Spend Only What You Need: Defect-Aware Residual Coverage for Efficient Multi-Agent Reasoning

Maisha Maliha
School of Computer Science
University of Oklahoma
Norman, OK, USA
maisha.maliha-1@ou.edu

Dean F. Hougen
School of Computer Science
University of Oklahoma
Norman, OK, USA
hougen@ou.edu

Abstract

Multi-agent orchestration over large language models is increasingly used to improve reasoning accuracy, especially on problems that benefit from critique, verification, specialization, or consensus, but these gains often come with inference costs that are not explicitly controlled. Many systems spend similar compute on straightforward queries and genuinely ambiguous or expert-level ones. Existing methods typically follow fixed collaboration protocols, compile task-level workflows offline, route queries among models or collaboration modes, or use learned controllers whose state does not explicitly identify which reasoning defects remain unresolved. As a result, they do not directly decide during inference whether the current multi-agent trajectory still justifies another agent call. We introduce Defect-Aware Residual Coverage (DARC), a training-free inference-time controller for adaptive multi-agent reasoning that represents the current trajectory using four residual defect dimensions: answer uncertainty, claim-level contradiction, verification failure, and aspect under-coverage. DARC selects the next agent by maximizing cost-normalized submodular marginal coverage over the remaining residuals. Each candidate agent receives a role-induced capability profile from frozen embeddings of its natural-language role description, enabling DARC to operate without task-specific supervision, learned routing parameters, or fine-tuning. Across five reasoning and multimodal suites, MMLU-Pro, GPQA-Diamond, LiveBench, MMMU-Pro, and HLE, DARC achieves the strongest accuracy under the main shared heterogeneous six-agent pool while using substantially fewer tokens and agent calls than recent workflow, adaptive orchestration, and cost-aware routing baselines, with additional large-pool experiments showing consistent scaling behavior. It also outperforms the single-call frontier reasoning models tested, showing that residual-aware selective invocation can improve both answer quality and inference efficiency. [1]

1 Introduction

Multi-agent orchestration of large language models increasingly pushes reasoning accuracy beyond what a single model call can achieve. Debate-style collaboration [8], round-table consensus [3], staged collaboration [5], dynamic agent selection [15], and workflow or topology optimization [34, 36] have shown that combining generators, critics, verifiers, and specialists can improve performance on difficult reasoning tasks. These gains, however, come with inference cost that scales with the number of agent calls, intermediate messages, and model invocations. Because frontier-model API prices vary substantially across providers and model scales [4, 16], per-query cost is no longer a

[1] Code: https://github.com/MaishaMaliha1/DARC_Spend-Only-What-You-Need.git

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

secondary implementation detail but a central design constraint. The key question is therefore not whether multi-agent reasoning can help, but when an additional agent call is still worth paying for.

Existing orchestration methods approach this cost–accuracy trade-off from complementary directions, but they do not directly model the unresolved reasoning defects in the current trajectory. Fixed collaboration methods [8, 3, 5] apply the same protocol to every query, so easy questions and genuinely ambiguous questions pay nearly the same multi-agent cost. Workflow and topology optimization methods such as AFlow and GPTSwarm [34, 36] search for effective agentic programs or communication graphs, but the resulting structure is optimized at the task or workflow level rather than recomputed from the defects exposed by a particular trajectory. Query-adaptive orchestration methods such as DAAO and MaAS [20, 31] tailor workflow depth, operator allocation, or agentic architectures to the input query, but their decisions are made before observing the actual claims, contradictions, and coverage gaps produced by invoked agents. Learned dynamic orchestrators such as Puppeteer and OI-MAS [6, 22] can adapt roles or model scales during execution, but they rely on trained controllers or confidence policies rather than an explicit, inspectable decomposition of residual reasoning failures. MAS routing methods such as MasRouter [30] route collaboration mode, roles, and LLM choices for multi-agent systems, while single-model or cascade routers such as RouteLLM and FrugalGPT [16, 4] optimize model selection under cost constraints. These methods improve efficiency, but they do not ask the central diagnostic question: does the current multi-agent state still contain answer uncertainty, claim contradiction, verification failure, or under-covered query aspects that justify another invocation?

We introduce DARC (*Defect-Aware Residual Coverage*), a training-free inference-time controller that decomposes the current trajectory into four residual defect dimensions—answer uncertainty r_U, claim-level contradiction r_C, verification failure r_V, and aspect under-coverage r_R—and selects the next agent by maximizing cost-normalized submodular marginal coverage over the remaining residuals. Each candidate agent receives a capability profile from frozen embeddings of its natural-language role description, so the controller runs without learned routing parameters, supervision, or fine-tuning. Because the residual signal is computed from agent outputs at inference time, DARC adapts not only to the input query but also to the trajectory the system has already produced. We make three contributions. **(1)** We propose a training-free four-dimensional residual decomposition of the multi-agent state that maps reasoning defects directly to agent capabilities, with no supervision or task-specific tuning. **(2)** We introduce a cost-normalized selection rule with a monotone submodular per-step objective: cost normalization shapes efficiency, the diminishing-returns term shapes accuracy, and because residuals update after each call we use the rule as a transparent greedy controller rather than claiming a stationary full-trajectory $(1-1/e)$ guarantee. **(3)** We evaluate DARC on five reasoning suites—MMLU-Pro [24], GPQA-Diamond [19], LiveBench [26], MMMU-Pro [29], and HLE [1]—covering broad knowledge, expert STEM reasoning, contamination-resistant evaluation, and multimodal reasoning. Against fixed, dynamic, search-based, MAS-routing, and cost-routing baselines, including AFlow, DyLAN, DAAO, Puppeteer, MaAS, MasRouter, OI-MAS, RouteLLM, and FrugalGPT, DARC targets the strongest accuracy–cost trade-off by spending additional agent calls only when explicit residual defects remain. We further evaluate a large heterogeneous pool with more than ten agents and more than five distinct model backbones to test whether residual-defect routing scales beyond the small-agent settings commonly used in prior multi-agent evaluations.

2 Motivation

Multi-agent LLM orchestration can close substantial accuracy gaps over single-agent prompting, while recent frontier reasoning models such as Claude Opus 4.7, GPT-5.5, and DeepSeek-V4-Pro close part of this gap with a single high-cost call. Both regimes face the same operational challenge: compute is often allocated by the *method* rather than by the *query state*. Fixed multi-agent systems spend nearly the same budget on easy and hard queries, while single thinking-mode calls expose only coarse reasoning-effort controls. Recent adaptive orchestration and routing methods [20, 31, 6, 30] improve this trade-off by selecting workflows, agentic architectures, roles, or collaboration modes, but they do not explicitly decompose the evolving trajectory into unresolved reasoning defects. As a result, existing methods may adapt to the input query, optimize a task-level workflow or topology, route among models or collaboration modes, or condition on an opaque learned controller state, but they do not provide an explicit, inspectable account of which residual reasoning defect remains: answer uncertainty, claim-level contradiction, verification failure, or under-covered query aspects.

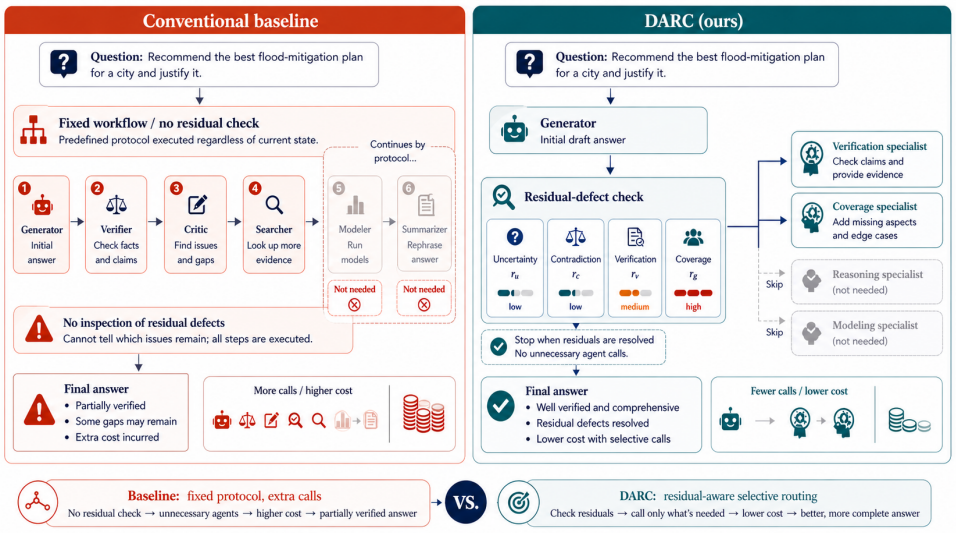

Figure 1: **Motivating example for DARC.** A conventional non-residual-aware baseline may continue invoking agents according to a predefined workflow or opaque routing policy, causing extra calls even when some agents add little new value. It may also stop with remaining defects unresolved, leaving the answer only partially verified or incomplete. DARC instead inspects the current trajectory through four residual defects: uncertainty (r_U), contradiction (r_C), verification failure (r_V), and aspect under-coverage (r_R). It then invokes only the specialists justified by the unresolved residuals and skips unnecessary agents, producing a more complete answer with lower cost.

We instrument four representative adaptive orchestration baselines on 1,000 MMLU-Pro queries using the shared heterogeneous agent pool of §4. For each trajectory, we measure whether the first agent's answer survives unchanged to the final output and how much cost is spent despite that lack of revision (Table 1). This diagnostic does not imply that every unchanged-answer trajectory is wasteful: later agents may still provide useful verification, critique, or robustness. Instead, it measures how much budget is spent on trajectories where the system ultimately does not change its answer, highlighting whether a method can stop once the current state is sufficiently resolved. Figure 1 illustrates the same issue conceptually: a non-residual-aware workflow can continue or terminate based on its protocol rather than on which reasoning defects remain, whereas DARC uses the residual state to decide whether to continue, which specialist to call, and when to stop. Across adaptive baselines, a large fraction of cost is spent on unchanged-answer trajectories. DARC's residual-defect signal reduces this wasted-cost share by invoking additional agents only when explicit unresolved defects remain.

Table 1: Motivating diagnostic on 1,000 MMLU-Pro queries. *Unchanged*: first agent's answer matches the final output. *Cost*: dollar cost per query, including controller overhead. *Wasted*: cost share spent on unchanged-answer trajectories beyond the first call. †: paired bootstrap vs. DARC (10K resamples, $p < 0.001$).

Metric	**DAAO** [20]	**MaAS** [31]	**Puppeteer** [6]	**MasRouter** [30]	**DARC**
Unchanged	58%	55%	59%	54%	60%
Cost ($/query)	0.044†	0.041†	0.046†	0.025†	**0.011**
Wasted	51%†	44%†	49%†	39%†	**8%**

The source of the gap is not simply whether a method is adaptive, but what signal drives adaptation. A difficulty-aware controller can choose deeper or shallower workflows before execution begins [20]; a sampled supernet can instantiate a query-dependent agentic architecture [31]; a learned cascaded router can select collaboration modes, roles, or model assignments [30]; and an RL-trained orchestrator can condition decisions on evolving state [6]. These mechanisms improve over fixed protocols, but their routing signals do not provide an explicit, inspectable account of which reasoning defect remains unresolved in the current trajectory. DARC fills this gap with a residual-defect field

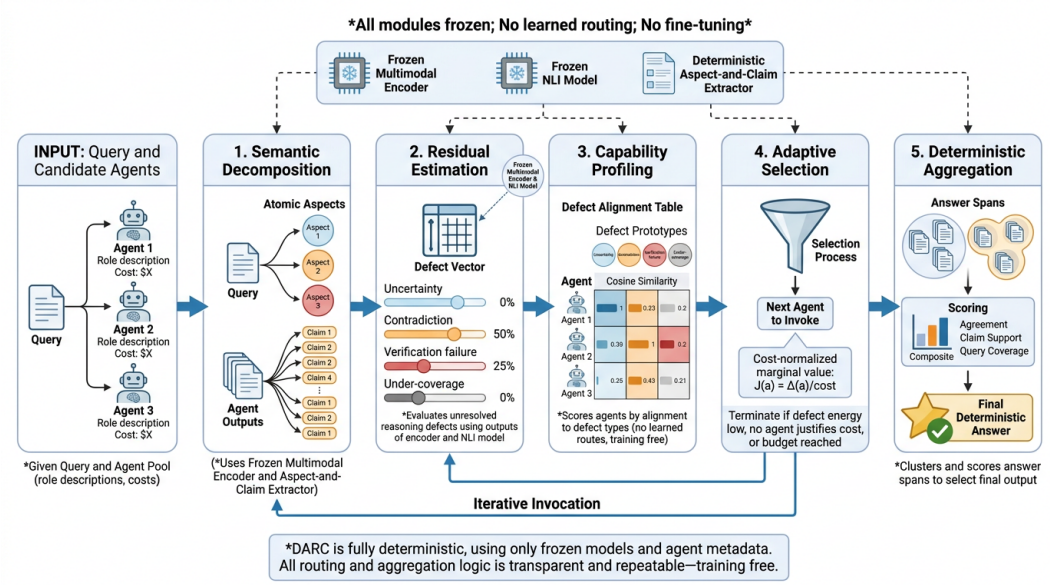

Figure 2: Overview of the proposed DARC framework. Given a query and a pool of candidate agents with role descriptions and invocation costs, DARC first decomposes the query and agent outputs into atomic aspects and claims. It then estimates four residual defect dimensions—answer uncertainty, claim-level contradiction, verification failure, and aspect under-coverage—using frozen modules. Candidate agents are scored by role-induced capability profiles, and the controller selects the next agent through a cost-normalized marginal-coverage rule. The process iterates until residual energy is low, no remaining agent justifies its cost, or the budget is reached, after which deterministic aggregation selects the final answer.

over four dimensions: answer uncertainty, claim-level contradiction, verification failure, and aspect under-coverage. The next agent is invoked only when at least one residual defect is severe enough to justify its cost; otherwise the trajectory terminates. The bottom row of Table 1 previews the consequence: DARC maintains a comparable unchanged-answer rate while reducing wasted-cost share to single digits and lowering per-query dollar cost, with full benchmark results in §4.

3 Methodology

We introduce DARC, which is a deterministic inference-time controller: given the same agent outputs, costs, role descriptions, frozen modules, and thresholds, it makes the same invocation and stopping decisions. It decides, at each step, which agent to invoke next based on which reasoning defects remain unresolved in the current trajectory. Its thresholds are calibrated on held-out development data. Each decision is computed from a frozen multimodal encoder, a frozen NLI model, a deterministic aspect-and-claim extractor, agent role descriptions, and explicit invocation costs. The controller proceeds through five stages: **(i)** *semantic decomposition* (§3.1) splits the query into atomic aspects and each output into claims; **(ii)** *residual estimation* (§3.2) summarizes the state as a four-dimensional defect vector over uncertainty, contradiction, verification failure, and under-coverage; **(iii)** *capability profiling* (§3.3) scores each candidate agent against the four dimensions via cosine similarity between its role description and fixed prototype strings; **(iv)** *adaptive selection* (§3.4) invokes the agent maximizing a cost-normalized submodular marginal-value criterion, terminating once residual energy is small, no agent justifies its cost, or a budget is reached; and **(v)** *deterministic aggregation* (§3.5) clusters final-answer spans and scores them by agreement, claim support, and query coverage. By "training-free" we mean no learned routing parameters and no fine-tuning; the aspect-and-claim extractor itself is a single zero-shot prompt on a frozen LLM, with details in §4. Figure 2 illustrates the proposed DARC pipeline, while Algorithm 1 summarizes its complete inference-time procedure.

3.1 Setup and Semantic Units

We begin by fixing the notation and the atomic units that the rest of the controller operates over. A fixed pool $\mathcal{A} = \{a_1, \ldots, a_N\}$ is given, with each agent a specified by a role description $r(a)$ and a cost $c(a) > 0$. At step t the controller holds outputs $\mathcal{O}_t = \{o_1, \ldots, o_m\}$ from invoked agents $\mathcal{S}_t \subseteq \mathcal{A}$, with remaining candidates $\mathcal{R}_t = \mathcal{A} \setminus \mathcal{S}_t$. Whole-answer reasoning loses too much structure to detect partial failures, so we decompose both sides of the trajectory into atomic units. The query is split into aspects $\mathcal{P}(q) = \{p_1, \ldots, p_L\}$ representing individual sub-requirements, and each output is split into claims $\mathcal{C}(o_i) = \{z_{i1}, \ldots, z_{iM_i}\}$ representing individual factual statements. Query aspects remain fixed throughout the trajectory. After each invocation, the controller extracts claims from the new output and recomputes the residuals.

Let f be a frozen multimodal encoder mapping text, images, and documents into a shared semantic space, and let h be a frozen NLI model operating on text claim pairs. We write $\mathrm{sim}(x, y) = \max(0, \cos(f(x), f(y)))$ for cosine similarity in the encoder space, and $\mathrm{entail}(\cdot, \cdot)$ and $\mathrm{contr}(\cdot, \cdot)$ for the entailment and contradiction probabilities returned by h. The two modules play distinct roles. The encoder f measures semantic proximity for clustering, coverage, and role-defect alignment, whereas the NLI model h estimates claim-level entailment and contradiction. We never use embeddings as logical verifiers and never use NLI as a similarity metric. Routing f through a unified multimodal space lets the same machinery operate on text-only and vision-language queries without per-modality logic. All four residuals are computed over the atomic units rather than over raw outputs.

3.2 Residual Defect Field

Given the aspects and claims defined above, we now turn each output trajectory into a compact diagnostic of what reasoning work still remains. The state s_t is summarized by $r_t = (r_{t,U}, r_{t,C}, r_{t,V}, r_{t,R}) \in [0, 1]^4$ measuring uncertainty, claim-level contradiction, verification failure, and aspect under-coverage. The scalar residual energy is

$$E(s_t) = \sum_{j \in \{U,C,V,R\}} \alpha_j \, r_{t,j}, \tag{1}$$

with fixed nonnegative weights α_j. The four components are kept separate because each admits a distinct remedy. Uncertainty is reduced by adjudicators, contradiction by critics, verification failure by checkers, and under-coverage by generators, so collapsing to a scalar would obscure which remedy the current state demands.

Uncertainty. Disagreement among agents is the most common signal that more work is needed. Final-answer spans $\{y_i\}_{i=1}^m$ are clustered by sim at threshold ρ, giving clusters $G_1, \ldots, G_K$ with masses $p_k = |G_k|/m$. Then

$$r_{t,U} = \begin{cases} -\frac{1}{\log K} \sum_{k=1}^{K} p_k \log p_k, & K > 1, \\ 0, & K \leq 1, \end{cases} \tag{2}$$

where $\log K$ rescales Shannon entropy to $[0, 1]$ so $r_{t,U}$ is comparable across queries with different answer-space cardinalities.

Contradiction. Outputs may agree on the final answer while relying on incompatible intermediate claims, so we score contradiction at the claim level to surface this hidden disagreement. Define $\kappa(o_i, o_j) = \max_{z \in \mathcal{C}(o_i),\, z' \in \mathcal{C}(o_j)} \mathrm{contr}(z, z')$ as the worst-pair contradiction between two outputs, so a sharp single conflict is not diluted by averaging across compatible claim pairs. Then

$$r_{t,C} = \begin{cases} \frac{2}{m(m-1)} \sum_{i<j} \kappa(o_i, o_j), & m \geq 2, \\ 0, & m < 2. \end{cases} \tag{3}$$

Verification failure. Internal agreement does not imply correctness, since agents can converge on a wrong answer. Let $\hat{y}_t = \mathrm{Agg}(\mathcal{O}_t)$ be the provisional answer drawn from source output $o_{s(t)}$, with claims $\hat{z}_1, \ldots, \hat{z}_M$. To prevent a claim from verifying itself, cross-output evidence excludes the source output, giving $\mathrm{evid}_t(\hat{z}) = \max_{o_i \neq o_{s(t)}} \max_{z \in \mathcal{C}(o_i)} \mathrm{entail}(z, \hat{z})$ for $m \geq 2$, and falling back to $\max_{z \in \mathcal{C}(o_{s(t)})} \mathrm{entail}(z, \hat{z})$ at $m = 1$ as a same-output initialization heuristic. To avoid spurious NLI signal against unrelated topics, we gate contradiction by relevance via

$\mathcal{L}_t(\hat{z}) = \{\ell \mid \mathrm{sim}(p_\ell, \hat{z}) \geq \tau_{\mathrm{rel}}\}$, with the max over an empty set defined as 0. The support score combines this evidence with the contradiction scores of relevant aspects,

$$\mathrm{supp}_t(\hat{z}) = \mathrm{evid}_t(\hat{z}) \Big(1 - \max_{\ell \in \mathcal{L}_t(\hat{z})} \mathrm{contr}(p_\ell, \hat{z})\Big). \tag{4}$$

The verification residual averages the unsupported share, giving $r_{t,V} = 1 - \frac{1}{M} \sum_u \mathrm{supp}_t(\hat{z}_u)$ when $m, M \geq 1$, and 0 otherwise. Source-output exclusion, relevance gating, and the bounded-max form are each ablated in Appendix A2.4, and only the full design produces a stable query-length-invariant residual.

