \mathcal{A}_+} C(s, a)$ ▷ bind s to its lowest-cost covering agent
    $\mathrm{lb} \leftarrow \mathrm{Cost}(P') +$ OPTIMISTICREMAINDER(P') ▷ lower bound (lb): cost committed so far
    if $\mathrm{lb} < \mathrm{Cost}(P^\star)$ **then**
      $\mathcal{F} \leftarrow \mathcal{F} \cup \{P'\}$; **if** P' is complete **then** $P^\star \leftarrow P'$
    end if ▷ lb never over-estimates, so pruning cannot discard the optimum
  end for
end while

Emit

$S, A \leftarrow$ the subtasks of $P^\star$ and their assignments
$G \leftarrow$ LIFT$(G_{\mathcal{T}}, S)$ ▷ lift the atom DAG to the subtask level
for all $s_i \in S$ **do**
  $\sigma^i_{\mathrm{in}}, \sigma^i_{\mathrm{out}} \leftarrow$ input/output conditions of s_i's member atoms ▷ contract checked in §3.2
end for
return $W = \langle S, A, G \rangle$

Subroutine: *cost matrix*

function COSTMATRIX$(S, \mathcal{A})$
  for all $s \in S$ **and** $a \in \mathcal{A}$ **do**
    $\hat{p}(s, a) \leftarrow \mathrm{match}(s, a)$ ▷ match the subtask's demands against the agent's card
    $C(s, a) \leftarrow -\log \hat{p}(s, a)$; $\ell(s, a) \leftarrow$ latency of a on s ▷ Eq. 16
  end for
  for all $s \in S$ **do**
    $\rho(s) \leftarrow \mathbb{1}\left[\min_a C(s, a) > \bar{C}\right]$ ▷ coverage residual (Eq. 1): nothing in the pool fits s
  end for
  return C, ℓ, ρ
end function

D.3 TAXONOMY OF COST

This section complements our cost taxonomy definition in §3. A constructed workflow is charged along two axes: the *reliability cost* R of entrusting each subtask to an agent that may not realize it correctly, and the *latency cost* L of running the agents it assigns. Both follow from the same label-free matching of what a subtask demands against what an agent supplies (§3.1), so a workflow is priced before it runs. Together with

Algorithm 4 INFLOWOP(W, C): In-Flow Dynamic Optimization (§3.2)

Require: workflow $W = \langle S, A, G \rangle$ with contracts $\{(\sigma^i_{\text{in}}, \sigma^i_{\text{out}})\}$, cost matrix C, unmet bar $\bar{u}$, re-assign budget B_{ra}, re-decompose B_{rd}
Ensure: final answer, and the tempered workflow $W^\dagger$ (Eq. 5)