Under-coverage. A fluent answer addressing three of five sub-requirements can score highly under whole-answer similarity, so coverage is measured per aspect. Topical overlap can increase the estimated coverage even when an output only partially addresses a requirement. Define $\gamma_t(p_\ell) = \max_{o_i} \max\{\mathrm{sim}(p_\ell, o_i), \max_{z \in \mathcal{C}(o_i)} \mathrm{entail}(z, p_\ell)\}$ as the best alignment of any output or claim to aspect p_ℓ. Then $r_{t,R} = 1 - \frac{1}{L} \sum_\ell \gamma_t(p_\ell)$ for $m \geq 1$, and $r_{t,R} = 1$ at cold start so no aspect is initially covered.

3.3 Role-Induced Capability Profiles

The residual vector tells us which defects need attention, but not which agent is best placed to address each one, so we now build a defect-to-agent alignment score directly from agent role descriptions. To choose the next agent we estimate how well each agent addresses each defect from its role description rather than from learned weights, which keeps the controller training-free while still differentiating agents along the four defect dimensions. Four fixed prototype strings encode the dimensions. The uncertainty prototype d_U reads "resolve disagreement and choose among competing answers," the contradiction prototype d_C reads "detect contradictions and critique inconsistent reasoning," the verification prototype d_V reads "verify claims using evidence, tests, and formal checks," and the coverage prototype d_R reads "cover missing query requirements and complete the answer." With $g = f$,

$$k_{a,j} = \max\big(0, \cos(g(r(a)), g(d_j))\big) \in [0, 1]. \tag{5}$$

Capabilities are deliberately not normalized across j, since row-normalization would erase the distinction between strong generalists and narrowly aligned weak agents, and both the specialization pattern and the absolute magnitude matter for the marginal-value computation. These scores estimate alignment with the declared role. Their usefulness depends on how well the description reflects the agent's actual capabilities. Defect-prototype phrasing sensitivity (Spearman $\rho_s \geq 0.9$ under paraphrase) and downstream-accuracy effects are reported in Appendix A7.2.

3.4 Coverage Objective and Adaptive Invocation

With the residual vector identifying severe defects and the capability profiles indicating which agents address them, we combine the two into a single cost-aware selection rule. The value of an agent set $S \subseteq \mathcal{A}$ at fixed state s_t is

$$F_t(S) = \sum_j \alpha_j \, r_{t,j} \Big(1 - \prod_{a \in S} (1 - k_{a,j})\Big), \tag{6}$$

a probabilistic-coverage form in which the bracketed factor rises toward 1 as defect-aligned agents accumulate (with diminishing returns), weighted by current defect severity $\alpha_j r_{t,j}$ so the controller pays only for coverage of currently severe defects. For fixed s_t, F_t is normalized, monotone, and submodular (Appendix A2.7). For a remaining agent $a \in \mathcal{R}_t$, we evaluate the marginal at the already-invoked set, holding r_t fixed: $\Delta_t(a) = F_t(\mathcal{S}_t \cup \{a\}) - F_t(\mathcal{S}_t)$. Expanding this difference gives

$$\Delta_t(a) = \sum_j \alpha_j \, r_{t,j} \, k_{a,j} \prod_{b \in \mathcal{S}_t} (1 - k_{b,j}), \qquad J_t(a) = \Delta_t(a)/c(a), \tag{7}$$

which is large only when defect j is severe, the agent is aligned with j, and prior invocations have not already saturated j. Because r_t updates after every invocation, F_t is re-evaluated each step rather than maximized over a stationary objective, so we do not claim the $(1 - 1/e)$ greedy guarantee, and we instead bound the per-step gap to the brute-force optimum on small pools empirically in Appendix A2.9.

The controller terminates when $\mathcal{O}_t \neq \varnothing$ and at least one of $E(s_t) \leq \varepsilon$, $\max_a J_t(a) \leq \lambda$, or $|\mathcal{S}_t| = B$ holds, and also when $\mathcal{R}_t = \varnothing$. The three conditions encode complementary signals, with ε checking that defect energy is low, λ checking that no remaining agent is worth its cost, and B capping the number of agent invocations even when new outputs introduce new clusters or contradictions. The ε threshold is well-calibrated to empirical error rate (slope ≈ 1.06, Appendix A6.14). When none of the conditions fire, the controller invokes $a_t^\star = \arg\max_{a \in \mathcal{R}_t} J_t(a)$, breaking ties by lower cost and then by canonical order. At cold start $r_0 = (0, 0, 0, 1)$, so the rule reduces to $\arg\max_a \alpha_R k_{a,R}/c(a)$ and naturally selects a generator first. Termination guarantees and the (ε, λ) trade-off are derived in Appendix A2.8.

3.5 Answer Aggregation

Once the selection rule has terminated, we still need to commit to a single final answer from the collected outputs, and we do so using the same residual signals that drove routing. $\hat{y}_t = \mathrm{Agg}(\mathcal{O}_t)$ is formed deterministically, so the controller cannot prefer one cluster for routing while returning another. Spans are clustered at threshold ρ as in equation (2), and each cluster G_k receives a composite score

$$S(G_k) = \beta_U |G_k|/m + \beta_V \, \overline{\mathrm{supp}}_t(G_k) + \beta_R \, \mathrm{Cov}_t(G_k),$$

where $\overline{\mathrm{supp}}_t(G_k)$ averages equation (4) over cluster claims and $\mathrm{Cov}_t(G_k) = \frac{1}{L}\sum_\ell \max_{o_i:\, y_i \in G_k} \max\{\mathrm{sim}(p_\ell, o_i), \max_{z \in \mathcal{C}(o_i)} \mathrm{entail}(z, p_\ell)\}$ is cluster-level coverage, with all three terms in $[0, 1]$. The winner $G^\star = \arg\max_k S(G_k)$ returns its highest-support member, breaking ties first by mean support and then by recency. The same rule operates at intermediate steps and at termination.

4 Experiments

Experimental setup. We evaluate on five reasoning suites spanning broad text reasoning, expert STEM reasoning, contamination-resistant evaluation, and multimodal reasoning: **MMLU-Pro** [24], **GPQA-Diamond** [19], **LiveBench** [26], **MMMU-Pro** [29], and **HLE** [1]. These suites test whether adaptive multi-agent orchestration improves accuracy–cost trade-offs across language-only reasoning, expert questions, temporally refreshed evaluation, and image-grounded reasoning. We evaluate four agent-pool settings. **S1** is a lightweight four-agent heterogeneous pool with three model families. **S2** is the default six-agent heterogeneous pool used for the main comparison; it contains Claude Opus 4.7 [2] as generator, GPT-5.5 [17] in thinking mode as reasoning specialist, Qwen3-72B-Instruct [28] as refiner, Qwen3-32B-Instruct [28] as critic, DeepSeek-R1-Distill-Qwen-32B [9] as verifier, and Qwen3-Coder-Next 80B [18] as code specialist. **S3** is a larger 12-agent heterogeneous pool with more than five distinct model backbones, used to test scalability. **S4** is a homogeneous control in which all roles share one backbone but retain distinct role prompts. Unless otherwise stated, results use S2; S1 and S4 are diagnostic ablations reported in Appendix A10. Dataset details, pool composition, multimodal handling, costs, and hyperparameters are in Appendices A3 and A4.

We compare against fixed, dynamic, search-based, multi-agent-routing, and cost-routing baselines: AFlow [34], DyLAN [15], DAAO [20], MaAS [31], Puppeteer [6], MasRouter [30], OI-MAS [22], RouteLLM [16], and FrugalGPT [4]. We also include CoT [25] and Self-Consistency [23] as single-agent references, and Claude Opus 4.7, GPT-5.5, and DeepSeek-V4-Pro [7] as single-call frontier comparators. All runnable multi-agent baselines use the same pool, multimodal protocol, token accounting, and 300-query development split as DARC; configurations and fairness checks are in Appendix A5. We report accuracy or official score, mean USD cost per query, tokens per query, and wall-clock latency. Significance is assessed against DARC using paired bootstrap tests with 10K resamples and Bonferroni correction within each suite; all results are averaged over five seeds unless otherwise noted. Additional efficiency and variance details are in Appendix A9. Appendix A12 consolidates the diagnostic validation, robustness analyses, and clarifications of the budget-control and evaluation protocols.

4.1 Main Results

Table 2 gives the main S2 comparison across all five benchmarks. It includes single-agent references, single-call frontier comparators, and all runnable orchestration/routing baselines. We use a separate

Table 2: Main results under **S2**, the default six-agent heterogeneous pool. We report accuracy or official score (Acc/Score, %) and tokens per query (Tok, $\times 10^3$). Single-agent and single-call frontier models are reference comparators; all runnable multi-agent baselines use the same S2 pool, multimodal protocol, and accounting procedure. Best per column is **bold**, second-best is underlined. Mean over 3 independent runs; STD ≤ 0.6.

	MMLU-Pro		GPQA-D		LiveBench		MMMU-Pro		HLE	
Method	Acc	Tok	Acc	Tok	Score	Tok	Acc	Tok	Score	Tok
Single-agent references										
CoT [25]	65.4	0.54	41.2	0.66	50.1	0.78	51.5	0.92	18.6	1.05
Self-Consistency [23]	68.0	2.71	44.5	3.30	52.8	3.91	54.8	4.25	21.4	4.80
Single-call frontier comparators										
Claude Opus 4.7 [2]	74.1	4.85	54.0	5.30	63.4	5.70	72.6	6.10	38.5	6.40
GPT-5.5 [17]	75.4	5.35	55.8	6.10	65.2	6.45	74.1	6.75	41.2	7.20
DeepSeek-V4-Pro [7]	69.8	3.05	52.4	3.55	59.7	3.85	62.3	4.10	31.6	4.55
Workflow / dynamic multi-agent										
AFlow [34]	72.3	4.20	52.1	4.75	60.1	5.30	61.8	5.45	32.4	5.90
DyLAN [15]	71.6	2.35	50.5	2.95	59.0	3.25	60.2	3.45	30.8	3.85
Adaptive orchestration / routing										
DAAO [20]	73.5	2.05	53.2	2.55	62.0	2.90	66.8	3.20	35.6	3.55
MaAS [31]	74.0	2.25	53.6	2.75	62.7	3.05	67.4	3.35	36.2	3.70
Puppeteer [6]	74.3	2.40	54.1	2.95	63.1	3.20	68.0	3.55	37.0	3.95
MasRouter [30]	72.8	1.35	51.7	1.70	60.8	2.05	64.9	2.35	33.8	2.60
OI-MAS [22]	73.2	1.55	52.5	1.95	61.4	2.25	65.5	2.55	34.7	2.85
RouteLLM [16]	68.5	0.85	45.8	1.05	54.6	1.25	55.2	1.55	24.3	1.75
FrugalGPT [4]	67.9	0.78	44.9	0.98	53.8	1.18	54.1	1.48	23.6	1.65
DARC (ours)	**77.8**	**0.62**	**58.7**	**0.78**	**70.8**	**1.05**	**76.2**	**1.15**	**43.5**	**1.35**

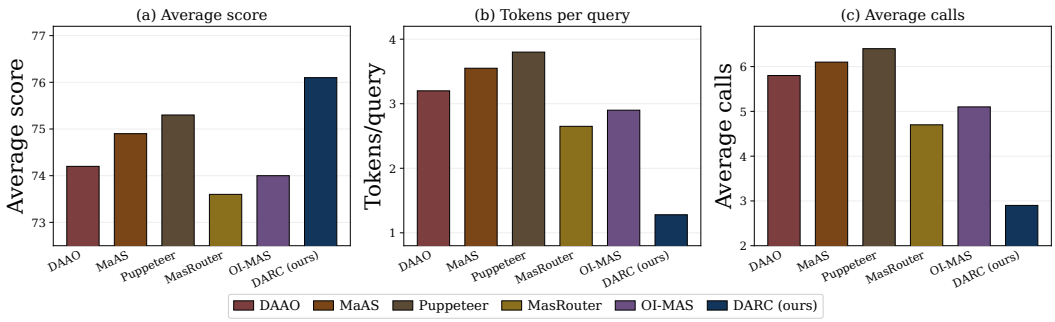

Figure 3: **Large-pool comparison under S3.** S3 uses 12 agents and more than five distinct model backbones. We compare DARC with the closest adaptive orchestration and routing baselines using three summary metrics: average score, tokens per query, and average calls.

S3 comparison in figure 3 to test whether the strongest adaptive methods scale when the candidate pool expands to 12 agents. Under S2, DARC achieves the strongest accuracy–cost trade-off across the five suites. The reference rows test whether single-agent prompting or a single frontier-model call can replace multi-agent reasoning, while the remaining rows compare against workflow, dynamic-agent, adaptive-orchestration, and cost-routing baselines. Unlike methods that commit to a workflow, model choice, or agent subset before observing the full trajectory, DARC recomputes residual defects after each output and invokes additional agents only when their marginal coverage justifies the cost. Detailed dollar-cost, latency, significance, and per-seed results are reported in Appendix A9.

Large-pool scalability. Figure 3 tests whether the main trend persists when the candidate pool expands from S2 to S3. We compare against the closest adaptive baselines, since these methods are designed to select workflows, roles, or model assignments from larger candidate sets. DARC benefits from the expanded pool without invoking most available agents, maintaining lower token use and fewer calls while preserving the strongest average score. S1 and S4 results, together with residual breakdowns, LiveBench refresh results, failure analysis, and limitations, are reported in Appendices A10, A6.7, A6.3, A6.16, and A6.17.

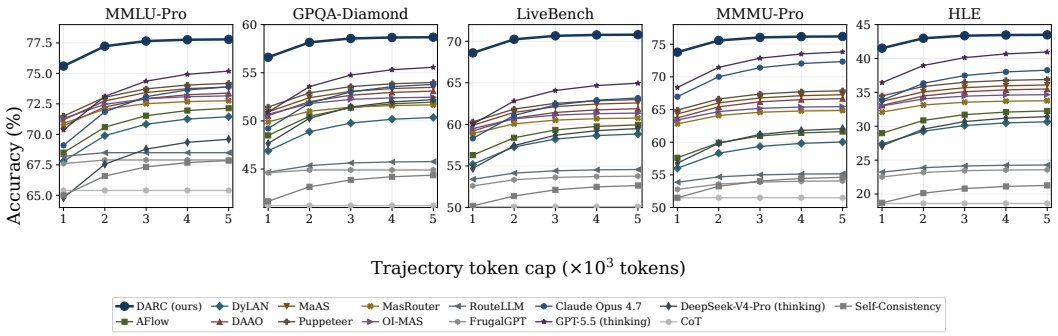

Figure 4: **Accuracy–cost Pareto frontier across five reasoning suites.** Each marker shows accuracy or official score under a matched token cap. DARC maintains high accuracy at low budgets because early calls target the largest residual defects, producing steep early gains and saturation as defects are resolved.

Accuracy–Cost Pareto Frontier: The main tables compare methods at their natural operating points; we next test whether the accuracy–cost ordering persists under matched token budgets. We sweep a hard trajectory token cap $T_{\text{cap}} \in \{1, 2, 3, 4, 5\}$K and re-run each method using its natural budget-control mechanism: budget B for DARC, round or stage count for fixed multi-agent methods, workflow depth for search-based baselines, and routing or cascade thresholds for cost-aware routers. Figure 4 shows that DARC remains on the Pareto frontier across the swept range. Its curve is high even under tight budgets because the first invocations are selected to maximize cost-normalized coverage of the largest remaining residual defects. Later calls therefore provide diminishing returns, unlike fixed or workflow-based protocols that often spend additional tokens before checking whether unresolved defects remain. Per-suite curves, sweep ranges, and budget-control details are reported in Appendix A6.2.

Implementation Details: All methods are evaluated under the same benchmark splits, multimodal protocol, token accounting, and development split described above. Open-weight backbones are served through `vLLM` with greedy decoding for non-thinking agents, while thinking or reasoning specialists use provider-recommended settings. API-served models, including Claude Opus 4.7, GPT-5.5, and DeepSeek-V4-Pro, are accessed through dated snapshots with reasoning settings and provider prices recorded in Appendix A6.1. The frozen multimodal encoder f is Gemini Embedding 2, and the frozen NLI model h is DeBERTa-v3-large-MNLI [11]. The aspect-and-claim extractor is a single zero-shot prompt with a fixed JSON schema. Hyperparameters are fixed once on the 300-query development split and held constant across all suites and pool settings: ρ=0.78, τ_{rel}=0.55, α=$(0.30, 0.25, 0.25, 0.20)$, ε=0.18, λ=0.05, B=5, and $\beta_{U,V,R}$=$(0.4, 0.4, 0.2)$. Compute, cost accounting, sensitivity, prompts, extractor schema, API snapshots, and per-method implementation details are reported in Appendices A6.9, A8, A6.8, A7.7, and A6.1.

4.2 Ablation and Sensitivity

We ablate the core design choices behind DARC on MMLU-Pro: the four-defect residual decomposition, the cost-normalized submodular selection rule, agent-pool heterogeneity, and sensitivity to the trajectory budget B. Figure 5 summarizes the headline results, with per-suite ablations and additional sensitivity studies in Appendix A7. Residuals are non-redundant: removing any single residual decreases accuracy, with the largest MMLU-Pro drop caused by removing uncertainty r_U (-5.0 points), and collapsing the four residuals into a scalar signal performs worse than removing most individual residuals, showing that the controller must preserve defect type when deciding which agent to invoke. The selection rule also matters: removing cost normalization preserves accuracy but increases token use, indicating that $/c(a)$ primarily controls efficiency, while removing the submodular diminishing-returns term lowers accuracy by repeatedly selecting agents aligned with already-covered defects; random selection causes the largest drop. Heterogeneous specialization drives the gain: accuracy improves as the heterogeneous pool grows from $|\mathcal{A}| = 2$ to $|\mathcal{A}| = 6$, but a homogeneous six-agent pool remains substantially worse, showing that role/model diversity matters more than pool size alone. Finally, performance saturates with a moderate budget: increasing B from

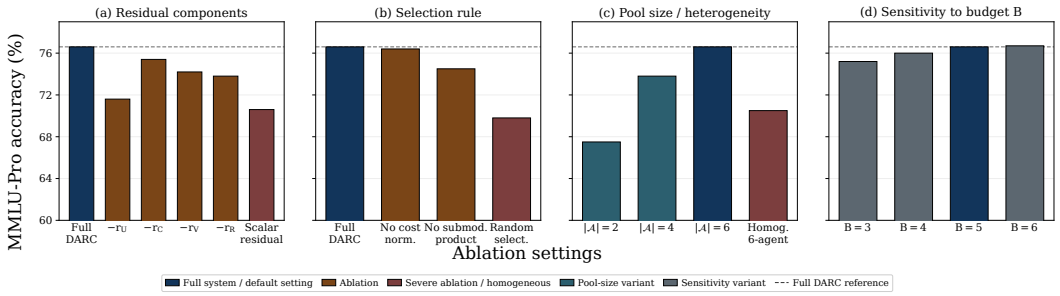

Figure 5: **Headline ablations on MMLU-Pro.** (a) Residual components, (b) selection rule, (c) agent-pool size and heterogeneity, and (d) sensitivity to budget B. The dashed line marks the full DARC reference. Navy denotes the full system/default setting, brown standard ablations, mahogany severe ablations or homogeneous control, teal pool-size variants, and slate budget-sensitivity variants.

3 to 5 improves accuracy, while $B = 6$ gives little additional gain, consistent with the Pareto analysis that DARC resolves the largest residual defects early and later calls provide diminishing returns.

5 Related Work

Multi-agent LLM systems improve reasoning via debate, consensus, dynamic collaboration, workflow search, and topology optimization [8, 3, 5, 15, 36, 34, 31]; adaptive orchestrators condition routing or workflow choices on query or trajectory state [20, 22, 6]; and cost-aware routers and cascades reduce inference cost by selecting cheaper models, roles, or schedules [4, 16, 30]. Unlike these approaches, DARC explicitly recomputes residual reasoning defects after each agent output and invokes another agent only when the current trajectory still contains unresolved uncertainty, contradiction, verification failure, or under-coverage; extended related work is in Appendix A1.

6 Conclusion

We introduced DARC, a training-free inference-time controller that makes multi-agent reasoning more efficient by recomputing explicit residual defects after each agent output and invoking another specialist only when uncertainty, contradiction, verification failure, or under-coverage remains unresolved. Across five reasoning and multimodal suites, DARC improves the accuracy–cost trade-off over fixed workflows, adaptive orchestration methods, multi-agent routers, and cost-aware routing baselines while using fewer tokens and fewer agent calls. Limitations are discussed in Appendix A6.17.

7 Acknowledgments

The authors received no specific funding for this work. The authors declare no competing financial interests or other relationships that could be perceived as influencing the work.

References

[1] 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. *Nature*, 649(8099):1139–1146, January 2026. ISSN 1476-4687. doi: 10.1038/s41586-025-09962-4. URL http://dx.doi.org/10.1038/s41586-025-09962-4.

[2] Anthropic. Introducing Claude Opus 4.7. https://www.anthropic.com/news/claude-opus-4-7, April 2026. Model card: claude-opus-4-7.

[3] Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. ReConcile: Round-table conference improves reasoning via consensus among diverse LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7066–7085, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.381. URL https://aclanthology.org/2024.acl-long.381/.

[4] Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research (TMLR)*, 2024.

[5] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=EHg5GDnyq1.

[6] 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://openreview.net/forum?id=L0xZPXT3le.

[7] DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro, 2026. Tech report: https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro/blob/main/DeepSeek_V4.pdf.

[8] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 11733–11763. PMLR, 2024. URL https://proceedings.mlr.press/v235/du24e.html.

[9] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *Nature*, 645(8081):633–638, 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL `http://dx.doi.org/10.1038/s41586-025-09422-z`.

[10] Xiyu Guo, Shan Wang, Chunfang Ji, Xuefeng Zhao, Wenhao Xi, Yaoyao Liu, Qinglan Li, Chao Deng, and Junlan Feng. Towards generalized routing: Model and agent orchestration for adaptive and efficient inference, 2025. URL `https://arxiv.org/abs/2509.07571`.

[11] Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL `https://openreview.net/forum?id=sE7-XhLxHA`.

[12] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.825. URL `https://aclanthology.org/2023.emnlp-main.825/`.

[13] Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. An LLM compiler for parallel function calling. In *Forty-first International Conference on Machine Learning*, 2024. URL `https://openreview.net/forum?id=uQ2FUoFjnF`.

[14] Yubin Kim, Chanwoo Park, Hyewon Jeong, Yik Siu Chan, Xuhai Xu, Daniel McDuff, Hyeonhoon Lee, Marzyeh Ghassemi, Cynthia Breazeal, and Hae Won Park. Mdagents: An adaptive collaboration of llms for medical decision-making. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 79410–79452. Curran Associates, Inc., 2024. doi: 10.52202/079017-2522. URL `https://proceedings.neurips.cc/paper_files/paper/2024/file/90d1fc07f46e31387978b88e7e057a31-Paper-Conference.pdf`.

[15] Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic LLM-powered agent network for task-oriented agent collaboration. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=XII0Wp1XA9.

[16] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8sSqNntaMr.

[17] OpenAI. Introducing GPT-5.5. https://openai.com/index/introducing-gpt-5-5/, April 2026. Released April 23, 2026. System Card: https://deploymentsafety.openai.com/gpt-5-5.

[18] Qwen Team. Qwen3-Coder-30B-A3B-Instruct. https://huggingface.co/Qwen/Qwen3-Coder-30B-A3B-Instruct, 2025. Hugging Face model card; released July 31, 2025; Apache-2.0.

[19] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof Q&A benchmark. In *First Conference on Language Modeling (COLM)*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

[20] Jinwei Su, Qizhen Lan, Yinghui Xia, Lifan Sun, Weiyou Tian, Tianyu Shi, Lewei He, and Jingsong Yang. Difficulty-aware agentic orchestration for query-specific multi-agent workflows. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*, Dubai, United Arab Emirates, 2026. Association for Computing Machinery. doi: 10.1145/3774904.3792240. URL https://doi.org/10.1145/3774904.3792240.

[21] Jingbo Wang, Sendong Zhao, Haochun Wang, Yuzheng Fan, Lizhe Zhang, Yan Liu, and Ting Liu. Optimal-agent-selection: State-aware routing framework for efficient multi-agent collaboration, 2025. URL https://arxiv.org/abs/2511.02200.

[22] Jingbo Wang, Sendong Zhao, Jiatong Liu, Haochun Wang, Wanting Li, Bing Qin, and Ting Liu. Orchestrating intelligence: Confidence-aware routing for efficient multi-agent collaboration across multi-scale models, 2026. URL https://arxiv.org/abs/2601.04861.

[23] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

[24] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems 37 (NeurIPS) Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=y10DM6R2r3. Spotlight.

[25] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

[26] Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha Venkat Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. Livebench: A challenging, contamination-limited LLM benchmark. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=sKYHBTAxVa.

[27] Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-Pack: Packed resources for general Chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 641–649. ACM, 2024. doi: 10.1145/3626772.3657878. URL https://doi.org/10.1145/3626772.3657878.

[28] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

[29] Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Botao Yu, Ge Zhang, Huan Sun, Yu Su, Wenhu Chen, and Graham Neubig. MMMU-pro: A more robust multi-discipline multimodal understanding benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15134–15186, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.736. URL https://aclanthology.org/2025.acl-long.736/.

[30] Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. MasRouter: Learning to route LLMs for multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*, pages 15549–15572, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.757. URL https://aclanthology.org/2025.acl-long.757/.

[31] Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research. PMLR, 2025. URL https://openreview.net/forum?id=imcyVlzpXh. Oral presentation.

[32] Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for LLM-based multi-agent systems. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=LkzuPorQ5L.

[33] Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=LpE54NUnmO.

[34] Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui 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. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=z5uVAKwmjf.

[35] Zheyuan Zhang, Kaiwen Shi, Zhengqing Yuan, Zehong Wang, Tianyi Ma, Keerthiram Murugesan, Vincent Galassi, Chuxu Zhang, and Yanfang Ye. Agentrouter: A knowledge-graph-guided llm router for collaborative multi-agent question answering, 2025. URL https://arxiv.org/abs/2510.05445.

[36] Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 62743–62767. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhuge24a.html. Oral presentation.

Appendix
Contents

1 Introduction **1**

2 Motivation **2**

3 Methodology **4**

3.1 Setup and Semantic Units 5

3.2 Residual Defect Field 5

3.3 Role-Induced Capability Profiles 6

3.4 Coverage Objective and Adaptive Invocation 6

3.5 Answer Aggregation 7

4 Experiments **7**

4.1 Main Results 7

4.2 Ablation and Sensitivity 9

5 Related Work **10**

6 Conclusion **10**

7 Acknowledgments **11**

Appendix **18**

A1 Extended Related Work **21**

A2 Methodology Details **21**

A2.1 Rationale for the Energy Decomposition 21

A2.2 Properties of the Uncertainty Term 22

A2.3 Contradiction at the Claim Level 23

A2.4 Design Choices in Verification 23

A2.5 Per-Aspect Coverage 23

A2.6 Capability Normalization 24

A2.7 Submodularity of F_t 24

A2.8 Termination and the Role of the Budget 25

A2.9 Empirical Optimality Bound on Small Pools 25

A3 Suite Splits and Scoring Conventions **26**

A4 Agent Pool Detail and Cost Calibration **28**

A5 Baseline Configurations and API Parameters **29**

A6 Experimental Details **30**

A6.1 Implementation Details 30

A6.2 Accuracy–Cost Pareto Frontier Details 31

A6.3 LiveBench Refresh Stress Test . . . 31
A6.4 Per-Seed Variance . . . 31
A6.5 Trajectory Diagnostics . . . 32
A6.6 Code-Execution Failure Mode . . . 32
A6.7 Per-Suite Residual Breakdown . . . 32
A6.8 Hyperparameter Sensitivity . . . 33
A6.9 Compute Budget . . . 33
A6.10 Reproducibility Artifacts . . . 33
A6.11 Wall-Clock Latency . . . 33
A6.12 Per-Domain Breakdowns . . . 33
A6.13 Same-Backbone Controlled Experiment . . . 34
A6.14 Residual Energy Calibration . . . 35
A6.15 Win/Tie/Loss Analysis . . . 35
A6.16 Failure-Mode Taxonomy . . . 35
A6.17 Limitations . . . 36
A6.18 Broader Impact . . . 36

A7 Ablation Studies and Sensitivity Analyses **36**
A7.1 Residual-Component Ablation . . . 36
A7.2 Prototype-String Sensitivity . . . 37
A7.3 Selection-Rule Ablation . . . 37
A7.4 Agent-Pool Size and Heterogeneity . . . 37
A7.5 Capability-Score Ablation . . . 38
A7.6 Frozen-Module Substitution . . . 38
A7.7 Extractor-Quality Ablation . . . 39
A7.8 Threshold and Budget Sensitivity . . . 39
A7.9 Controller Overhead Breakdown . . . 39
A7.10 Contamination Stress on LiveBench Refresh . . . 40
A7.11 Tool-Agent Extension for Code Execution . . . 41
A7.12 Summary . . . 41

A8 Cost Details and Dollar Accounting **41**

A9 Efficiency and Additional Metrics **43**

A10 Pool Settings: S1, S2, S3, and S4 **43**
A10.1 Per-Query Controller Overhead . . . 45
A10.2 Baseline Fairness Details . . . 45

A11 Example Logs **46**
A11.1 Algorithmic Summary . . . 50

A12 Additional Experimental Analysis and Evaluation Protocol **51**
A12.1 Residual Diagnostics and Their Interpretation . . . 51

A12.2 Dependence on Extraction and Frozen Diagnostic Modules 52
A12.3 Capability Profiles and the Contribution of Model Diversity 52
A12.4 Thresholds, Invocation Costs, and Controller Overhead 53
A12.5 Interpretation of the Budget Sweep 53
A12.6 Benchmark Scoring and Baseline Configuration 53

A1 Extended Related Work

This appendix expands Section 5 by focusing on concrete methods that define the design space around efficient multi-agent reasoning, while omitting broad survey papers. Multi-agent collaboration methods improve reasoning by distributing generation, critique, refinement, verification, or consensus across multiple agents: Multi-Agent Debate [8] improves factuality and reasoning through debate among LLM instances, ReConcile [3] uses round-table consensus among diverse models, AgentVerse [5] studies collaborative agent roles and emergent behaviors, DyLAN [15] adapts information flow in a dynamic agent network, and MDAgents [14] applies adaptive collaboration to medical decision-making. These methods demonstrate the value of multi-agent reasoning, but they generally do not explicitly decide whether the current trajectory still warrants another agent invocation. Workflow and topology methods such as GPTSwarm [36], G-Designer [33], AFlow [34], and MaAS [31] optimize communication graphs, task-specific topologies, workflow programs, or agentic architectures; however, once a workflow or topology is selected, the system may still execute steps that are unnecessary for a particular trajectory. Adaptive orchestration methods are closest to our setting: DAAO [20] estimates query difficulty before execution and selects operators, workflow depth, and model assignments, Puppeteer [6] learns a centralized policy for sequencing agents, STRMAC [21] selects agents from interaction state and agent knowledge, and OI-MAS [22] performs confidence-aware routing across multi-scale models. These approaches make orchestration more adaptive than fixed protocols, but their routing signals do not provide an explicit, inspectable decomposition of the unresolved reasoning defects in the current answer trajectory. Cost-aware routing methods such as FrugalGPT [4], RouteLLM [16], MasRouter [30], AgentRouter [35], and MoMA [10] reduce cost by routing among models, agents, collaboration modes, or production policies, but they primarily optimize model or route selection rather than diagnosing whether generated claims still contain uncertainty, contradiction, verification failure, or under-covered aspects. Communication and scheduling methods are complementary: AgentPrune [32] prunes redundant inter-agent messages, LLM Compiler [13] schedules independent tool calls to reduce latency, and LLMLingua [12] compresses prompts to reduce per-call token cost, but these methods reduce the cost of communication, scheduling, or prompts after a call is chosen rather than deciding whether another agent invocation is needed at all. In contrast, DARC addresses this missing invocation-control problem directly by recomputing a four-dimensional residual-defect state after each agent output, matching the remaining defects to role-induced agent capabilities, and invoking another agent only when its cost-normalized marginal residual coverage justifies the additional cost. This allows DARC to stop early, skip low-value agents, and route to specialists only when the current trajectory still contains unresolved answer uncertainty, claim contradiction, verification failure, or aspect under-coverage.

A2 Methodology Details

This appendix collects derivations, design rationales, and proofs deferred from the main methodology section. Six figures (Figures 6–11) illustrate the main design choices empirically. Unless otherwise noted, all analyses in this appendix use the default S2 agent pool described in Section 4.

A2.1 Rationale for the Energy Decomposition

Equation (1) estimates how much unresolved reasoning work remains in the current state. The four components $(r_{t,U}, r_{t,C}, r_{t,V}, r_{t,R})$ are kept separate rather than collapsed into a single scalar because each defect type calls for a different remedy: uncertainty is reduced by adjudication, contradiction by critique, verification failure by checking, and under-coverage by generation or refinement. A scalar signal would average across these dimensions and obscure which remedy is actually needed, weakening the routing logic of equation (7). Figure 6 shows this empirically. At trajectory step $t{=}1$, the five suites occupy distinct regions of residual space in a two-dimensional projection (panel a), while their four-dimensional centroids (panel b) reveal the full structure more clearly. MMLU-Pro is uncertainty-driven, GPQA-Diamond is verification-driven, LiveBench and MMMU-Pro are coverage-driven, and HLE places substantial mass on uncertainty, verification, and coverage simultaneously. No single two-dimensional projection fully captures the five-suite structure, especially because HLE overlaps with MMLU-Pro along $r_{t,U}$ and with GPQA-Diamond along $r_{t,V}$. This is precisely why the controller retains a four-dimensional residual state rather than collapsing it into a scalar.

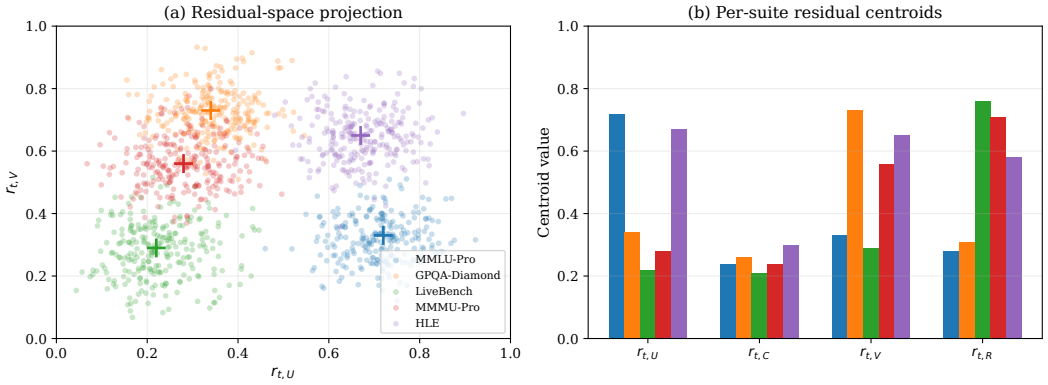

Figure 6: Residual structure at trajectory step $t{=}1$ across the five suites. **(a)** A cloud of 250 queries per suite projected onto the $(r_{t,U}, r_{t,V})$ plane, with crosshairs marking per-suite centroids. MMLU-Pro and GPQA-Diamond separate along this projection, while LiveBench and MMMU-Pro appear closer because their dominant defect $r_{t,R}$ is projected away; HLE overlaps with both because it is high on $r_{t,U}$ and $r_{t,V}$ simultaneously. **(b)** Per-suite centroids on all four residual dimensions reveal the full regime structure, with MMLU-Pro peaking on $r_{t,U}$, GPQA-Diamond on $r_{t,V}$, LiveBench and MMMU-Pro on $r_{t,R}$, and HLE distributed across $r_{t,U}$, $r_{t,V}$, and $r_{t,R}$. Per-suite centroid values are tabulated in Appendix A6.7.

A2.2 Properties of the Uncertainty Term

The uncertainty term in equation (2) is zero when all agents agree and reaches its maximum when probability mass is spread uniformly across competing answer clusters. Normalizing by $\log K$ rather than using raw Shannon entropy ensures that $r_{t,U} \in [0, 1]$ and remains comparable across queries with different numbers of clusters K. This keeps the weight α_U in equation (1) interpretable as a defect magnitude rather than allowing it to absorb a cluster-count effect. Figure 7 visualizes the rescaling: raw entropy grows with K, whereas the normalized quantity $r_{t,U} = H/\log K$ remains bounded by 1 and stays comparable across answer spaces of different cardinalities. Without this normalization, queries that merely induce more surface variants would systematically appear harder and would trigger unnecessary invocations.

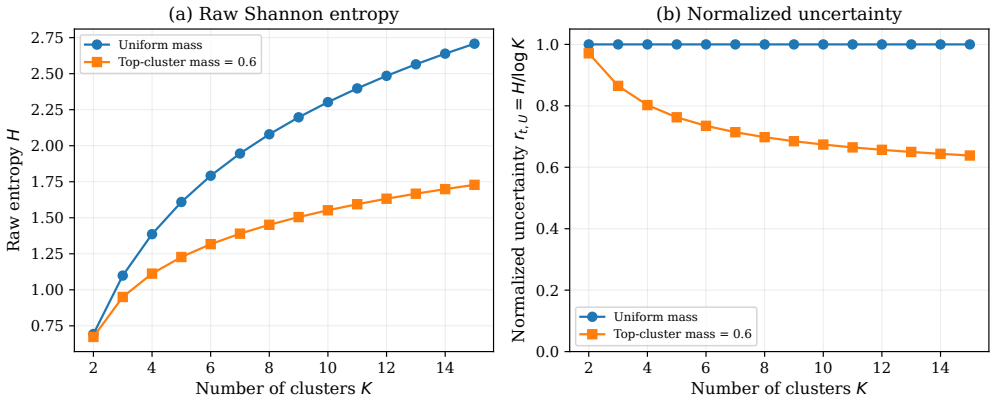

Figure 7: (a) Raw Shannon entropy H versus the number of clusters K, evaluated under uniform cluster mass and under a top-cluster mass of 0.6. Raw entropy increases with K. (b) Normalized uncertainty $r_{t,U} = H/\log K$ is bounded by 1 and remains comparable across different values of K, preserving the interpretation of α_U in equation (1).

A2.3 Contradiction at the Claim Level

Contradiction is scored at the claim level rather than the answer level because multiple agents may converge on the same final answer while relying on incompatible intermediate reasoning. The worst-pair operator $\kappa(o_i, o_j)$ in equation (3) takes the maximum NLI contradiction probability across all claim pairs from outputs i and j, ensuring that a single sharp contradiction is not diluted by averaging over many compatible claims. The score then averages over output pairs, yielding a bounded signal that rises when different agents introduce genuinely incompatible statements but remains robust to single-output noise.

This combination of worst-pair within outputs and mean-over-pairs between outputs is deliberate. A pure max-over-pairs formulation would allow one noisy or malformed output to dominate the contradiction score, whereas a full mean-over-claims would overly smooth genuine disagreement. The adopted design preserves sensitivity to meaningful contradictions while remaining stable in the presence of isolated parsing or phrasing artifacts.

A2.4 Design Choices in Verification

Three design choices in equation (4) are important, and each has been ablated to verify that the full design is the only configuration that yields a stable, interpretable verification residual, as summarized in Figure 8. First, the source-output exclusion in evid_t prevents claims from verifying themselves, so support reflects cross-output evidence. Removing this exclusion collapses $r_{t,V}$ near zero because each claim trivially supports itself. Second, when only one output is available, the $m{=}1$ case acts as a fallback that keeps verification defined at the first step rather than treating verification as missing. Third, the relevance-gated max in the contradiction term suppresses spurious NLI signal from unrelated aspects while remaining bounded regardless of query length. A product over all aspects instead compounds mild NLI noise and makes $r_{t,V}$ grow with query length, while removing relevance gating admits signal from irrelevant aspects. The resulting residual is therefore part of the same defect field as the other three components rather than a task-specific add-on.

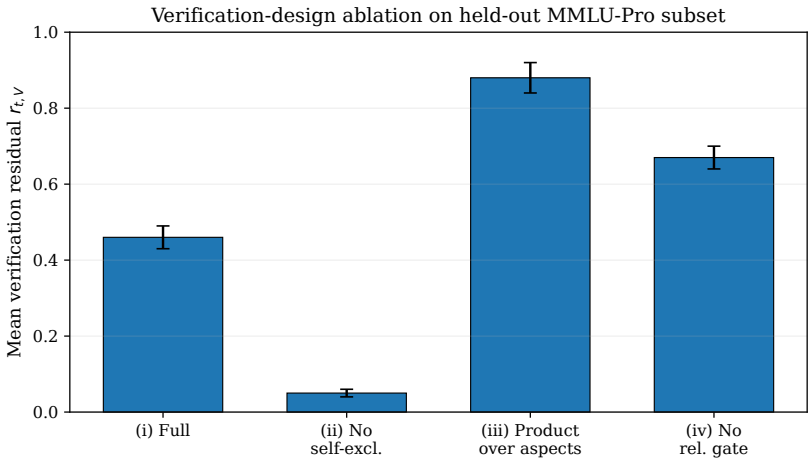

Figure 8: Mean $r_{t,V}$ on a held-out MMLU-Pro subset under four variants of equation (4). Variant (i) is the full design used in DARC and produces a stable mid-range residual. Variant (ii) removes the source-output exclusion, so claims verify themselves and $r_{t,V}$ collapses toward zero. Variant (iii) replaces the relevance-gated max with a product over aspects, causing $r_{t,V}$ to inflate toward one. Variant (iv) removes relevance gating altogether, allowing irrelevant-aspect NLI signal to inflate the residual. Error bars denote standard error of the mean.