Mem $\leftarrow \varnothing$; $\mathcal{X} \leftarrow \varnothing$ ▷ realized results; breached pairings
Local: *detect and temper in flow*
for all $s_i \in S$ in topological order of G **do**
  if $s_i \in$ Mem **then** ▷ RESUME: a valid result is never recomputed
    continue
  end if
  $(\mathcal{I}_i, \mathcal{O}_i) \leftarrow$ EXECUTE$(s_i, a(s_i))$; Mem$[s_i] \leftarrow (\mathcal{I}_i, \mathcal{O}_i)$
  $b \leftarrow$ DETECT$(s_i, \mathcal{I}_i, \mathcal{O}_i)$
  if $b \neq$ *none* **then**
    PAUSE(s_i) ▷ halt at the fault, not at the flow
    $W \leftarrow$ TEMPER(W, s_i, b); Mem $\leftarrow$ Mem $\setminus I(s_i)$ ▷ invalidate only the closure (Eq. 21)
  end if
end for
return final answer; $W^\dagger \leftarrow W$
Subroutine: *detection*
function DETECT$(s_i, \mathcal{I}_i, \mathcal{O}_i)$
  if $\mathcal{O}_i$ unusable **then** ▷ causal signal: faulty by observation, no matching needed
    return *assignment*
  end if
  $U_{\text{in}} \leftarrow \{\sigma \in \sigma^i_{\text{in}} : u(\sigma, \mathcal{I}_i) \geq \bar{u}\}$; $U_{\text{out}} \leftarrow \{\sigma \in \sigma^i_{\text{out}} : u(\sigma, \mathcal{O}_i) \geq \bar{u}\}$ ▷ credit matrix (Eq. 3)
  return blame(s_i) given by $U_{\text{in}}, U_{\text{out}}$ ▷ Eq. 4
end function
Subroutine: *tempering along the cost-ordered ladder*
function TEMPER(W, s_i, b)
  if $b =$ *assignment* **then** ▷ RE-ASSIGN: the cheaper rung (Eq. 19)
    for $b = 1$ **to** B_{ra} **do**
      $a' \leftarrow \arg\min_{a \notin \mathcal{X}(s_i)} C(s_i, a)$; $\mathcal{X}(s_i) \leftarrow \mathcal{X}(s_i) \cup \{a'\}$ ▷ Eq. 20
      $(\mathcal{I}_i, \mathcal{O}_i) \leftarrow$ EXECUTE(s_i, a')
      if DETECT$(s_i, \mathcal{I}_i, \mathcal{O}_i) =$ *none* **then** ▷ commit only on a clean pass
        $a(s_i) \leftarrow a'$; **return** W
      end if
    end for
    restore $a(s_i)$ ▷ no unverified change survives
  end if
  for $b = 1$ **to** B_{rd} **do** ▷ RE-DECOMPOSE: climb, no agent can satisfy s_i as scoped
    $\langle S_{\text{sub}}, G_{\text{sub}} \rangle \leftarrow$ COALESCE$\big(s_i \mid \sigma^i_{\text{in}}, \sigma^i_{\text{out}}\big)$ ▷ Alg. 3, re-cut under the same contract
    $W \leftarrow$ SPLICE$(W, s_i, \langle S_{\text{sub}}, G_{\text{sub}} \rangle)$; **return** W ▷ the contract keeps the splice local
  end for
  return W ▷ budget spent: keep the original
end function

the price of each agent created for it, they compose the *workflow cost* Cost(W) that COALESCE minimizes (Eq. 2). The *optimization cost* of the taxonomy stays outside Cost(W), as what improving a workflow charges rather than what running one charges (§E).

Reliability cost. A workflow succeeds only when its subtasks are each realized correctly. We model the realization of a subtask s_i by its assigned agent $a(s_i)$ as a success with probability $\hat{p}\big(s_i, a(s_i)\big) \in (0, 1]$. As subtask realizations are independent, a workflow's success probability factorizes as $\prod_{s_i} \hat{p}(s_i, a(s_i))$. Thus, maximizing it is equivalent to minimizing its negative logarithm:

$$R(W) = -\log \prod_{s_i \in S} \hat{p}\big(s_i, a(s_i)\big) = \sum_{s_i \in S} C\big(s_i, a(s_i)\big), \qquad C(s, a) := -\log \hat{p}(s, a) \tag{16}$$

where $C(s, a)$ is the *reliability cost* of realizing s with a, and $R(W)$ is the workflow's total reliability cost.

Latency cost. Each subtask also incurs a latency $\ell\big(s_i, a(s_i)\big)$, the cost of running its agent. Because G admits parallelism, a workflow's latency is not the sum of these but its *critical path* (§3.1):

$$L(W) = \max_{\text{path } \pi \subseteq G} \sum_{s_i \in \pi} \ell\big(s_i, a(s_i)\big) \tag{17}$$

Parallel subtasks thus cost no more than the slowest among them, so spreading work across concurrent agents lowers L even where it raises R.

Workflow cost. The cost of a workflow combines the two axes, reliability (Eq. 16) and latency (Eq. 17), scalarized by a trade-off weight β, with a penalty γ for each newly created agent (§2):

$$\mathrm{Cost}(W) = \underbrace{R(W)}_{\text{reliability}} + \beta \underbrace{L(W)}_{\text{latency}} + \gamma \, |\mathcal{A}_+(W)| \tag{18}$$

where $\mathcal{A}_+(W)$ is the set of agents created for W, β trades reliability against latency, and γ prices each creation. Sweeping β traces a reliability-latency Pareto frontier: small β favors reliable, finer-decomposed workflows; while large β favors fast, compact ones.

D.4 Cost-Ordered Tempering and Scoped Resumption

Cost-ordered ladder. We define two moves to correct a workflow at a faulty subtask, and the cost model of §3.1 prices both. Re-assign leaves S and G untouched and updates the agent, perturbing a single term of Eq. 16. Re-decompose updates s_i by a finer sub-DAG, changing S and G together by adding subtasks, dependencies, and possibly a created agent, and paying a fresh run of Coalesce besides. Pricing the two in that shared currency orders them:

$$\textsc{Re-assign} \prec \textsc{Re-decompose} \tag{19}$$

which defines the *cost-ordered ladder*: spend the cheapest move the fault admits, and climb only once that move is exhausted. The blame (Eq. 4) fixes the entry rung: a fault blamed on *assignment* enters at Re-assign, one blamed on *decomposition* at Re-decompose.

Temper. At the Re-assign rung, the *cost matrix* answers directly: the next-best agent is the same column-wise minimum construction takes, now restricted to agents not yet observed to fail on s_i,

$$a'(s_i) = \arg \min_{a \in (\mathcal{A} \cup \mathcal{A}_+) \setminus \mathcal{X}(s_i)} C\big(s_i, a\big) \tag{20}$$

where $\mathcal{X}(s_i)$ collects those attempted and ruled out. Re-assign is committed only if s_i executes *and* its unmet conditions clear under Eq. 3; otherwise it is rolled back so that no unverified change survives. When the candidates are exhausted, the evidence has ruled out the assignment as the cause and the ladder climbs: Coalesce is re-invoked on s_i locally, *under its declared contract* $(\sigma^i_{\text{in}}, \sigma^i_{\text{out}})$, and the resulting sub-DAG is spliced in its place. Every move is budgeted, B_{ra} for Re-assign and B_{rd} for Re-decompose, and a

Subtask *declared condition:* the output identifies images that contain required information.

SYSTEM You are the `image_analyzer` agent. Your job is to execute the assigned instruction faithfully and `[...]`

...

USER Which images contain the "annual net gains" that `[...]` *# Inputs:* image 1: `[...]`, image 2: `[...]`, `[...]`.

Before tempering

ASSISTANT `[...]` image 2 (`/workspace/bussiness_page12_crop4.png`) plots "gross income" across 12 months; image 5 (`/workspace/bussiness_page17_crop1.png`) plots market share variations across 2024-2025. Other images show no annual information. So only image 2 and image 5 contain the required information. `[...]`

✓ *condition met* ✗ *final answer wrong*

After tempering

ASSISTANT `[...]` No image can provide the required information. `[...]`

✗ *condition violated* ✓ *final answer correct*

Figure 8: **The Worth of A Step Is Settled Downstream.** (a) Before tempering: the output meets its declared condition and gives two images. However, neither carries the information the subtask demands, and the flow reaches the wrong final answer. (b) After tempering: the output that violates the condition confirms that no image qualifies, and the flow reaches the correct final answer. However, if scored against its own condition, the step that helped is recorded as the step that failed.

subtask that exhausts its budget reverts to its original realization: *tempering can improve a workflow, but never leaves it worse than the one construction produces.*

Scoped resumption. Execution pauses at s_i, and tempering it invalidates only the subtasks that depend on it, so every result computed outside that region remains valid. Execution therefore resumes over the *affected closure* alone:

$$I(s_i) = \{s_i\} \cup \mathrm{Desc}(s_i) \tag{21}$$

where $\mathrm{Desc}(s_i)$ are the subtasks reachable from s_i in G, and replay cached results for every subtask outside $I(s_i)$. The price of an optimization step is thus the cost of the *local* correction only, bounded by $I(s_i)$ rather than by $|S|$. An optional global localization is elaborated in §D.6.