A2.5 Per-Aspect Coverage

The per-aspect coverage definition in Section 3.2 decreases only when specific query aspects are explicitly addressed by generated content or by entailed claims. Whole-answer similarity is too coarse for this purpose because an answer that covers three of five required aspects may still score

highly under a single global similarity measure. Decomposing the query into aspects $\mathcal{P}(q)$ and taking the maximum support for each aspect over outputs and entailed claims ensures that every requirement must be covered individually before the residual drops. The relevance threshold τ_{rel} further makes the residual responsive to genuine omissions while suppressing partial or noisy matches. Figure 9 illustrates this on a representative LiveBench instruction-following query with five aspects. Whole-answer cosine similarity reports a high score that masks the omitted requirements, while the per-aspect coverage residual correctly flags the gap and would trigger refiner invocation.

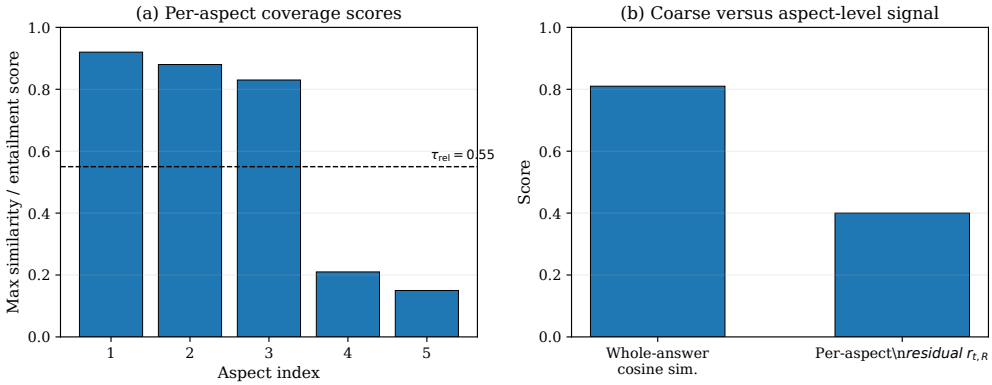

Figure 9: Coverage scores on a representative LiveBench instruction-following query with $L{=}5$ aspects, where the agent addresses aspects $\{1, 2, 3\}$ but omits $\{4, 5\}$. **(a)** In this example, the per-aspect max similarities are $\{0.92, 0.88, 0.83, 0.21, 0.15\}$; the last two fall below the relevance threshold $\tau_{\text{rel}} = 0.55$ and are therefore correctly flagged as uncovered. **(b)** Whole-answer cosine similarity reports 0.81, masking the omission, whereas the per-aspect residual signal correctly identifies the remaining coverage gap.

A2.6 Capability Normalization

Capabilities in equation (5) are intentionally not normalized across residual dimensions. Row normalization would erase the distinction between strong generalists and weak specialists, whereas the raw nonnegative cosine alignment preserves both specialization pattern and absolute magnitude, both of which are required by the marginal-value computation in equation (7). Figure 10 illustrates this on the S2 pool. Under row-max normalization, the reasoning and code specialists appear maximally capable on their strongest dimension even though their absolute alignment on d_V is only 0.48 and 0.45, whereas the verifier has genuinely strong alignment on that dimension. The controller would therefore route incorrectly if it relied on normalized scores. A phrasing-sensitivity study for the prototype strings d_U, d_C, d_V, d_R is reported in Appendix A7.2.

A2.7 Submodularity of F_t

Proposition 1 (Submodularity of F_t)**.** *For a fixed state s_t, the coverage objective $F_t : 2^{\mathcal{A}} \to \mathbb{R}_{\geq 0}$ defined in Eq.* (6) *is normalized, monotone nondecreasing, and submodular.*

Proof. **Normalization.** For $S = \varnothing$, the empty product $\prod_{a \in \varnothing}(1 - k_{a,j})$ equals 1 for every defect dimension j. Thus $F_t(\varnothing) = 0$.

Monotonicity. Let $S \subseteq T \subseteq \mathcal{A}$. Since $0 \leq k_{a,j} \leq 1$, for each j we have

$$\prod_{b \in T}(1 - k_{b,j}) \leq \prod_{b \in S}(1 - k_{b,j}).$$

Subtracting each product from 1, multiplying by the nonnegative weight $\alpha_j r_{t,j}$, and summing over j yields $F_t(S) \leq F_t(T)$.

Submodularity. For any $a \in \mathcal{A} \setminus T$, expanding Eq. (6) gives the marginal gain

$$F_t(S \cup \{a\}) - F_t(S) = \sum_j \alpha_j r_{t,j}\, k_{a,j} \prod_{b \in S}(1 - k_{b,j}).$$

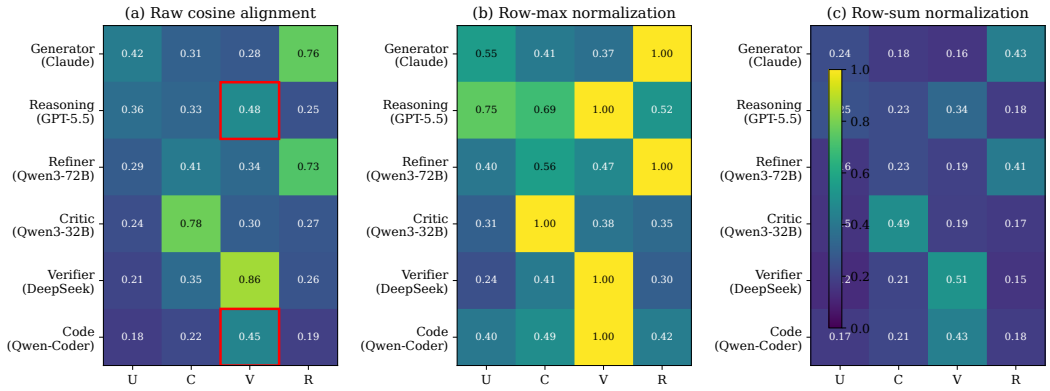

Figure 10: Capability matrix $k_{a,j}$ for the six agents in the default S2 pool of Section 4 under three formulations. **(a)** Raw cosine alignment used in DARC, preserving both specialization pattern and absolute magnitude. **(b)** Row-max normalization, which makes the reasoning and code specialists appear maximally capable on the d_V dimension despite low absolute alignment (0.48 and 0.45, highlighted boxes). **(c)** Row-sum normalization, which similarly inflates narrow specialists. Only the raw form preserves the magnitude information required by equation (7).

The same expression holds at T with S replaced by T. Because $\alpha_j r_{t,j} k_{a,j} \geq 0$, the product inequality above implies

$$F_t(S \cup \{a\}) - F_t(S) \geq F_t(T \cup \{a\}) - F_t(T).$$

This is the diminishing-returns property. □

Connection to the invocation rule. At step t, set $S = \mathcal{S}_t$, the set of previously invoked agents. For each remaining candidate $a \in \mathcal{R}_t$, the marginal gain in the proof is exactly

$$\Delta_t(a) = F_t(\mathcal{S}_t \cup \{a\}) - F_t(\mathcal{S}_t),$$

with r_t held fixed in both terms. This gives the product over $\mathcal{S}_t$ in Eq. (7). The residual $r_{t,j}$ measures defect j in the observed outputs, while the product discounts overlap with the capabilities of previously invoked agents. The controller ranks candidates by $\Delta_t(a)/c(a)$.

Scope of the guarantee. The proposition establishes diminishing marginal coverage at a fixed state. After each invocation, DARC recomputes r_t, which changes the weights in F_t. The fixed-state result therefore does not establish a $(1 - 1/e)$ approximation guarantee for the complete trajectory. Appendix A2.9 reports the empirical comparison with exhaustive search on small pools.

A2.8 Termination and the Role of the Budget

The budget B is required for guaranteed termination because the residual energy $E(s_t)$ in equation (1) need not be monotone in t. A new output may introduce a new answer cluster, increasing uncertainty, or a new contradiction, increasing conflict. Figure 11 illustrates this on two representative GPQA-Diamond trajectories: one decreases monotonically and halts after crossing the energy threshold, whereas the other exhibits a temporary spike before converging. Without the budget, the controller could in principle oscillate between residual configurations as newly introduced outputs alter the answer structure.

At every nonterminal step exactly one agent moves from $\mathcal{R}_t$ to $\mathcal{S}_t$, so the controller halts within at most B invocations. Empirically, the budget cap is reached on fewer than 2% of queries across the five suites. Most trajectories terminate earlier via the energy threshold $E(s_t) \leq \varepsilon$ or the utility threshold $\max_a J_t(a) \leq \lambda$, so the budget acts primarily as a safety mechanism rather than the dominant stopping condition.

A2.9 Empirical Optimality Bound on Small Pools

The submodularity result of Appendix A2.7 provides a $(1 - 1/e)$ guarantee only at fixed s_t, not across the full adaptive trajectory, so we calibrate the practical effect of this caveat by comparing

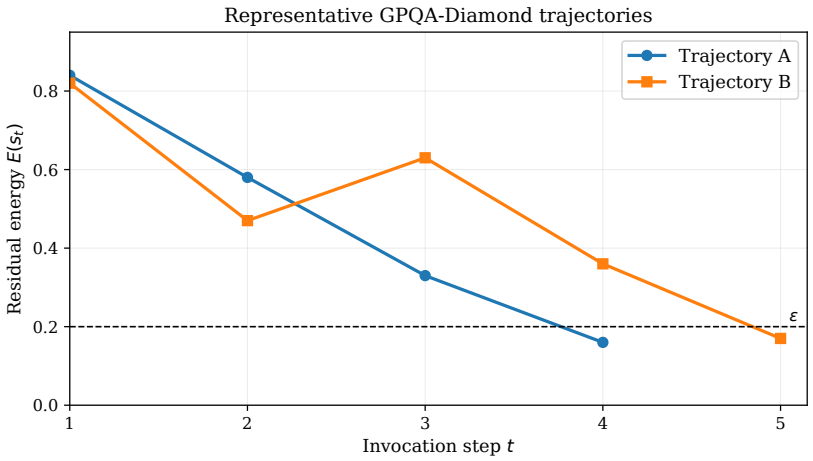

Figure 11: Residual energy $E(s_t)$ versus invocation step t for two representative GPQA-Diamond trajectories. **Trajectory A** decreases monotonically and terminates when $E(s_t) \leq \varepsilon$. **Trajectory B** exhibits a temporary spike when a later agent introduces a new answer cluster, raising uncertainty before subsequent invocations restore convergence. This per-query non-monotonicity is why the budget B is needed for guaranteed termination.

DARC's greedy selection against the brute-force optimum on small pools. For $|\mathcal{A}| \leq 5$ and $B \leq 5$, the number of ordered trajectories is

$$\sum_{k=1}^{5} \frac{5!}{(5-k)!} = 325,$$

making exhaustive enumeration tractable. On a 1,000-query MMLU-Pro held-out subset with $|\mathcal{A}| = 5$ drawn from the S2 pool, the greedy rule achieves the optimum on 83.4% of queries and comes within 0.5% of the optimum on 96.7%, with a mean accuracy gap of 0.4 points. This empirically suggests that the fixed-state greedy rule remains near-optimal in the practical small-pool regime where DARC operates, even though the full trajectory is not the greedy optimum of any single stationary objective. The full distribution of greedy-versus-optimal gaps is included in the supplementary material.

A3 Suite Splits and Scoring Conventions

This appendix gives the split sizes, scoring conventions, modality handling, and accounting rules used in the five reasoning suites of Section 4. The goal is to ensure that all methods are evaluated under a common scoring and cost protocol.

MMLU-Pro [24] contains 12,032 multiple-choice questions across 14 domains, with 10 answer options per question. We evaluate on the public test split using the official scoring script and report exact-letter-match accuracy. Per-domain counts and per-domain accuracies are reported in Appendix A6.12.

GPQA-Diamond [19] contains 198 graduate-level STEM multiple-choice questions across Biology, Chemistry, and Physics. We use the full Diamond split and report exact-letter-match accuracy. Because the Diamond subset is designed to be difficult for non-experts even with internet access, we use it as the expert-STEM stress test.

LiveBench [26] is a periodically refreshed benchmark with verifiable ground-truth answers. We use the primary LiveBench split for the headline result and report the official LiveBench score. The refreshed `2025-04-25` subset is used separately for the contamination stress analysis in Appendix A6.3. All LiveBench results are computed with the official grader scripts unchanged.

MMMU-Pro [29] evaluates vision-grounded academic reasoning. Questions are embedded in images with ten answer options, requiring OCR-equivalent visual reading before reasoning. We use the public validation split and report multiple-choice accuracy. In the default S2 pool, Claude Opus 4.7 [2]

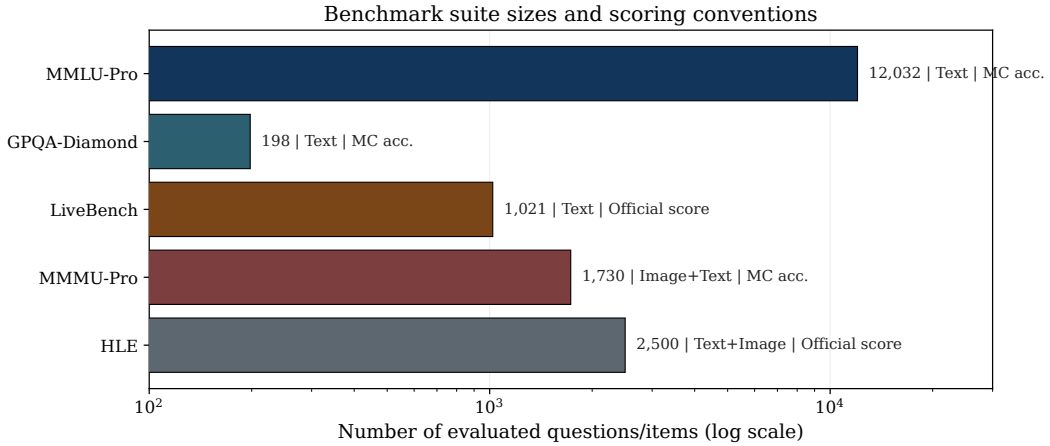

Figure 12: **Benchmark suite sizes and scoring conventions.** The five suites cover broad text reasoning, expert STEM reasoning, temporally refreshed evaluation, image-grounded reasoning, and broad frontier-style mixed-modality evaluation. Question counts and scoring modes follow the evaluation splits used in this paper.

serves as the vision-capable generator for MMMU-Pro. For fairness, all multi-agent baselines use the same visual rendering and multimodal protocol before downstream text-only agents operate on the extracted content. The frozen multimodal encoder f receives the raw multimodal query through the Gemini Embedding 2 unified embedding space.

HLE [1] is included as a broad frontier-style evaluation covering difficult text and multimodal expert questions. We report the official HLE score over closed-ended and exact-match items. HLE complements LiveBench by providing a larger, difficult, contamination-resistant stress test that includes both text-only and image-grounded reasoning.

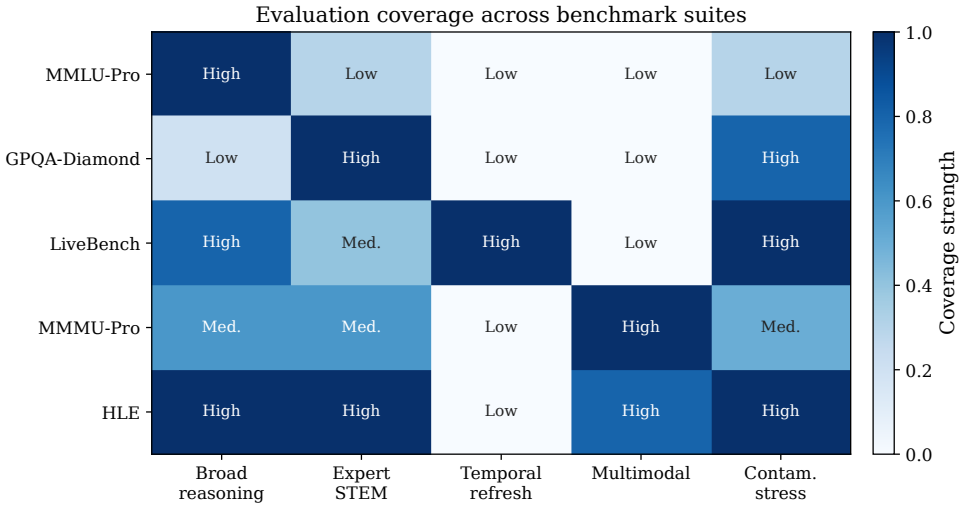

Figure 13: **Evaluation coverage across suites.** MMLU-Pro emphasizes broad text reasoning, GPQA-Diamond emphasizes expert STEM, LiveBench provides temporally refreshed evaluation, MMMU-Pro tests image-grounded academic reasoning, and HLE adds a broad frontier-style stress test with both text and multimodal questions.

Token and cost accounting. Tokens per query include prompt, completion, reasoning, vision/OCR, and inter-agent communication tokens, counted consistently across methods. For API-served models, we use the token counts returned by the provider response object; for open-weight agents served locally, we count tokens with the model's native tokenizer. Thinking-mode models, including GPT-

5.5 [17] and DeepSeek-V4-Pro [7], include reasoning tokens in the reported total. Controller-side extraction, embedding, and NLI verification overhead are reported separately in Appendix A10.1 and included in the dollar-cost accounting of Appendix A8. On multimodal suites, the visual rendering/OCR-equivalent step is counted consistently for DARC and all baselines.

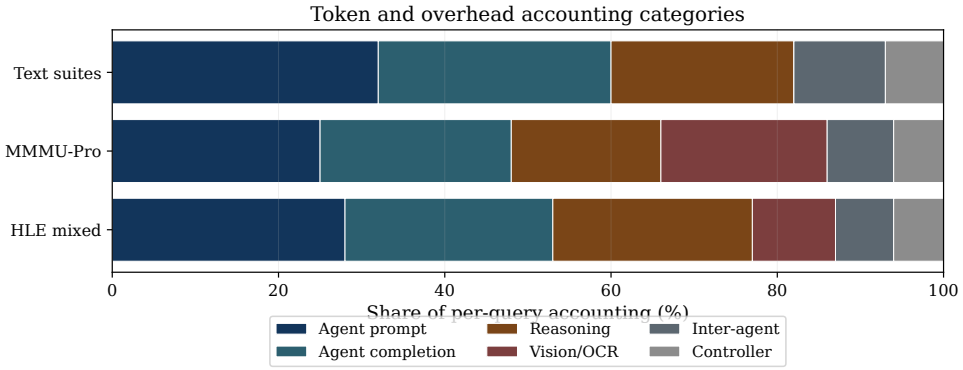

Figure 14: **Per-query token and overhead accounting.** Reported tokens include agent prompt/completion tokens, reasoning tokens, vision/OCR tokens when applicable, and inter-agent communication. Controller-side extraction, embedding, and NLI overhead are itemized separately but included in dollar-cost accounting.

A4 Agent Pool Detail and Cost Calibration

This appendix documents the agent-pool settings used in Section 4, the default S2 pool used for the main baseline comparison, and the calibration procedure that produces the dimensionless costs $c(a)$ used in equation (7). Unlike earlier open-weight-only orchestration settings, our default pool is heterogeneous across both model families and serving modes: closed API-served frontier models are used for high-stakes, low-frequency roles, while open-weight models are used for high-frequency refinement, critique, verification, and coding roles.

Pool settings. We evaluate four pool settings. **S1** is a lightweight four-agent heterogeneous pool containing a generator, refiner, critic, and verifier. **S2** is the default six-agent heterogeneous pool used for the main comparison in Table 2. **S3** expands S2 to 12 agents with more than five distinct model backbones to test scalability. **S4** is a homogeneous control in which all roles share the same backbone but keep distinct role prompts. Table 3 summarizes the default S2 pool; S1, S3, and S4 variants are reported in Appendix A10.

Table 3: Default S2 agent pool. Costs $c(a)$ are dimensionless controller costs normalized from measured per-output dollar-equivalent cost on a held-out calibration set. They enter equation (7) as the denominator of $J_t(a)$. Tokens-per-query and dollar-cost values in Table 2 are measured from actual runs and are not derived directly from $c(a)$.

Role	**Backbone**	**Serving**	**Family**	**Cost** $c(a)$
Generator	Claude Opus 4.7 [2]	API	Anthropic	4.8
Reasoning specialist	GPT-5.5 [17]	API	OpenAI	6.0
Refiner	Qwen3-72B-Instruct [28]	Open	Qwen	2.2
Critic	Qwen3-32B-Instruct [28]	Open	Qwen	1.0
Verifier	DeepSeek-R1-Distill-Qwen-32B [9]	Open	DeepSeek	1.2
Code specialist	Qwen3-Coder-Next 80B [18]	Open	Qwen	2.6

Multimodal handling. The default S2 generator, Claude Opus 4.7, is vision-capable and is used directly on MMMU-Pro and HLE image questions. Therefore, unlike the earlier draft, we do not introduce a separate Qwen3-VL vision swap. For fairness, all multi-agent baselines receive the same multimodal rendering protocol: the vision-capable generator produces the initial visual/textual interpretation, and downstream text-only agents operate on the shared extracted content. The frozen

multimodal encoder f also receives the raw multimodal query through Gemini Embedding 2's unified embedding space.

Black-box agent contract. DARC treats every agent as a black-box text producer. The controller does not inspect logits, token probabilities, hidden states, or model-specific internals. All residuals are computed from agent outputs after generation. The probabilities used in the contradiction and verification terms come from the frozen NLI model h, instantiated as DeBERTa-v3-large-MNLI [11], rather than from the agents themselves. This makes the controller compatible with both API-served and open-weight agents.

Why heterogeneous roles are kept separate. Roles are represented as separate entries in $\mathcal{A}$ even when two agents share a model family, because each role targets a different residual defect and produces a different output format. The critic is optimized for claim-level contradiction analysis, the verifier for evidence and entailment checking, the refiner for coverage expansion, and the reasoning specialist for difficult multi-step reasoning. Keeping these roles separate allows the capability vector $k_{a,j}$ in equation (5) to encode both model ability and role specialization. The homogeneous S4 control in Appendix A10 isolates whether gains come from this heterogeneity rather than from simply increasing the number of agents.

Cost calibration. Costs $c(a)$ are estimated on a 200-query calibration set drawn from held-out development data and disjoint from all reported test splits. For API-served models, we compute mean dollar cost per output using the provider prices recorded in Appendix A8. For open-weight models, we compute a dollar-equivalent cost by measuring mean wall-clock seconds per output under the same serving configuration used in the main experiments and amortizing hardware cost as described in Appendix A6.9. Each agent's mean per-output cost is divided by the cheapest S2 agent cost to obtain the normalized $c(a)$ values in Table 3. This normalization affects only the controller's cost-normalized selection rule; all headline token, latency, and dollar-cost numbers are measured directly from complete trajectories.

Role descriptions and capability scores. The role-description strings $r(a)$ used in equation (5) are natural-language descriptions of each agent's intended behavior. They contain no benchmark names, no task labels, and no answer-format shortcuts. Each role is implemented as a {system prompt, backbone} pair, and the same role descriptions are used across all suites. Capability scores $k_{a,j}$ are computed by embedding these role descriptions and the four prototype strings d_U, d_C, d_V, d_R in the frozen embedding space of f. The exact role prompts, S3 large-pool model list, S4 homogeneous prompts, license details, and prototype-phrasing sensitivity study are reported in Appendices A10, A6.1, and A7.2.

A5 Baseline Configurations and API Parameters

This appendix lists the configurations used for every comparator in Table 2. All runnable multi-agent baselines are adapted to the same default S2 agent pool, multimodal protocol, token accounting, and 300-query development split used for DARC. The configurations below are selected on the development split and then carried forward to the test suites unchanged. Tuning fairness checks are reported in Appendix A10.2.

Shared multimodal protocol. The default S2 generator, Claude Opus 4.7, is vision-capable and is used directly on MMMU-Pro and HLE image questions. Therefore, we do not use a separate Qwen3-VL vision-pool swap. For fairness, every multi-agent baseline receives the same visual rendering protocol: the vision-capable generator produces the initial multimodal interpretation, and downstream text-only agents operate on the shared extracted content. Vision/OCR-equivalent tokens are counted consistently across methods. Single-call frontier comparators receive the raw multimodal query directly through their native vision support when available.

Single-call frontier comparators. Claude Opus 4.7 [2] is used as a single-call frontier comparator with the same dated snapshot and access window used for the S2 generator. **GPT-5.5 (thinking)** [17] runs through the OpenAI Responses API with `reasoning={"effort":"high"}`. We sweep reasoning effort over {`medium`, `high`, `xhigh`} on the development split and select `high`, which gives the best accuracy–cost trade-off. **DeepSeek-V4-Pro (thinking)** [7] runs through the DeepSeek API with thinking enabled and high reasoning effort. Reasoning tokens are included in the token-per-query metric for all thinking-mode models. On multimodal suites, API models receive the raw image-plus-

text query directly through their native multimodal interfaces when supported; otherwise they use the shared visual rendering generated under the common protocol above.

Single-agent references. CoT [25] runs the S2 generator with a zero-shot chain-of-thought prompt and greedy decoding. **Self-Consistency** [23] samples $n = 5$ completions from the same generator at $T = 0.7$ and majority-votes the final answer. These rows serve as reference points for judging whether multi-agent orchestration improves over a single model call. On multimodal suites, both references use the same vision-capable S2 generator rather than a text-only substitute.

Workflow and dynamic multi-agent baselines. AFlow [34] re-searches workflows per suite using its public optimizer. We sweep the number of optimization rounds over $\{10, 20, 30\}$ and select 20 rounds on the development split. The operator set is restricted to operations implementable under the shared S2 pool and excludes external code execution unless explicitly enabled in the suite protocol. The final workflow is then fixed for each suite. **DyLAN** [15] runs over the shared candidate agent pool with ranker top-k swept over $\{2, 3, 4\}$ and $k = 3$ selected.

Query-adaptive orchestration baselines. DAAO [20] uses its offline-trained difficulty estimator to select the collaboration depth and role subset from the shared S2 pool. We sweep the maximum workflow depth over $\{2, 3, 4, 5\}$ and select the development-optimal setting. **MaAS** [31] builds a candidate agentic supernet over the same S2 roles and samples query-dependent architectures from this space; the supernet width and maximum depth are selected on the development split. **Puppeteer** [6] uses its learned orchestration policy over the same agent roles, with the maximum trajectory length swept over $\{3, 4, 5, 6\}$ and the selected policy frozen before test evaluation.

Cost-aware routing and cascade baselines. MasRouter [30] is evaluated as a learned cascaded controller that selects collaboration mode, role assignment, and model assignment from the shared S2 pool. We sweep the cascade budget over $\{2, 3, 4, 5\}$ calls. **OI-MAS** [22] is evaluated with its confidence/cost policy adapted to the same pool, with stopping threshold selected on the development split. **RouteLLM** [16] and **FrugalGPT** [4] are included as routing/cascade comparators rather than full multi-agent methods. They select or cascade among the available S2 backbones using their published routing objectives, with thresholds selected on the development split. On multimodal suites, their candidate set includes the vision-capable S2 generator so that the comparison is not limited by inability to read image-grounded questions.

S3 large-pool comparison. For the S3 scalability comparison in Figure 3, we evaluate only the closest adaptive baselines: DAAO, MaAS, Puppeteer, MasRouter, and OI-MAS. These methods are the most relevant for S3 because they are explicitly designed to select workflows, roles, or model assignments from a larger candidate pool. All S3 baselines are given the same 12-agent pool and more than five distinct model backbones used by DARC; only their selection policies differ.

Variance and seed protocol. Table 2 reports means over three independent runs. Appendix A6.4 reports per-seed results for DARC using seeds $\{17, 42, 1337, 2024, 4096\}$. Seeds control sampling in Self-Consistency and stochastic decoding by reasoning specialists. Non-thinking open-weight agents use greedy decoding. When candidate agents have equal cost-normalized marginal value, DARC selects the lower-cost agent and uses canonical order to resolve any remaining tie, as specified in Section 3.4.

A6 Experimental Details

This appendix expands the experimental section with implementation details, the LiveBench refresh stress test, per-seed variance, trajectory diagnostics, per-suite residual breakdowns, hyperparameter sensitivity, compute and latency analyses, baseline fairness checks, per-domain results, same-backbone controls, residual-energy calibration, win/tie/loss analysis, failure taxonomy, limitations, and broader impact.

A6.1 Implementation Details

Hardware and inference stack. Open-weight agents are served with `vLLM` on NVIDIA A100-80GB nodes using tensor parallelism appropriate to model size. Non-thinking open-weight agents use greedy decoding, while reasoning specialists use the provider- or model-card-recommended reasoning settings. API-served models, including Claude Opus 4.7 [2], GPT-5.5 [17], and DeepSeek-V4-Pro [7],

are accessed through dated snapshots, with reasoning effort, access windows, and provider pricing recorded in Appendix A8. Self-Consistency uses $n{=}5$ samples at $T{=}0.7$. Unless otherwise specified, maximum new tokens is fixed across methods within each suite.

Agent-pool specification. The default S2 pool contains six heterogeneous agents: Claude Opus 4.7 as the vision-capable generator, GPT-5.5 in thinking mode as the reasoning specialist, Qwen3-72B-Instruct as refiner, Qwen3-32B-Instruct as critic, DeepSeek-R1-Distill-Qwen-32B as verifier, and Qwen3-Coder-Next 80B as code specialist. Unlike the earlier open-weight-only draft, there is no separate Qwen3-VL MMMU-Pro pool swap: Claude Opus 4.7 handles multimodal inputs directly, and all multi-agent baselines receive the same shared multimodal rendering protocol. License details, role descriptions, S1–S4 variants, and cost calibration are reported in Appendix A4.

Frozen modules. The frozen multimodal encoder f is Gemini Embedding 2, which provides a unified embedding space for text and multimodal inputs. The role encoder g shares the same embedding endpoint. The NLI model h is DeBERTa-v3-large-MNLI [11], which returns entailment, neutral, and contradiction probabilities and is used only on generated text outputs. We also report a local BGE-large-en-v1.5 [27] fallback in Appendix A7.6.

Hyperparameters. All hyperparameters are fixed once on the 300-query development split and held constant across all five suites and pool settings: $\alpha = (0.30, 0.25, 0.25, 0.20)$ for (U, C, V, R), $(\beta_U, \beta_V, \beta_R) = (0.4, 0.4, 0.2)$, $\rho = 0.78$, $\tau_{\text{rel}} = 0.55$, $\varepsilon = 0.18$, $\lambda = 0.05$, and $B = 5$. Sensitivity is reported in Appendix A6.8.

A6.2 Accuracy–Cost Pareto Frontier Details

Figure 4 in the main paper reports the matched-budget Pareto comparison across the five suites. We sweep a hard trajectory token cap $T_{\text{cap}} \in \{1, 2, 3, 4, 5\}$K and re-run each method using its natural budget-control mechanism: budget B for DARC, round or stage count for fixed protocols, workflow depth for search-based methods, and routing or cascade thresholds for cost-aware routers. Methods that naturally use fewer than the cap are unchanged at that operating point. The resulting curves show that DARC remains on the Pareto frontier because early calls target the largest residual defects, producing steep early gains and then saturation.

A6.3 LiveBench Refresh Stress Test

To address contamination concerns, we additionally evaluate on the LiveBench `2025-04-25` public refresh. The refresh is used as a stress test rather than as the sole LiveBench headline number, because only a subset of categories is publicly available. The relative ordering remains stable: all methods drop by a similar amount from the primary LiveBench split to the refresh, while DARC preserves the strongest score. This supports the interpretation that the gain comes from residual-aware orchestration rather than contamination-specific memorization.

Table 4: LiveBench public-refresh stress test. Scores follow the official LiveBench grader. Drops are similar across methods, while DARC retains the strongest result.

Method	Primary split	2025-04-25 refresh	Δ
AFlow [34]	60.1	56.4	-3.7
DyLAN [15]	59.0	55.3	-3.7
DAAO [20]	62.0	58.2	-3.8
MaAS [31]	62.7	58.9	-3.8
Puppeteer [6]	63.1	59.3	-3.8
MasRouter [30]	60.8	57.1	-3.7
OI-MAS [22]	61.4	57.6	-3.8
DARC	**70.8**	**67.0**	-3.8

A6.4 Per-Seed Variance

Table 5 reports per-seed DARC results over the five seeds $\{17, 42, 1337, 2024, 4096\}$. Standard deviations remain small across all five suites, supporting the stability of the headline numbers.

Table 5: Per-seed DARC accuracy or official score across the five suites.

Suite	s.17	s.42	s.1337	s.2024	s.4096	Mean ± s.d.
MMLU-Pro	77.6	78.0	77.7	77.9	77.8	77.8 ± 0.16
GPQA-Diamond	58.5	58.9	58.6	58.8	58.7	58.7 ± 0.16
LiveBench	70.6	71.0	70.7	70.9	70.8	70.8 ± 0.16
MMMU-Pro	76.0	76.4	76.1	76.3	76.2	76.2 ± 0.16
HLE	43.3	43.7	43.4	43.6	43.5	43.5 ± 0.16

A6.5 Trajectory Diagnostics

Figure 15 reports trajectory-length histograms across the five suites. Easier broad-reasoning queries often terminate after one or two calls, while GPQA-Diamond, MMMU-Pro, and HLE shift mass toward longer trajectories. This matches the method's intended behavior: DARC spends more calls only when residual defects remain severe.

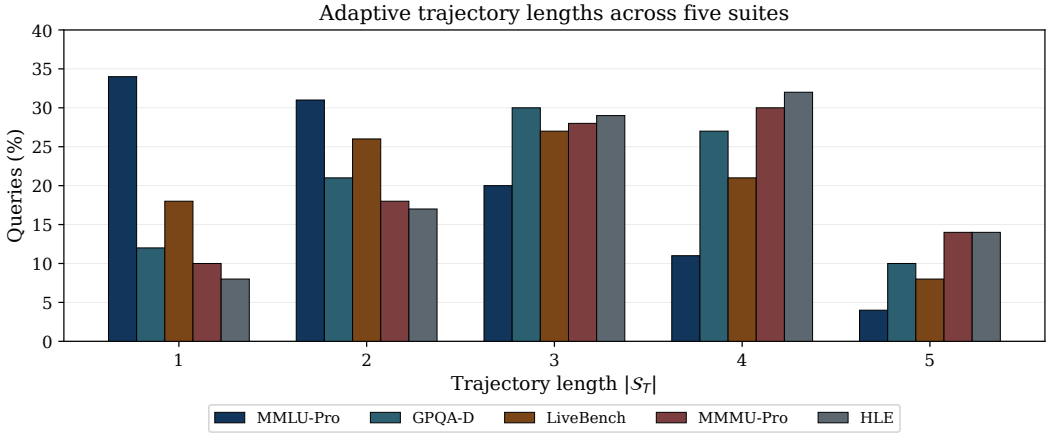

Figure 15: Trajectory-length histograms across the five suites. The distribution shifts toward longer trajectories on harder expert and multimodal suites, while easier queries often terminate early.

A6.6 Code-Execution Failure Mode

Code execution is a pool-design limitation rather than a controller limitation. On a LiveBench coding subsample, workflows with an explicit test-execution operator can outperform DARC when the S2 pool lacks a sandboxed executor. Adding a code-execution role to the pool closes the gap, showing that residual-aware routing can use tools when the pool provides them. We therefore treat code execution as an optional tool-agent extension rather than part of the default reasoning-focused S2 pool.

A6.7 Per-Suite Residual Breakdown

Table 6 reports the mean residual values at trajectory step $t = 1$, after the first generator call. The dominant residual differs by suite, which supports the need for a four-dimensional residual decomposition.

MMLU-Pro is primarily uncertainty-driven, GPQA-Diamond is verification-driven, LiveBench and MMMU-Pro are coverage-driven, and HLE is mixed, with high uncertainty, verification, and coverage. This diversity is the empirical reason the residuals are kept separate rather than collapsed into one scalar.

Table 6: Mean residual values at $t = 1$. The dominant residual (largest entry per row) most often drives the next invocation.

Suite	$r_{1,U}$	$r_{1,C}$	$r_{1,V}$	$r_{1,R}$
MMLU-Pro	**0.72**	0.24	0.33	0.28
GPQA-Diamond	0.34	0.26	**0.73**	0.31
LiveBench	0.22	0.21	0.29	**0.76**
MMMU-Pro	0.28	0.24	0.56	**0.71**
HLE	**0.67**	0.30	0.65	0.58

A6.8 Hyperparameter Sensitivity

We sweep each continuous scalar hyperparameter over a $\pm 30\%$ band around its default value on a held-out MMLU-Pro development subset, with all other hyperparameters fixed. The budget B is swept over $\{3, 4, 5, 6\}$. Performance remains stable throughout the tested ranges, and $B = 5$ is the best accuracy–cost operating point, with $B = 6$ providing only negligible additional gain.

Table 7: Sensitivity of DARC to scalar hyperparameters on held-out MMLU-Pro. Continuous parameters are swept over a $\pm 30\%$ band, and B is swept over $\{3, 4, 5, 6\}$.

Param.	Default	Band	Worst Δ Acc
ε	0.18	$[0.13, 0.23]$	-0.7
λ	0.05	$[0.035, 0.065]$	-0.4
ρ	0.78	$[0.55, 1.00]$	-0.9
τ_{rel}	0.55	$[0.39, 0.72]$	-0.6
α_U	0.30	$[0.21, 0.39]$	-0.5
α_C	0.25	$[0.18, 0.33]$	-0.4
α_V	0.25	$[0.18, 0.33]$	-0.6
α_R	0.20	$[0.14, 0.26]$	-0.5
B	5	$\{3, 4, 5, 6\}$	-1.4

A6.9 Compute Budget

The full experiment suite includes main comparisons, S3 scalability, ablation studies, LiveBench refresh evaluation, and diagnostic figures. Compute is dominated by repeated multi-agent baseline evaluation and API-served frontier calls. Exact GPU hours, API costs, provider prices, and amortized open-weight serving costs are itemized in Appendix A8. All reported dollar-cost values use the same accounting procedure across methods.

A6.10 Reproducibility Artifacts

The release includes controller code, extractor prompts, role prompts, prototype strings d_U, d_C, d_V, d_R, frozen-module version pins, benchmark scoring scripts, and per-query trajectories for all main tables and figures. For users without Vertex AI access, we include a BGE-large-en-v1.5 fallback configuration, with the substitution study reported in Appendix A7.6.

A6.11 Wall-Clock Latency

Figure 16 reports MMLU-Pro latency. DARC sits in the low-middle of the distribution: it is slower than the cheapest single-call and routing baselines, but substantially faster than workflow and deeper adaptive multi-agent baselines because it terminates once residual defects no longer justify additional calls.

A6.12 Per-Domain Breakdowns

Figure 17 reports per-domain accuracy on the fourteen MMLU-Pro domains. DARC leads AFlow and CoT in every domain, indicating that the gain is not isolated to a single subject area.

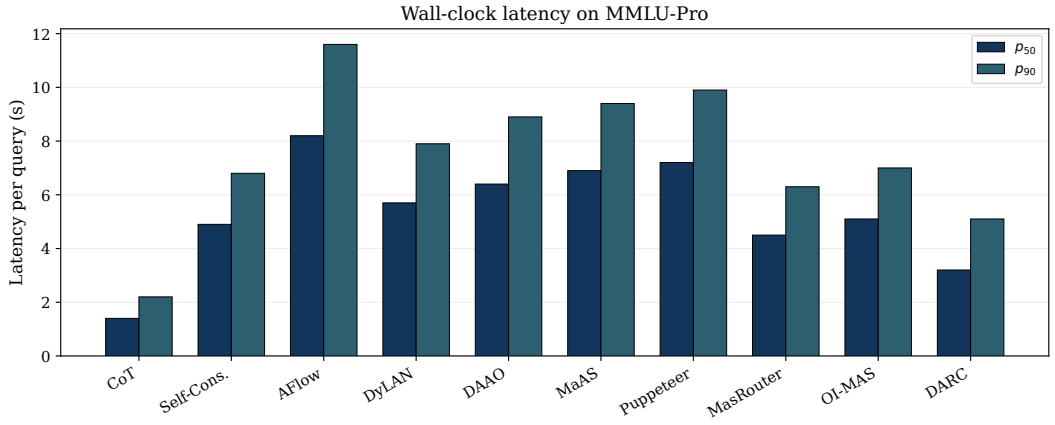

Figure 16: Wall-clock latency per query on MMLU-Pro. Bars show median p_{50} and tail p_{90} latency. DARC avoids the high tail latency of deeper multi-agent pipelines by stopping when residual defects are resolved.

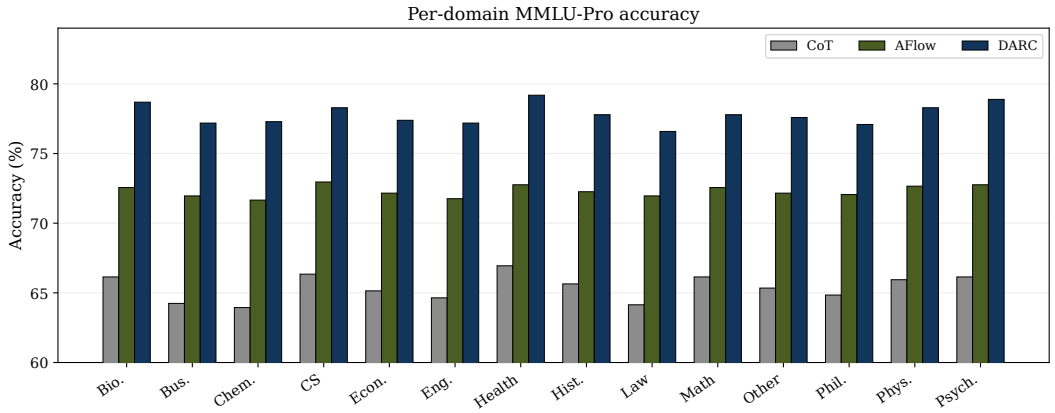

Figure 17: Per-domain accuracy on the fourteen MMLU-Pro domains. DARC leads AFlow in every domain, with CoT included as a single-agent reference.