D.5 A Step Is Worth What It Leads To

The Verdict Lies Downstream, Not in the Middle

Whether an intermediate output helps or harms cannot be determined from that output alone. A step may satisfy the conditions it is expected to meet yet still steer the flow wrong, while it may also violate them yet still bridge the flow to the correct final answer. The sign of a deviation is thus settled by where it drives the rest of the flow, not by the step where it appears or by what that step produces. Therefore, no local judgment of that step, however it is computed, recovers it.

Figure 8 shows one such case: the subtask asks which images carry a particular set of information, so its declared condition is that the output identify the useful ones. Before tempering, the agent names two images with wrong evidence but successfully meeting that condition. Thus, neither image carries the information the subtask demands, and the flow continues on wrong evidence that misleads its final answer. After tempering, the agent responds that no image contains what is required. That answer violates the condition, yet it is the accurate one. Consequently, the flow carries on with it and reaches the correct final answer. *However, if scored against its declared condition, the step that helped is recorded as the step that failed.*

We therefore judge every temper by the outcome of the flow it belongs to, without relying on step-level assessment. A step-level metric is not a reliable or stricter measure of the same quantity, as on cases of this kind it carries the opposite sign. For the same reason, we do not evaluate on benchmarks that score intermediate steps against annotated ground truth (Chen et al., 2026b), since such a benchmark penalizes exactly the behavior this example shows to be correct.

D.6 Temper Locally, Look Globally

Local detection (§3.2) acts where a fault breaches the contract of its own subtask. However, a workflow may nonetheless terminate without a usable answer for a cause lying several steps upstream, without violating local contract. Rather than fall back on a global re-search, we extend in-flow optimization with an optional *global localization*: the *temper* stays local, and only *where to look* becomes global. Once the local pass ends without a usable answer, the whole graph is scanned to rank subtasks by *suspicion*:

$$\mathrm{susp}(s_i) = \big(1 - \hat{p}_i\big)\big(1 + |\mathrm{Desc}(s_i)|\big) + \mathbb{1}[s_i \text{ flagged}] \tag{22}$$

where a subtask already flagged by local detection receives an additive bonus. Suspicion prefers subtasks both unreliable and consequential: an unreliable subtask whose output many others consume is a likelier root cause than an unreliable leaf. The most suspicious untried subtask is then handed to the cost-ordered ladder (Eq. 19), committed under the same evidence, and followed by a scoped resumption of its closure (Eq. 21), until a usable answer emerges or the probe budget B_{g} is spent (Alg. 5). Global localization therefore buys only *where to look* at the final failure, while every correction is committed locally. It is disabled in all main experiments, and we evaluate it as an ablation (§5.3).

Algorithm 5 LOCALIZE(W, Mem): Global Localization (§D.6)

Require: workflow W after the local pass of Alg. 4, its realized results Mem, global-probe budget B_{g}

$\mathcal{B} \leftarrow \varnothing$ ▷ probed suspects
while no usable final answer **and** $|\mathcal{B}| < B_{\mathrm{g}}$ **do**
  $s \leftarrow \arg\max_{s \notin \mathcal{B}} \mathrm{susp}(s)$; $\mathcal{B} \leftarrow \mathcal{B} \cup \{s\}$ ▷ Eq. 22: global only localizes
  $b \leftarrow \textsc{Detect}(s, \mathrm{Mem}[s])$ ▷ a suspected but clean subtask is left alone
  if $b \neq$ *none* **then**
    $W \leftarrow \textsc{Temper}(W, s, b)$; $\mathrm{Mem} \leftarrow \mathrm{Mem} \setminus I(s)$ ▷ re-walk the closure: the same local correction
  end if
end while
return final answer; $W^{\dagger} \leftarrow W$

E Implementation Details

Matched Compute Budget. The compute budget each single agent baseline receives is matched to the workflow it is compared against on every task rather than on average, so no task lets the workflow spend what the baseline could not.

Configuration. We summarize our experiment configurations in Tab. 4, including the core parameters' definitions and configurations, as well as the default values our ablation studies explore against.

Cost Accounting. For any system, its problem-solving cost of a task is the total number of tokens the system consumes, including both input and output over construction, execution, and in-flow optimization.