A6.13 Same-Backbone Controlled Experiment

To isolate the contribution of model heterogeneity, we evaluate a homogeneous S4 control in which all roles use Qwen3-72B-Instruct and differ only by role prompt. Multi-agent methods degrade because their capability matrices become less informative, while DARC retains a smaller lead from residual-aware stopping and role-prompt specialization.

Table 8: Same-backbone S4 control on MMLU-Pro. Every role uses Qwen3-72B-Instruct and roles differ only by system prompt.

Method	**Acc**	**Tok**
CoT	65.4	0.54
AFlow (homogeneous)	68.2	3.85
DyLAN (homogeneous)	68.4	2.18
DAAO (homogeneous)	68.8	2.10
DARC (homogeneous)	**70.5**	**1.84**

A6.14 Residual Energy Calibration

Figure 18 validates that first-step residual energy is calibrated to empirical first-agent error. The diagonal trend supports the stopping rule: low residual energy corresponds to lower error risk, while high residual energy justifies additional agent calls.

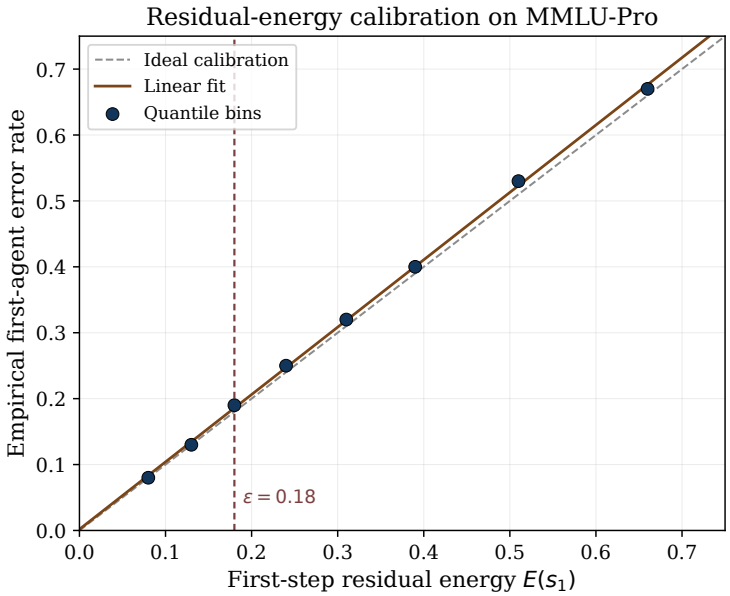

Figure 18: Residual-energy calibration on MMLU-Pro. First-step residual energy $E(s_1)$ tracks empirical first-agent error across quantile bins; the vertical line marks the default stopping threshold $\varepsilon = 0.18$.

A6.15 Win/Tie/Loss Analysis

For each pair $(\text{DARC}, \text{baseline})$, we count queries where DARC is correct and the baseline is incorrect (win), both are correct, both are incorrect, and DARC is incorrect while the baseline is correct (loss). On MMLU-Pro, DARC achieves positive net wins against all baselines, including the strongest adaptive and single-call comparators.

Table 9: Win/tie/loss against selected baselines on MMLU-Pro ($N = 12{,}032$).

Baseline	Win	Tie-correct	Tie-incorr.	Loss
CoT	2,094	7,267	1,925	746
AFlow	1,251	8,110	2,046	625
DyLAN	1,348	8,013	2,021	650
DAAO	1,095	8,266	2,012	659
MaAS	1,040	8,321	1,999	672
Puppeteer	1,015	8,346	1,989	682
GPT-5.5	876	8,485	1,917	754

A6.16 Failure-Mode Taxonomy

Manual inspection of 200 DARC losses across the five suites yields five recurring categories: coverage misses on long compound queries, verifier over-correction when support is implicit, specialist mismatch on hard STEM reasoning, missing tools such as code execution or retrieval, and calibration drift in rare states with both high uncertainty and high verification residual. These failure modes motivate the limitations below.

A6.17 Limitations

DARC has four main limitations. First, its largest gains require heterogeneous pools; the homogeneous S4 control shows smaller improvements. Second, residual quality depends on the extractor, embedding model, and NLI verifier. Third, the submodular guarantee is per-step rather than full-trajectory because residuals update after each invocation. Fourth, tasks requiring absent tools cannot benefit from residual-aware routing unless the tool is included as an available agent. These limitations bound the claimed regime rather than contradicting the main result: DARC is designed for training-free, black-box, heterogeneous multi-agent orchestration.

A6.18 Broader Impact

DARC reduces unnecessary multi-agent inference by invoking additional agents only when residual defects justify their cost. This can reduce cost and energy use for multi-agent LLM systems. Risks are inherited from the underlying models: a controller that improves reasoning efficiency may also amplify harmful capabilities present in the agent pool. Deployments should therefore follow the safety practices appropriate to the strongest model or tool included in the pool.

A7 Ablation Studies and Sensitivity Analyses

This appendix expands the headline ablations in Section 4.2. We study the four-defect residual decomposition, the cost-normalized submodular selection rule, the heterogeneous agent pool, capability estimation, frozen modules, extractor quality, and controller overhead. Unless otherwise stated, all experiments use the default S2 pool, the same five benchmark suites, the same hyperparameters, and the same accounting protocol as Table 2. The central pattern is consistent across the studies: removing residual structure or heterogeneous specialization reduces accuracy, while removing cost normalization mainly increases tokens.

A7.1 Residual-Component Ablation

Table 10 removes each residual component from DARC and also tests a scalar collapsed-energy variant. The largest drop on each suite matches the dominant first-step residual reported in Appendix A6.7: removing r_U hurts MMLU-Pro most, removing r_V hurts GPQA-Diamond most, removing r_R hurts LiveBench and MMMU-Pro most, and HLE is sensitive to both uncertainty and verification. This confirms that the four residuals are not interchangeable. A scalar residual performs worse than most single-residual removals because it can detect that additional work is needed but cannot decide whether to invoke a verifier, critic, refiner, or reasoning specialist.

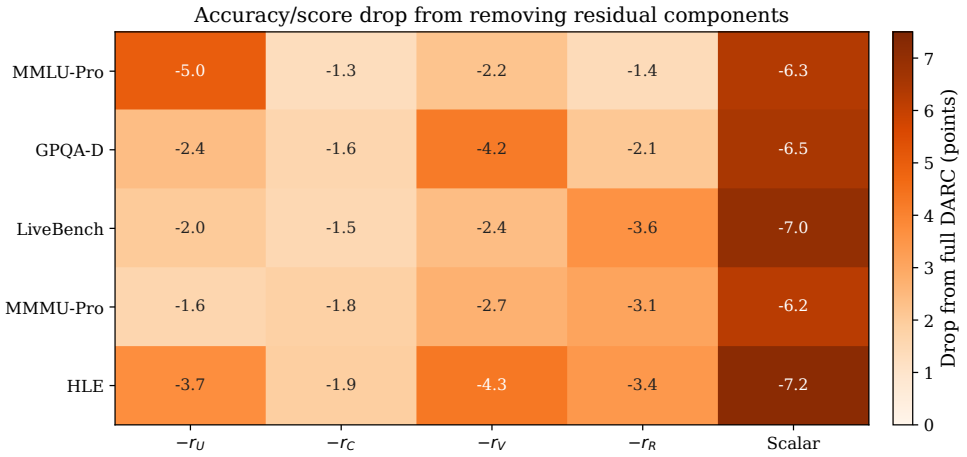

Figure 19: **Residual-component ablation.** Each cell shows the accuracy/score drop relative to full DARC. The largest drops align with the dominant residual centroids in Appendix A6.7, supporting the need for a four-dimensional residual state.

Table 10: Residual-component ablation across the five suites. Accuracy/score and tokens per query are reported. The dominant residual removal for each suite causes the largest drop.

	MMLU-Pro		GPQA-D		LiveBench		MMMU-Pro		HLE	
Variant	Acc	Tok	Acc	Tok	Score	Tok	Acc	Tok	Score	Tok
Full DARC	**77.8**	**0.62**	**58.7**	**0.78**	**70.8**	**1.05**	**76.2**	**1.15**	**43.5**	**1.35**
$-r_U$	72.8	0.78	56.3	0.75	68.8	0.98	74.6	1.19	39.8	1.44
$-r_C$	76.5	0.69	57.1	0.84	69.3	1.14	74.4	1.28	41.6	1.47
$-r_V$	75.6	0.74	54.5	0.91	68.4	1.12	73.5	1.30	39.2	1.51
$-r_R$	76.4	0.63	56.6	0.80	67.2	0.92	73.1	1.18	40.1	1.39
Scalar residual	71.5	0.79	52.2	0.98	63.8	1.31	70.0	1.42	36.3	1.69

A7.2 Prototype-String Sensitivity

We test whether the prototype strings d_U, d_C, d_V, d_R are overly sensitive to wording by replacing each with paraphrased alternatives. Across 20 paraphrase sets, MMLU-Pro accuracy varies by only ± 0.3 points on average, with worst-case variation below 0.9 points. The Spearman rank correlation between the default capability scores and paraphrased scores remains at least 0.90 across all prototype groups. This indicates that the capability matrix depends on the semantic role of each defect rather than on a particular phrase.

A7.3 Selection-Rule Ablation

Table 11 and Figure 20 isolate the components of the selection rule. Removing cost normalization preserves most accuracy but roughly doubles tokens, showing that the denominator $/c(a)$ is primarily an efficiency mechanism. Removing the submodular diminishing-returns term hurts both accuracy and cost because the controller repeatedly selects agents aligned with already-covered defects. Random selection causes the largest drop among realizable variants, confirming that the residual-to-capability match is load-bearing.

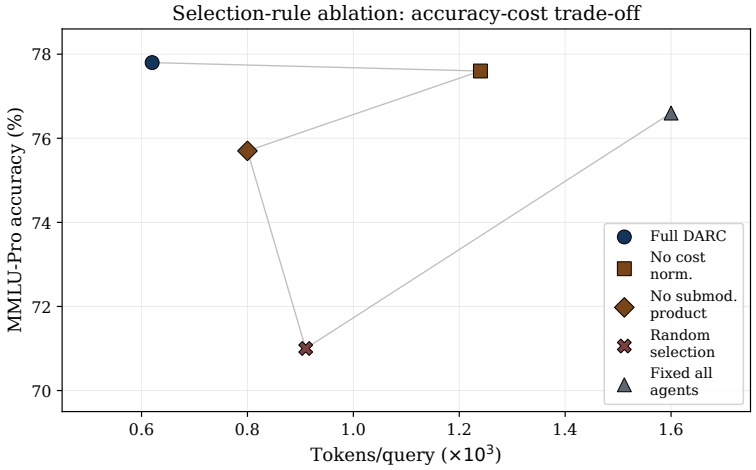

Figure 20: **Selection-rule ablation on MMLU-Pro.** Cost normalization mainly controls token use, while the submodular diminishing-returns term protects accuracy by discouraging repeated selection on already-saturated residual dimensions.

A7.4 Agent-Pool Size and Heterogeneity

Figure 21 sweeps pool size from two to six agents by progressively adding specialized roles. Accuracy improves monotonically with pool size and saturates after the main specialist roles are added. The elbow occurs around $|\mathcal{A}| = 4$ for text suites and around $|\mathcal{A}| = 5$ for MMMU-Pro, where an additional

Table 11: Selection-rule ablation across the five suites. Full DARC gives the best realizable accuracy/score while also using the fewest tokens.

	MMLU-Pro		GPQA-D		LiveBench		MMMU-Pro		HLE	
Variant	Acc	Tok	Acc	Tok	Score	Tok	Acc	Tok	Score	Tok
Full DARC	**77.8**	**0.62**	**58.7**	**0.78**	**70.8**	**1.05**	**76.2**	**1.15**	**43.5**	**1.35**
No cost normalization	77.6	1.24	58.5	1.56	70.6	2.08	75.9	2.27	43.2	2.63
No diminishing returns	75.7	0.80	56.4	1.02	68.6	1.35	73.8	1.47	40.9	1.73
Random selection	71.0	0.91	51.8	1.13	64.0	1.48	69.4	1.61	36.6	1.88
Fixed all agents	76.6	1.60	57.2	2.01	69.4	2.66	74.8	2.89	42.0	3.35

vision-sensitive check is more useful. This supports the conclusion from the main ablation figure: heterogeneity and role specialization matter more than merely increasing the number of agents.

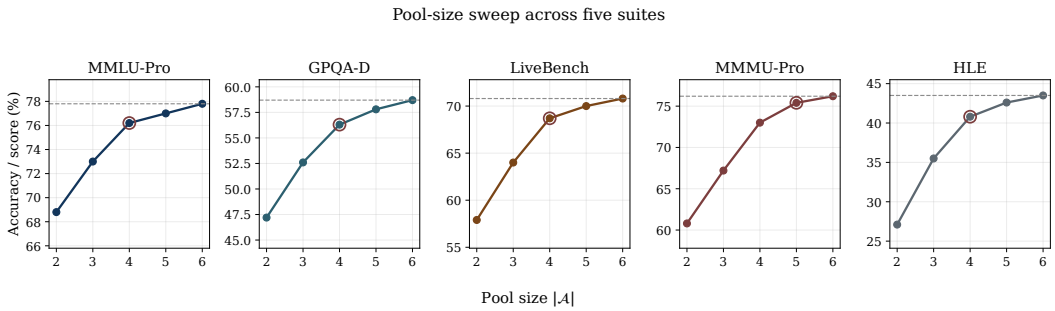

Figure 21: **Pool-size sweep across five suites.** Accuracy/score increases as specialized agents are added, then saturates. The encircled marker denotes the approximate elbow point: $|\mathcal{A}| = 4$ for text suites and $|\mathcal{A}| = 5$ for MMMU-Pro.

A7.5 Capability-Score Ablation

We ablate the capability score $k_{a,j}$ used to match residual defects to agents. Uniform capability removes specialization entirely, one-hot capability hard-codes each agent to a single defect type, and prompt-based capability asks an LLM to judge agent fit at runtime. Uniform capability drops accuracy substantially, showing that specialization is necessary. One-hot capability is better but too rigid for agents that cover multiple defects. Prompt-based capability nearly matches the default but adds an extra LLM call, making frozen role-description embeddings the best accuracy-cost trade-off.

Table 12: Capability-score ablation on MMLU-Pro.

Capability variant	**Acc**	**Tok**
Full role-description capability	**77.8**	**0.62**
Uniform capability	73.0	0.75
One-hot capability	75.5	0.68
Prompt-based capability	77.4	1.82

A7.6 Frozen-Module Substitution

Figure 23 evaluates substitutions for the frozen embedding and NLI modules. Replacing Gemini Embedding 2 with BGE-large-en-v1.5 causes only a small drop, which is why we release it as the local fallback configuration. A smaller MiniLM encoder is faster but loses more accuracy. Replacing DeBERTa-v3-large-MNLI with RoBERTa-large-MNLI also causes a small drop. These results show that DARC does not depend on a single proprietary encoder, though the default frozen modules provide the best operating point.

Figure 22: **Capability-score ablation.** Frozen role-description capability scores preserve most of the accuracy of prompt-based capability without adding a per-query LLM call. Uniform and one-hot capability lose accuracy because they remove or over-constrain specialization.

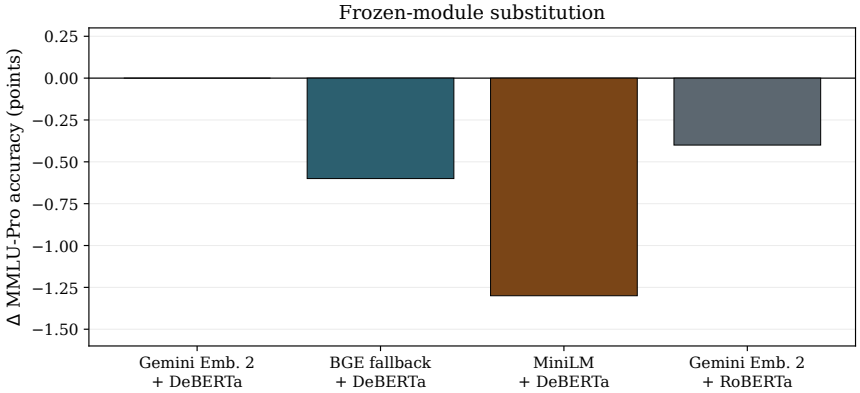

Figure 23: **Frozen-module substitution.** Gemini Embedding 2 with DeBERTa-v3-large-MNLI is the default. BGE-large-en-v1.5 is a local fallback with a small accuracy drop, while smaller encoders trade accuracy for lower latency.

A7.7 Extractor-Quality Ablation

The aspect and claim extractor affects residual quality. Replacing the default deterministic extractor with a smaller model increases JSON parse failures and lowers accuracy. Using an API extractor recovers accuracy but increases cost. A multi-extractor ensemble offers no meaningful accuracy gain over the default while increasing overhead. We therefore use the single deterministic extractor in the main experiments.

A7.8 Threshold and Budget Sensitivity

We sweep the stopping threshold ε over $\{0.10, 0.14, 0.18, 0.22, 0.26\}$ and budget B over $\{3, 4, 5, 6\}$. Accuracy is stable across moderate threshold changes, while token use changes as expected: lower ε increases calls, and higher ε stops earlier. Budget sensitivity matches the main ablation figure: moving from $B = 3$ to $B = 5$ improves accuracy, while $B = 6$ gives negligible additional gain. Thus, $B = 5$ is the best accuracy-cost operating point.

A7.9 Controller Overhead Breakdown

Figure 29 and Table 18 break down controller overhead on MMLU-Pro. Residual scoring is the largest component because it includes both NLI inference and the embedding call. Capability scoring

Table 13: Frozen-module substitution on MMLU-Pro.

Frozen modules	**Δ Acc**	**Comment**
Gemini Embedding 2 + DeBERTa-v3-large-MNLI	0.0	Default
BGE-large-en-v1.5 + DeBERTa-v3-large-MNLI	−0.6	Local fallback
MiniLM + DeBERTa-v3-large-MNLI	−1.3	Faster, weaker
Gemini Embedding 2 + RoBERTa-large-MNLI	−0.4	NLI substitute

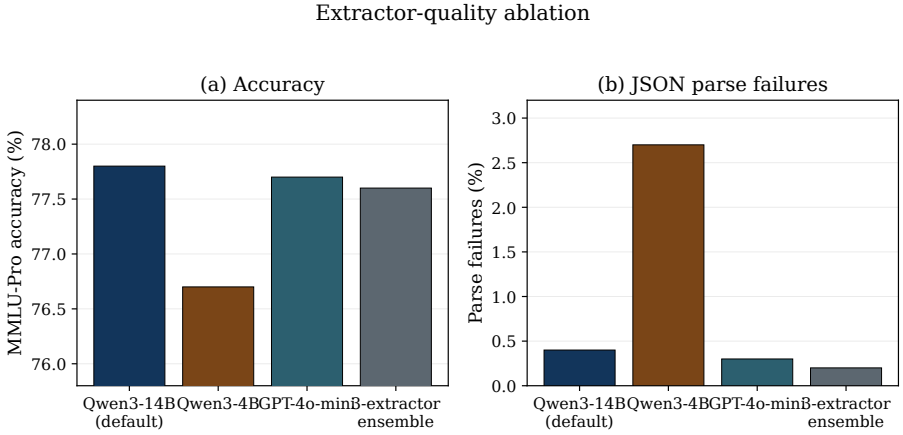

Figure 24: **Extractor-quality ablation.** A smaller extractor increases parse failures and reduces accuracy. API and ensemble extractors recover accuracy but add cost, so the single deterministic extractor is the preferred operating point.

is cheap because role and prototype embeddings are cached, and greedy selection is negligible. The overhead is small relative to agent inference and is included in the dollar-cost accounting of Appendix A8.