Accuracy Against Cost. A commonly used approach for improving workflow performance is to best-of-N, optimizing workflows by rerunning workflow construction and execution N times and and keeping the best outcome. However, this optimization gains limited accuracy at the cost of $\sim N\times$ compute cost.

To optimize cheaply, we propose INFLOWOP that optimizes locally in flow without global reruns of any construction or execution stage. To compare the effectiveness of different methods, we study on the single-agent baseline, best-of-N, and INFLOWOP at the matched compute cost by giving the same turn budget. Fig. 9 compares best-of-N ($N = 4$) and INFLOWOP against both the single agent baseline they extend and the single LLM baseline showing the backbone competence, with every cost stated as a multiple of one single-agent run on the same backbone. At matched turn budget, INFLOWOP gains $+7.15\%$, $+7.75\%$, $+10.68\%$, $+11.97\%$, and $+7.38\%$ over the single agent on `Qwen3.5-4B`, `Qwen3.5-9B`, `GPT-5-mini`, `GPT-5.4-mini`, and `GPT-5.6-luna`, respectively. Best-of-N spends $4\times$ those turns and reaches 15.83%, 20.63%, 21.72%, 21.25%, and 43.82%, consistently staying below INFLOWOP on every backbone. This indicates that a gain follows from how a workflow is constructed and optimized rather than from how many times it is attempted.

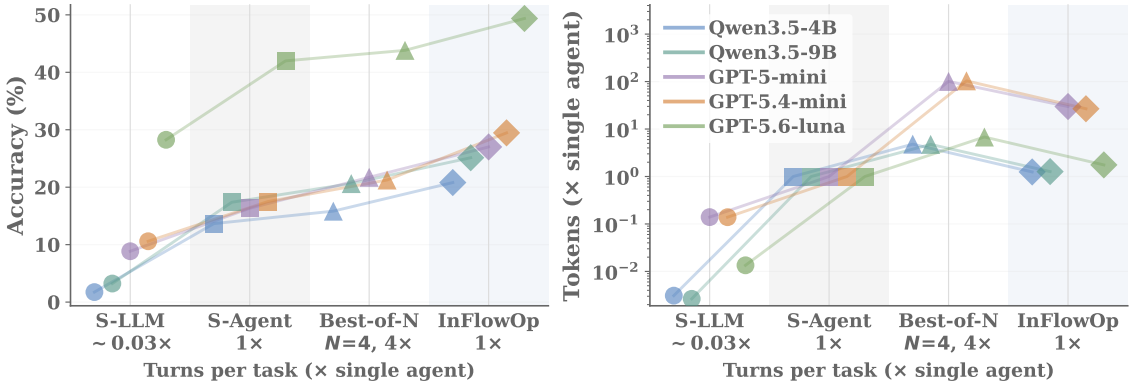

Figure 9: **What Each Method Spends, and What It Earns.** Mean accuracy (%) over the 16 arms of Tab. 1 (*left*), and the tokens a solver spends per task over the same arms (*right*, logarithmic), with one color per backbone. *S-LLM* and *S-Agent* are the single-LLM and single-agent baselines (§5.1). *Best-of-N* repeats workflow construction and execution $N = 4$ times and keeps its best outcome. Both costs are stated as a multiple of one single-agent run on the same backbone. Each method's turn cost is given beneath its name.

Table 3: **Data Statistics.** BRAID as built: 19 evaluation-only arms over 8 domains, with questions each yields at two complexity levels $k = 3$ and $k = 5$. *Original Dataset* shows where a BRAID task is adapted from. For SlideVQA, we construct two variants: (1) *SlideVQA-Doc*: providing slides as a document, and (2) *SlideVQA-Img*: providing slides as individual images. *Task Type* is the answer type a BRAID sample's k questions demand, and *Data Type* is the type of sources provided beside input questions. A level is absent where the dataset yields too few questions ($n_q < 80$) to evaluate on, and *Anchored* is absent where no source of that dataset carries k questions to anchor.