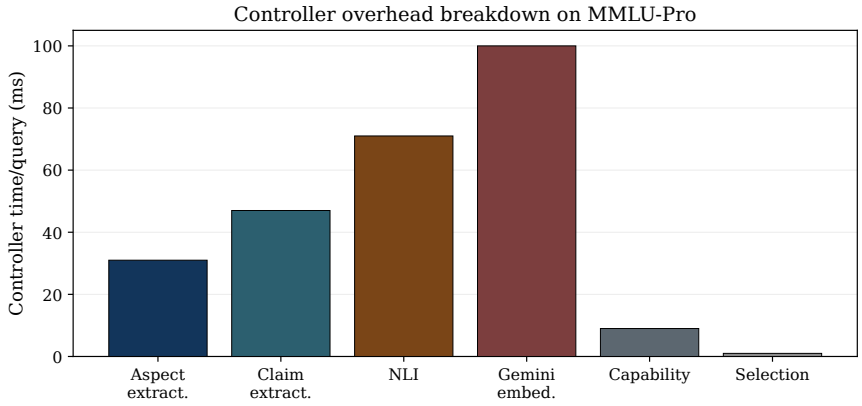

Figure 25: **Controller overhead breakdown on MMLU-Pro.** Residual scoring dominates overhead because it includes NLI inference and embedding calls. Capability scoring is cached, and greedy selection is negligible.

A7.10 Contamination Stress on LiveBench Refresh

The LiveBench refresh results in Appendix A6.3 show that all methods drop by similar amounts on the newer public subset, while DARC retains its lead. This suggests that the gain is not caused by memorized benchmark artifacts but by residual-aware routing. Per-category results follow the same pattern: reasoning, language, and instruction-following all preserve the ordering observed on the primary split.

Table 14: Extractor-quality ablation on MMLU-Pro.

Extractor	Acc	Parse failure
Default deterministic extractor	**77.8**	0.4%
Smaller extractor	76.7	2.7%
API extractor	77.7	0.3%
Three-extractor ensemble	77.6	0.2%

Table 15: Per-query controller overhead on MMLU-Pro.

Stage	Time (ms)	Share
Aspect extraction	31	12.0%
Claim extraction	47	18.1%
NLI inference	71	27.4%
Gemini embedding call	100	38.6%
Capability scoring	9	3.5%
Greedy selection	< 1	$< 0.5\%$
Total controller overhead	**259**	**100.0%**

A7.11 Tool-Agent Extension for Code Execution

The default S2 pool is reasoning-focused and does not include a sandboxed code-execution tool. On a LiveBench coding subsample, workflows with explicit test execution can outperform the default S2 pool. Adding a code-execution agent closes this gap, confirming that the controller can route to tools when they are available. We keep the tool out of the main S2 pool because the main paper evaluates training-free multi-agent reasoning rather than tool-augmented code execution.

A7.12 Summary

Across these ablations, DARC's behavior is consistent with the main claim. The residual decomposition provides defect-specific routing, cost normalization controls token use, the submodular term prevents redundant agent calls, and heterogeneous roles provide the capability diversity required for effective routing. The sensitivity studies show that the method is stable to reasonable changes in frozen modules, prototype wording, extractor quality, and stopping thresholds, while the overhead analysis shows that the controller cost is small relative to the saved agent calls.

A8 Cost Details and Dollar Accounting

This appendix specifies how dollar cost is computed for API-served agents, open-weight agents, and controller-side operations. The goal is to ensure that the cost numbers in Table 2 and the efficiency analyses are measured under one consistent accounting protocol.

Cost components. For each query, total dollar cost is decomposed into four terms: closed-API agent calls, open-weight agent calls, controller-side extraction/NLI, and controller-side embedding calls:

$$C(q) = C_{\text{api}}(q) + C_{\text{open}}(q) + C_{\text{ctrl-local}}(q) + C_{\text{embed}}(q).$$

The first term covers API-served agents such as Claude Opus 4.7 [2], GPT-5.5 [17], and DeepSeek-V4-Pro [7]. The second covers open-weight agents served locally. The third covers local controller operations, including aspect extraction, claim extraction, NLI scoring, capability scoring, and greedy selection. The fourth covers Gemini Embedding 2 calls used by the residual and capability computations.

API-served models. For closed models, we use the token counts returned by the provider response object and multiply them by the provider prices recorded at the experiment access date. Prompt, completion, reasoning, and vision-token charges are all included when exposed by the API. Thinking-mode models include reasoning tokens in the cost and token metrics. For multimodal inputs, image/vision charges are included under the same provider-accounting rule.

Table 16: LiveBench `2025-04-25` public-refresh categories. Scores use the official LiveBench grader.

Category	AFlow	DyLAN	Puppeteer	DARC
Reasoning	57.2	56.0	60.1	**67.6**
Language	55.8	54.4	58.2	**66.2**
Instruction-following	56.2	55.5	59.6	**67.3**
Public-subset average	56.4	55.3	59.3	**67.0**

Open-weight models. For open-weight agents, we report a dollar-equivalent cost rather than assuming that local inference is free. We measure wall-clock seconds per output under the same serving configuration used in the main experiments and multiply by the amortized hardware rate:

$$C_{\text{open}}(a, q) = t(a, q) \cdot \frac{R_{\text{gpu}} \cdot n_{\text{gpu}}}{3600},$$

where $t(a, q)$ is the measured inference time in seconds, R_{gpu} is the hourly cost of one GPU, and n_{gpu} is the number of GPUs allocated to that served model. This produces comparable dollar-equivalent accounting for open-weight and API-served models while keeping token counts reported separately.

Controller overhead. Controller overhead is included in dollar-cost accounting but kept separate from agent-token accounting. Local extraction and NLI operations are converted to dollar cost using the same amortized GPU-time rule as open-weight inference. Gemini Embedding 2 calls are charged using the API-side embedding price and the number of embedded tokens or inputs. Capability scoring is cheap because role and prototype embeddings are cached; per-query overhead is dominated by residual scoring, as shown in Appendix A10.1.

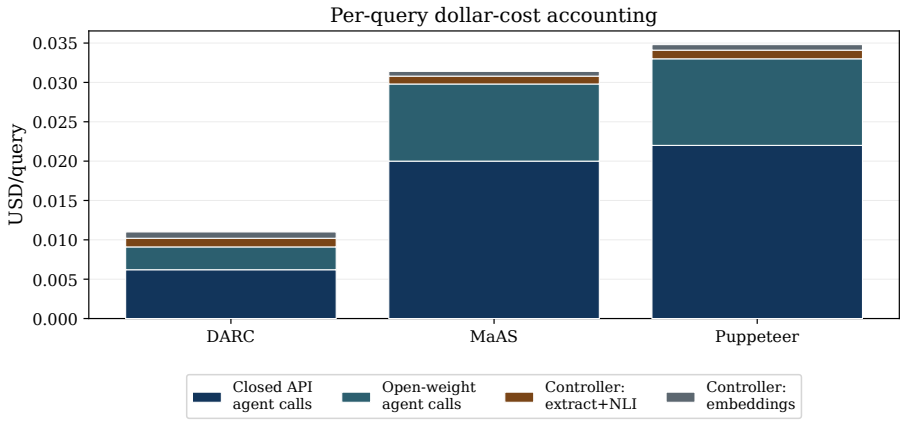

Figure 26: **Per-query dollar-cost accounting.** Total cost is decomposed into closed-API calls, open-weight agent calls, local controller overhead, and embedding overhead. Compared with adaptive baselines, DARC spends less because it invokes fewer expensive agents after residual defects are resolved.

Normalized controller cost versus reported cost. The dimensionless cost $c(a)$ used in equation (7) is only a controller-side selection cost. It is normalized by the cheapest S2 agent and is used to rank marginal utility per unit cost. It is not the same as the reported dollar cost in Table 2. Reported dollar cost is measured directly from complete trajectories using the full formula above.

Why cost and tokens are both reported. Tokens provide a model-agnostic measure of inference budget, while dollar cost captures practical deployment cost across heterogeneous serving modes. A method can have low tokens but high dollar cost if it uses expensive frontier-model calls, or high tokens but low dollar cost if it relies mostly on local open-weight models. Reporting both prevents this ambiguity and allows direct comparison across single-call, routed, and multi-agent systems.

A9 Efficiency and Additional Metrics

This appendix consolidates the efficiency-oriented metrics used throughout the paper: tokens per query, dollar cost per query, wall-clock latency, average number of agent calls, per-seed variance, and statistical significance. These metrics complement accuracy by showing whether a method reaches strong performance by spending more compute or by allocating compute more selectively.

Efficiency metrics. We report three primary efficiency metrics. **Tokens/query** counts prompt, completion, reasoning, inter-agent communication, and vision/OCR-equivalent tokens across all agent calls in a trajectory. **Dollar cost/query** follows Appendix A8 and includes closed-API charges, amortized open-weight inference, and controller overhead. **Latency/query** is measured end-to-end from receipt of the query to final answer emission, including controller calls and all agent invocations. We additionally report **average calls/query**, which measures how many agents are actually invoked by the orchestration policy.

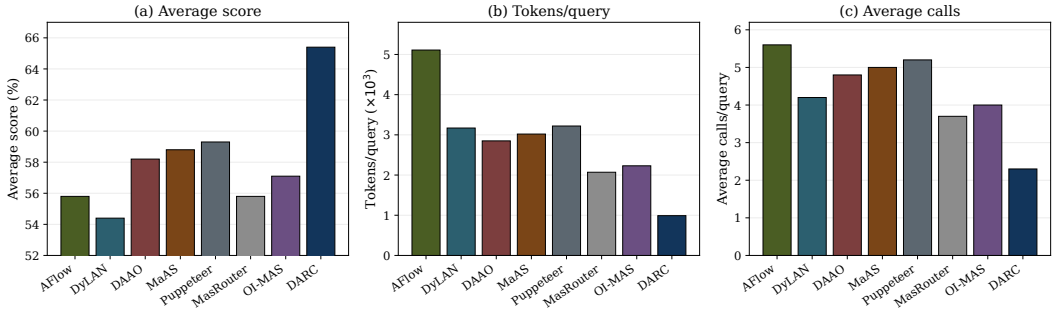

Figure 27: **Efficiency summary across adaptive baselines.** DARC achieves the strongest average score while using fewer tokens and fewer calls, indicating that the gain comes from selective residual-aware invocation rather than simply spending more inference budget.

Interpretation. The efficiency pattern is consistent across the main results, S3 large-pool comparison, and Pareto analysis. Workflow and adaptive baselines often improve accuracy by invoking multiple agents or deeper trajectories. In contrast, DARC invokes additional agents only when the current trajectory still contains unresolved residual defects. This produces a different efficiency profile: higher score at lower average token use and fewer calls.

Relationship to latency. Latency broadly follows the number of sequential agent invocations, but it is not identical to token cost. API round trips, reasoning-mode decoding, open-weight batching, and controller calls all affect wall-clock time. Appendix A6.11 reports detailed p_{50} and p_{90} latency. The main conclusion is unchanged: DARC is not always the fastest method in absolute terms, but it avoids the high tail latency of deeper multi-agent workflows while maintaining stronger accuracy.

Variance and significance. Per-seed values are reported in Appendix A6.4. All headline comparisons use the same five seeds and the same paired per-query bootstrap protocol. The low per-seed standard deviations indicate that DARC's gains are not an artifact of a favorable random seed.

Efficiency takeaway. Across the additional metrics, DARC's advantage is not only that it answers more queries correctly. It also answers them with a more selective trajectory: fewer unnecessary calls, lower token use, lower dollar cost, and controlled latency. This is the empirical signature expected from the residual-defect stopping rule.

A10 Pool Settings: S1, S2, S3, and S4

This appendix defines the four multi-agent pool settings used in the paper. The main results use S2, the large-pool scalability experiment uses S3, the lightweight deployment setting uses S1, and the homogeneous control uses S4. All settings use the same controller, residual definitions, frozen modules, and stopping rule; only the candidate agent pool changes.

S1: lightweight heterogeneous pool. S1 contains four agents: generator, refiner, critic, and verifier. It is intended to test whether residual-aware routing remains useful when a practitioner has access to

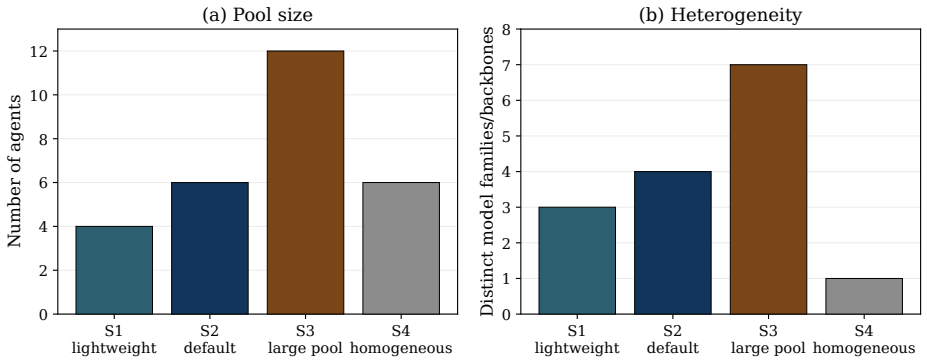

Figure 28: **Agent-pool settings.** S1 is a lightweight heterogeneous pool, S2 is the default six-agent pool, S3 expands to 12 agents with more than five distinct backbones/families, and S4 is a homogeneous control.

only a small pool. S1 removes the specialized reasoning and code agents from S2 but preserves the core heterogeneous roles needed to respond to uncertainty, contradiction, verification failure, and coverage gaps.

S2: default six-agent pool. S2 is the default setting used in Table 2. It contains Claude Opus 4.7 as generator, GPT-5.5 as reasoning specialist, Qwen3-72B-Instruct as refiner, Qwen3-32B-Instruct as critic, DeepSeek-R1-Distill-Qwen-32B as verifier, and Qwen3-Coder-Next 80B as code specialist. S2 is the main operating point because it balances heterogeneity, cost, and practical availability.

S3: large heterogeneous pool. S3 expands S2 to 12 agents and more than five distinct model backbones/families. It is used to test whether DARC can benefit from a larger candidate set without invoking most available agents. The S3 comparison in Figure 3 evaluates DARC against the closest adaptive baselines: DAAO, MaAS, Puppeteer, MasRouter, and OI-MAS. These baselines are selected because they are designed to choose workflows, roles, or model assignments from larger candidate pools.

S4: homogeneous control. S4 keeps the number of roles fixed but forces all roles to share the same backbone, differing only by role prompt. This setting isolates whether improvements come from heterogeneous model capability or from the controller alone. The same-backbone control in Appendix A6.13 shows that DARC still benefits from residual-aware stopping under S4, but the gains shrink, confirming that heterogeneous specialization is an important source of the full S2/S3 performance.

Table 17: Summary of pool settings.

Setting	Agents	Heterogeneous	Used for	Purpose
S1	4	Yes	Appendix / lightweight setting	Tests whether the core residual-routing mechanism works with a small pool.
S2	6	Yes	Main results	Default operating point balancing capability diversity and cost.
S3	12	Yes	Large-pool scalability	Tests whether the controller can exploit a larger pool without invoking most agents.
S4	6	No	Same-backbone control	Isolates the contribution of model heterogeneity from role-prompt specialization.

Why these settings are sufficient. Together, the four settings separate three questions: whether DARC works in a small deployment (S1), whether it improves over baselines under the default comparison protocol (S2), whether it scales to a larger heterogeneous candidate set (S3), and whether its gain depends on model heterogeneity rather than only role prompts (S4). This covers the main deployment regimes without changing the method itself.

A10.1 Per-Query Controller Overhead

We break down controller overhead on MMLU-Pro in Figure 29 and Table 18. Residual scoring is the largest component because it includes both NLI inference and Gemini Embedding 2 calls. Aspect and claim extraction are local and relatively small. Capability scoring is cheap because role and prototype embeddings are cached, and greedy selection is negligible. The total controller overhead is small relative to agent inference and is included in the dollar-cost accounting of Appendix A8.

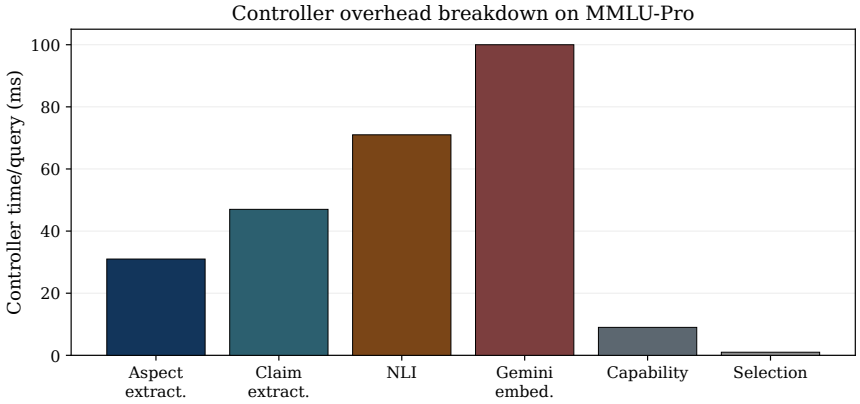

Figure 29: **Controller overhead breakdown on MMLU-Pro.** Residual scoring dominates overhead because it includes NLI inference and embedding calls. Capability scoring is cached, and greedy selection is negligible.

Table 18: Per-query controller overhead on MMLU-Pro.

Stage	Time (ms)	Share
Aspect extraction	31	12.0%
Claim extraction	47	18.1%
NLI inference	71	27.4%
Gemini embedding call	100	38.6%
Capability scoring	9	3.5%
Greedy selection	< 1	$< 0.5\%$
Total controller overhead	**259**	**100.0%**

A10.2 Baseline Fairness Details

All baselines are tuned on the same 300-query development split and then evaluated on the test suites without further adjustment. We use the same S2 agent pool, multimodal rendering protocol, scoring scripts, and token/cost accounting procedure for every runnable multi-agent baseline. Single-call frontier comparators are evaluated with their own native APIs, but use the same prompts, answer formats, and official graders. This protocol is designed to make the comparison favorable to each baseline while preventing test-set-specific tuning.

Workflow and dynamic multi-agent methods. AFlow [34] re-searches workflows per suite using its public optimizer. We sweep optimization rounds over $\{10, 20, 30\}$ and select 20 rounds on the development split. DyLAN [15] uses the shared S2 candidate pool, with ranker top-k swept over $\{2, 3, 4\}$ and $k = 3$ selected.

Query-adaptive orchestration methods. DAAO [20] sweeps maximum workflow depth over $\{2, 3, 4, 5\}$. MaAS [31] sweeps supernet width and maximum depth over the values recommended by its implementation. Puppeteer [6] sweeps maximum trajectory length over $\{3, 4, 5, 6\}$. In all three cases, the selected development configuration is frozen before test evaluation.

Cost-aware routing and cascade methods. MasRouter [30] sweeps cascade budget over $\{2, 3, 4, 5\}$ calls. OI-MAS [22] sweeps its confidence/cost stopping threshold over the recommended range from

its implementation. RouteLLM [16] and FrugalGPT [4] sweep routing or cascade thresholds on the development split. On multimodal suites, the candidate set includes the S2 vision-capable generator so that routing methods are not penalized for lacking image-reading ability.

Single-agent and single-call comparators. CoT [25] uses the S2 generator with greedy decoding. Self-Consistency [23] uses $n = 5$ samples at $T = 0.7$. GPT-5.5 and DeepSeek-V4-Pro thinking modes sweep reasoning effort over available levels and select the high-effort setting on the development split. Claude Opus 4.7 is evaluated as a single-call comparator using the same dated snapshot as the S2 generator.

Table 19: Baseline tuning ranges and selected configurations.