Original Dataset	BRAID	Composition	Task Type	Data Type	$k = 3$	$k = 5$
			Document			
MMLongBench-Doc (Ma et al., 2024)	MMLongBench-Doc	Distributed	Mixed	Document	426	175
MMLongBench-Doc (Ma et al., 2024)	MMLongBench-Doc	Anchored	Mixed	Document	498	305
LongDocURL (Deng et al., 2025)	LongDocURL	Distributed	Mixed	Document	441	340
LongDocURL (Deng et al., 2025)	LongDocURL	Anchored	Mixed	Document	399	325
			Slide			
SlideVQA (Tanaka et al., 2023)	SlideVQA-Doc	Distributed	Open-Ended	Document	1557	1000
SlideVQA (Tanaka et al., 2023)	SlideVQA-Doc	Anchored	Open-Ended	Document	1470	970
SlideVQA (Tanaka et al., 2023)	SlideVQA-Img	Distributed	Open-Ended	Image	1581	1005
SlideVQA (Tanaka et al., 2023)	SlideVQA-Img	Anchored	Open-Ended	Image	1470	1000
			Finance			
DocFinQA (Reddy et al., 2025)	DocFinQA	Distributed	Numeric	Document	84	–
DocFinQA (Reddy et al., 2025)	DocFinQA	Anchored	Numeric	Document	192	–
			Chart			
CharXiv (Wang et al., 2024)	CharXiv	Distributed	Mixed	Image	429	430
ChartMuseum (Tang et al., 2026)	ChartMuseum	Distributed	Mixed	Image	807	805
			Math			
MATH (Hendrycks et al., 2021)	MATH (Level 5)	Distributed	Numeric	Text	603	605
OlympiadBench (He et al., 2024)	OlympiadBench-Math (Text)	Distributed	Numeric	Text	660	660
OlympiadBench (He et al., 2024)	OlympiadBench-Math (Multimodal)	Distributed	Numeric	Image	150	150
			Physics			
OlympiadBench (He et al., 2024)	OlympiadBench-Physics (Text)	Distributed	Numeric	Text	234	200
OlympiadBench (He et al., 2024)	OlympiadBench-Physics (Multimodal)	Distributed	Numeric	Image	456	455
			Science			
HLE (and Phan et al., 2026)	HLE	Distributed	Mixed	Text	1470	785
			Coding			
TACO (Li et al., 2023)	TACO	Distributed	Code Generation	Text	1362	1360

Table 4: **Experiment Configurations.** We summarize our experiment configurations for core parameters (§5), and an ablation arm is studied against the configuration stated here.

Parameter	Definition	Configuration
	Decomposition	
Control-flow primitives	The structures a workflow may use beyond plain sequencing (§D.2)	AND, REPEAT, SELECT, OR
Reliability–latency weight β	What a unit of latency is worth against a unit of reliability in the workflow cost (Eq. 18)	0.25
Creation penalty γ	What creating one agent costs the same objective (Eq. 18)	1.0
Competence bar $\bar{C}$	The cost an agent clears to count as covering an atom (Eq. 1)	0.7
Agent creation	Whether a new agent may be created where no pooled agent covers a subtask	on
Cost matrix	Whether a created agent enters the matrix during COALESCE, or the matrix stays as first scored	dynamic
Expansion budget B	How many groupings COALESCE may expand before it returns its incumbent (§3.1)	20,000
Repetition budget B_{rep}	How many times REPEAT may re-run a subtask in place (Eq. 13)	1
	Estimator	
Estimator	How well an agent meets an atom's demands is scored (§3.1)	rubric
Scored on	The evidence about an agent supplied to the estimator	card and profile
	In-Flow Optimization	
Scope	Where a fault is looked for once execution falls short (§3.2)	local
Unmet bar $\bar{u}$	How far a declared condition falls short before it counts as unmet (Eq. 3)	0.8
Re-assign budget B_{ra}	How many alternative agents a faulty subtask may be given before the ladder climbs (Alg. 4)	1
Re-decompose budget B_{rd}	How many times a faulty subtask may be re-decomposed before it reverts (Alg. 4)	1
Global-probe budget B_{g}	How many suspected subtasks the optional global pass may probe (Alg. 5)	3