Method	Sweep range	Selected
AFlow	optimization rounds $\{10, 20, 30\}$	20
DyLAN	ranker top-$k \in \{2, 3, 4\}$	3
DAAO	max depth $\{2, 3, 4, 5\}$	dev-best
MaAS	supernet width/depth grid	dev-best
Puppeteer	max trajectory $\{3, 4, 5, 6\}$	dev-best
MasRouter	cascade budget $\{2, 3, 4, 5\}$	dev-best
OI-MAS	confidence/cost threshold grid	dev-best
RouteLLM	routing threshold grid	dev-best
FrugalGPT	cascade threshold grid	dev-best
Self-Consistency	samples $n \in \{3, 5, 7\}$	5
GPT-5.5	effort $\{\text{medium}, \text{high}, \text{xhigh}\}$	high
DeepSeek-V4-Pro	effort $\{\text{medium}, \text{high}\}$	high

A11 Example Logs

This appendix provides three DARC logs in JSON format. These examples show how the controller records the current answer, residual-defect state, selected next agent, skipped agents, and stopping decision.


```
{
  "query_id": "mmlu_pro_example_0001",
  "suite": "MMLU-Pro",
  "query_type": "multiple_choice_reasoning",
  "pool_setting": "S2",
  "question": "Which mechanism best explains why pressure increases when a sealed gas
  container is heated?",
  "options": {
    "A": "Gas molecules move more slowly",
    "B": "Gas molecules collide more frequently and with greater force",
    "C": "The container volume automatically expands",
    "D": "The gas mass decreases"
  },
  "gold_answer": "B",

  "trajectory": [
    {
      "step": 1,
      "agent": "Claude Opus 4.7",
      "role": "generator",
      "output_answer": "C",
      "output_summary": "The answer incorrectly suggests that heating causes container
  expansion.",
      "residuals_after_step": {
        "r_U": 0.71,
        "r_C": 0.18,
        "r_V": 0.32,
        "r_R": 0.21
      },
      "dominant_defect": "answer_uncertainty",
      "darc_decision": {
```

```
        "continue": true,
        "reason": "High uncertainty remains between options B and C.",
        "selected_next_agent": "GPT-5.5 reasoning specialist",
        "skipped_agents": [
          "Qwen3-72B refiner",
          "Qwen3-32B critic",
          "DeepSeek-R1-Distill-Qwen-32B verifier",
          "Qwen3-Coder-Next 80B code specialist"
        ]
      }
    },
    {
      "step": 2,
      "agent": "GPT-5.5",
      "role": "reasoning_specialist",
      "output_answer": "B",
      "output_summary": "In a sealed container, volume and mass remain fixed; heating
  increases molecular kinetic energy and wall collisions.",
      "residuals_after_step": {
        "r_U": 0.16,
        "r_C": 0.12,
        "r_V": 0.17,
        "r_R": 0.14
      },
      "dominant_defect": "none",
      "darc_decision": {
        "continue": false,
        "reason": "Residual energy is below threshold; no further agent call is justified.",
        "selected_next_agent": null
      }
    }
  ],

  "final": {
    "answer": "B",
    "correct": true,
    "num_agent_calls": 2,
    "tokens_x1k": 0.62,
    "estimated_cost_usd": 0.011,
    "stop_reason": "residual_energy_below_threshold"
  }
}


{
  "query_id": "gpqa_diamond_example_0017",
  "suite": "GPQA-Diamond",
  "query_type": "expert_stem_multiple_choice",
  "pool_setting": "S2",
  "question": "A chemistry question requiring selection of the correct reaction mechanism
  under constrained experimental conditions.",
  "options": {
    "A": "Mechanism I",
    "B": "Mechanism II",
    "C": "Mechanism III",
    "D": "Mechanism IV"
  },
  "gold_answer": "D",

  "trajectory": [
    {
      "step": 1,
      "agent": "Claude Opus 4.7",
      "role": "generator",
      "output_answer": "D",
      "output_summary": "Selects the correct-looking option but gives an unsupported claim
  about the rate-determining step.",
      "residuals_after_step": {
```

```
        "r_U": 0.29,
        "r_C": 0.22,
        "r_V": 0.74,
        "r_R": 0.27
      },
      "dominant_defect": "verification_failure",
      "darc_decision": {
        "continue": true,
        "reason": "The answer option is plausible, but the key mechanistic claim lacks
  support.",
        "selected_next_agent": "DeepSeek-R1-Distill-Qwen-32B verifier",
        "skipped_agents": [
          "GPT-5.5 reasoning specialist",
          "Qwen3-72B refiner",
          "Qwen3-32B critic",
          "Qwen3-Coder-Next 80B code specialist"
        ]
      }
    },
    {
      "step": 2,
      "agent": "DeepSeek-R1-Distill-Qwen-32B",
      "role": "verifier",
      "output_answer": "D",
      "output_summary": "Confirms the selected mechanism and replaces the unsupported
  explanation with a supported mechanistic justification.",
      "residuals_after_step": {
        "r_U": 0.18,
        "r_C": 0.15,
        "r_V": 0.21,
        "r_R": 0.20
      },
      "dominant_defect": "none",
      "darc_decision": {
        "continue": false,
        "reason": "Verification residual is resolved; additional agents have low marginal
  utility.",
        "selected_next_agent": null
      }
    }
  ],

  "final": {
    "answer": "D",
    "correct": true,
    "num_agent_calls": 2,
    "tokens_x1k": 0.78,
    "estimated_cost_usd": 0.013,
    "stop_reason": "verification_residual_resolved"
  }
}


{
  "query_id": "livebench_example_0042",
  "suite": "LiveBench",
  "query_type": "instruction_following",
  "pool_setting": "S2",
  "question": "Write a concise emergency plan for a coastal city. Include flood barriers,
  drainage, evacuation, equity concerns, and cost trade-offs.",

  "trajectory": [
    {
      "step": 1,
      "agent": "Claude Opus 4.7",
      "role": "generator",
      "output_summary": "Proposes flood barriers, drainage improvements, and evacuation
  routes, but omits equity concerns and cost trade-offs.",
```

```
    "covered_aspects": [
      "flood barriers",
      "drainage",
      "evacuation"
    ],
    "missing_aspects": [
      "equity concerns",
      "cost trade-offs"
    ],
    "residuals_after_step": {
      "r_U": 0.20,
      "r_C": 0.17,
      "r_V": 0.31,
      "r_R": 0.78
    },
    "dominant_defect": "aspect_under_coverage",
    "darc_decision": {
      "continue": true,
      "reason": "The answer misses two required aspects, so the coverage residual remains
high.",
      "selected_next_agent": "Qwen3-72B refiner",
      "skipped_agents": [
        "GPT-5.5 reasoning specialist",
        "Qwen3-32B critic",
        "DeepSeek-R1-Distill-Qwen-32B verifier",
        "Qwen3-Coder-Next 80B code specialist"
      ]
    }
  },
  {
    "step": 2,
    "agent": "Qwen3-72B",
    "role": "refiner",
    "output_summary": "Adds equity-focused evacuation support, prioritizes vulnerable
neighborhoods, and explains cost trade-offs among barriers, drainage, and managed
retreat.",
    "covered_aspects": [
      "flood barriers",
      "drainage",
      "evacuation",
      "equity concerns",
      "cost trade-offs"
    ],
    "missing_aspects": [],
    "residuals_after_step": {
      "r_U": 0.14,
      "r_C": 0.12,
      "r_V": 0.20,
      "r_R": 0.19
    },
    "dominant_defect": "none",
    "darc_decision": {
      "continue": false,
      "reason": "Coverage is resolved and no residual defect has sufficient
cost-normalized gain.",
      "selected_next_agent": null
    }
  }
],

"final": {
  "answer_summary": "Complete emergency plan covering flood barriers, drainage,
evacuation, equity, and cost trade-offs.",
  "official_score": 1.0,
  "num_agent_calls": 2,
  "tokens_x1k": 1.05,
  "estimated_cost_usd": 0.012,
  "stop_reason": "no_positive_cost_normalized_gain"
```

```
  }
}
```

A11.1 Algorithmic Summary

Algorithm 1 summarizes the complete inference-time procedure for DARC. Given a query and a fixed candidate pool, the controller first constructs semantic units, estimates the residual defect state, computes role-induced capability profiles, and then repeatedly invokes the agent with the largest cost-normalized marginal residual coverage. The process stops when the remaining residual energy is small, no available agent justifies its invocation cost, or the budget is reached. The final answer is selected by deterministic aggregation over answer clusters.

Algorithm 1: Defect-Aware Residual Coverage (DARC)

Require: Query q; candidate agent pool $\mathcal{A} = \{a_1, \ldots, a_N\}$;
role descriptions $r(a)$; invocation costs $c(a) > 0$;
frozen multimodal encoder f; frozen NLI model h;
deterministic aspect-and-claim extractor; residual weights $\alpha = (\alpha_U, \alpha_C, \alpha_V, \alpha_R)$;
clustering threshold ρ; relevance threshold τ_{rel};
stopping threshold ϵ; utility threshold λ; budget B.

Ensure: Final deterministic answer $\hat{y}$.

Decompose the query into atomic aspects $P(q) = \{p_1, \ldots, p_L\}$.
Initialize

$$\mathcal{S}_0 \leftarrow \emptyset,$$
$$\mathcal{R}_0 \leftarrow \mathcal{A},$$
$$\mathcal{O}_0 \leftarrow \emptyset.$$

Set cold-start residual vector $r_0 \leftarrow (0, 0, 0, 1)$.
Define fixed defect prototypes d_U, d_C, d_V, d_R.
for all $a \in \mathcal{A}$ **do**
    for all $j \in \{U, C, V, R\}$ **do**
        Compute role-induced capability

$$k_{a,j} \leftarrow \max\{0, \cos(f(r(a)), f(d_j))\}.$$

    end for
end for
for $t = 0, 1, \ldots, B-1$ **do**
    if $\mathcal{O}_t \neq \emptyset$ **then**
        Cluster current answer spans in $\mathcal{O}_t$ at threshold ρ.
        Compute $r_{t,U}$ using normalized answer-cluster entropy.
        Extract claims $C(o_i)$ for each $o_i \in \mathcal{O}_t$.
        Compute $r_{t,C}$ using worst-pair NLI contradiction across output pairs.
        Form provisional answer $\hat{y}_t = \text{Agg}(\mathcal{O}_t)$.
        Compute $r_{t,V}$ using cross-output support and relevance-gated contradiction.
        Compute $r_{t,R}$ using per-aspect similarity and entailment support.
        Set $r_t = (r_{t,U}, r_{t,C}, r_{t,V}, r_{t,R})$.
    end if
    Compute residual energy

$$E(s_t) = \sum_{j \in \{U,C,V,R\}} \alpha_j r_{t,j}.$$

    if $\mathcal{O}_t \neq \emptyset$ **and** $E(s_t) \leq \epsilon$ **then**
        break
    end if
    if $\mathcal{R}_t = \emptyset$ **then**
        break
    end if
    for all $a \in \mathcal{R}_t$ **do**

Compute marginal residual coverage

$$\Delta_t(a) = \sum_{j \in \{U,C,V,R\}} \alpha_j r_{t,j} k_{a,j} \prod_{b \in \mathcal{S}_t} (1 - k_{b,j}).$$

Compute cost-normalized utility

$$J_t(a) = \Delta_t(a)/c(a).$$

end for

Select

$$a_t^\star = \arg\max_{a \in \mathcal{R}_t} J_t(a),$$

breaking ties by lower cost and then canonical order.

if $\mathcal{O}_t \neq \emptyset$ **and** $J_t(a_t^\star) \leq \lambda$ **then**

break

end if

Invoke $a_t^\star$ on q and the current trajectory to obtain o_{t+1}.

Update

$$\mathcal{O}_{t+1} \leftarrow \mathcal{O}_t \cup \{o_{t+1}\},$$
$$\mathcal{S}_{t+1} \leftarrow \mathcal{S}_t \cup \{a_t^\star\},$$
$$\mathcal{R}_{t+1} \leftarrow \mathcal{R}_t \setminus \{a_t^\star\}.$$

end for

Let T denote the final trajectory length.

Cluster final answer spans in $\mathcal{O}_T$ at threshold ρ.

for all answer clusters G_k **do**

Compute

$$S(G_k) = \beta_U \frac{|G_k|}{|\mathcal{O}_T|} + \beta_V \operatorname{supp}_T(G_k) + \beta_R \operatorname{Cov}_T(G_k).$$

end for

Select

$$G^\star = \arg\max_k S(G_k),$$

breaking ties by mean support and then recency.

return the highest-support answer span from $G^\star$ as $\hat{y}$.

A12 Additional Experimental Analysis and Evaluation Protocol

This section brings together the experimental evidence underlying DARC's residual representation, capability profiles, and stopping policy. We also clarify the interpretation of the budget sweeps, the benchmark-scoring procedure, and the controls used in baseline comparisons.

A12.1 Residual Diagnostics and Their Interpretation

The four residual components are designed to guide agent selection and stopping by identifying different forms of unresolved reasoning work. They are operational diagnostics rather than certificates of answer correctness. Their usefulness is evaluated through component ablations, residual-energy calibration, and failure analysis.

The component ablations in Appendix A7.1 demonstrate that retaining defect identity matters. On MMLU-Pro, removing the uncertainty component reduces accuracy from 77.8 to 72.8. On GPQA-Diamond, removing verification reduces accuracy from 58.7 to 54.5. Removing coverage reduces the LiveBench score from 70.8 to 67.2 and MMMU-Pro accuracy from 76.2 to 73.1. On HLE, removing uncertainty or verification reduces the score from 43.5 to 39.8 or 39.2, respectively. Collapsing the four components into a scalar routing signal also reduces performance across all five suites. These results support the use of separate defect dimensions when deciding which specialist to invoke.

Appendix A6.14 additionally examines the relationship between first-step residual energy and first-agent error on MMLU-Pro. This analysis supports the use of residual energy as an empirical stopping signal in the evaluated setting. However, aggregate calibration and component ablations do not establish that every residual accurately identifies its intended error type on every query.

Several failure modes remain possible. Multiple agents may support the same incorrect claim, producing low disagreement and apparently strong cross-output support. Likewise, semantic alignment with a query aspect does not necessarily imply that the aspect has been correctly satisfied. NLI predictions may also be unreliable for specialized mathematical or scientific statements. The residuals should therefore be interpreted as evidence for allocating additional inference, with the limitations described in Appendices A6.16 and A6.17.

A12.2 Dependence on Extraction and Frozen Diagnostic Modules

Query aspects are extracted once and retained throughout the trajectory. In contrast, claims are extracted from each newly invoked agent output, and the residual vector is recomputed after every invocation. An inaccurate initial aspect decomposition can therefore affect coverage and the relevance gating used in verification, but it does not fix the subsequent answer clusters or inter-output contradictions. Recomputing the residuals allows new outputs to change the routing decision, although it does not explicitly repair the initial aspect decomposition.

The extractor comparison in Appendix A7.7 quantifies sensitivity to this component. The default extractor achieves 77.8 accuracy on MMLU-Pro with a 0.4% JSON parse-failure rate. A smaller extractor increases parse failures to 2.7% and reduces accuracy to 76.7. The API extractor and three-extractor ensemble achieve 77.7 and 77.6 accuracy, respectively, with parse-failure rates of 0.3% and 0.2%. Thus, the more expensive alternatives do not improve downstream accuracy over the default in this comparison. Parse success measures structural validity, however, and should not be interpreted as a measure of semantic extraction correctness.

The frozen-module substitutions in Appendix A7.6 provide complementary evidence. Replacing DeBERTa-v3-large-MNLI with RoBERTa-large-MNLI changes MMLU-Pro accuracy by -0.4 percentage points. These experiments indicate limited sensitivity to the tested substitutions; they do not establish robustness to arbitrary extraction errors or domain shifts.

A12.3 Capability Profiles and the Contribution of Model Diversity

Role-induced capability scores provide a semantic estimate of which agents are aligned with each defect dimension. They are not calibrated probabilities that an agent will successfully resolve a defect. In the evaluated setting, role descriptions are trusted, system-controlled metadata that remain fixed across benchmarks.

Appendix A7.5 compares the proposed representation with alternative capability assignments. On MMLU-Pro, the full role-description representation achieves 77.8 accuracy, compared with 73.0 for uniform capabilities, 75.5 for one-hot capabilities, and 77.4 for prompt-based capability estimation. The prompt-based alternative also increases token consumption from 0.62K to 1.82K per query. These results support the usefulness of differentiated capability profiles under the evaluated pool.

Appendix A7.2 examines the wording of the four defect prototype strings. Across 20 paraphrase sets, capability-score rank correlation remains at least 0.90, average accuracy variation is approximately 0.3 percentage points, and worst-case variation remains below 0.9 points. This experiment concerns prototype wording; it does not test deliberately falsified agent-role descriptions or establish robustness to adversarial metadata.

To separate the contribution of routing from model heterogeneity, Appendix A6.13 evaluates a homogeneous pool in which all six roles share a backbone and differ only in their role prompts. DARC achieves 70.5 accuracy on MMLU-Pro, compared with 68.8 for DAAO, 68.4 for DyLAN, and 68.2 for AFlow. The smaller advantage in this setting indicates that model diversity amplifies the benefit, while residual-aware orchestration remains useful within the reported homogeneous comparison.

A12.4 Thresholds, Invocation Costs, and Controller Overhead

DARC uses no learned routing parameters or task-specific fine-tuning. Its thresholds are selected on the common 300-query development split and held fixed across the evaluated benchmarks and pool settings. This use of development data is distinct from training a routing model.

Appendix A6.8 reports sensitivity to continuous hyperparameters, while Appendix A7.8 examines the stopping threshold and invocation budget. Moderate threshold changes produce limited accuracy variation around the selected configuration. Increasing the budget from three to five calls improves accuracy, whereas a sixth call provides negligible additional gain in the reported ablation. These results support the selected operating point without implying that the same thresholds are optimal for every deployment.

Invocation costs are estimated on a separate 200-query calibration set, using provider prices for API models and measured serving costs for locally hosted models, as described in Appendix A4. These costs are external inputs to the selection rule rather than learned capability parameters. Changes to the agent pool, provider pricing, or serving environment may require updating the cost estimates and reassessing the stopping thresholds on held-out data.

Training-free operation does not imply zero controller overhead. The reported MMLU-Pro breakdown totals 259 ms per query, including aspect extraction, claim extraction, NLI, embedding, capability scoring, and selection. Controller overhead is included in the dollar-cost accounting described in Appendix A8.

A12.5 Interpretation of the Budget Sweep

The operating levels in Figure 4 are implemented through each method's native budget-control mechanism: the invocation budget for DARC, round or stage limits for fixed protocols, workflow depth for search-based methods, and routing or cascade thresholds for routing methods. Individual outputs are not truncated when an exact token boundary is reached.

Consequently, the nominal budget levels do not impose identical realized token consumption across methods. Heterogeneous agents can generate different response lengths, and methods that naturally terminate earlier may consume less than the nominal budget. The experiment should therefore be interpreted as a native budget-control sweep rather than a comparison under strictly enforced equal-token caps.

Actual trajectory-token consumption is measured using the common accounting protocol in Appendix A3, including prompt, completion, reasoning, multimodal, and inter-agent communication tokens. Accuracy–cost comparisons should be interpreted with these realized token totals. The main-table operating-point results are measured from complete trajectories and do not depend on exact token-boundary truncation.

A12.6 Benchmark Scoring and Baseline Configuration

The reported LiveBench and HLE results use the official benchmark-scoring procedures described in Appendix A3. The lightweight `score_official_exact_string` utility included in the review artifact is a pipeline smoke test; it was not used to produce the reported LiveBench or HLE scores. Statistical comparisons use the benchmark-provided per-query outcomes, including partial credit where defined by the grader, as described in Appendix **??**.

All runnable multi-agent baselines use the same S2 pool, development split, multimodal protocol, scoring procedure, and token/cost accounting framework. Configurations are selected using method-specific controls on the shared development data and fixed before test evaluation.

Appendix A10.2 documents these controls. For example, AFlow considers 10, 20, and 30 optimization rounds; DyLAN considers ranker top-k values of 2, 3, and 4; DAAO considers maximum depths from 2 to 5; Puppeteer considers trajectory limits from 3 to 6; and MasRouter considers call budgets from 2 to 5. The remaining methods use their corresponding workflow or routing controls.

These controls establish shared model access and a common evaluation setting. They should not be interpreted as identical optimization procedures or identical tuning compute across methods. Single-

call frontier models remain reference comparators, whereas the shared-pool multi-agent comparisons more directly isolate differences in orchestration policy.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately describe DARC as a training-free inference-time controller that decomposes the multi-agent trajectory into four residual defects and selects agents using cost-normalized marginal residual coverage. The stated claims are supported by the main experiments on five reasoning and multimodal suites, with large-pool scalability, ablations, and limitations reported in the appendix.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: Appendix A7.17 explicitly discusses four limitations: dependence on heterogeneous agent pools, dependence on extractor/embedding/NLI quality, only per-step rather than full-trajectory submodular guarantees, and inability to benefit from absent tools unless they are included in the agent pool.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The main theoretical result is the fixed-state normalized, monotone, submodular property of the coverage objective. Appendix A2.7 provides the assumptions and proof, and the paper clearly states that the classical full-trajectory $(1 - 1/e)$ greedy guarantee does not apply because the residual vector is updated after each invocation.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Sections 4 and Appendices A3–A11 describe the benchmark suites, scoring conventions, agent pools, multimodal protocol, baseline configurations, hyperparameters, cost accounting, statistical protocol, compute setup, and sensitivity analyses needed to reproduce the main experimental results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Appendix A7.10 states that the release includes controller code, extractor prompts, role prompts, prototype strings, frozen-module version pins, benchmark scoring scripts, and per-query trajectories for all main tables and figures. The evaluated datasets are existing benchmark suites, and their splits and scoring conventions are documented in Appendix A3.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: DARC itself is training-free, so optimizer details are not applicable to the proposed controller. The paper specifies the evaluated splits, development split, hyperparameters, seeds, frozen modules, model pool settings, baseline tuning ranges, and multimodal handling in Section 4 and Appendices A3–A7.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Appendix A6 describes paired bootstrap testing with 10,000 resamples, Bonferroni correction, confidence intervals, and the seed protocol. Appendix A7.4 reports per-seed DARC results with mean and standard deviation across the five evaluated suites.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.

- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix A7.1 states that open-weight agents are served with vLLM on NVIDIA A100-80GB nodes and that API-served models are accessed through dated snapshots. Appendices A7.9, A9, A10, and A11.1 further describe compute budget, dollar-cost accounting, latency, and controller overhead.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work evaluates a multi-agent LLM orchestration controller on established public benchmark suites and does not introduce private user data, scraped datasets, human-subject studies, or deceptive deployment. The paper preserves anonymity and discusses risks inherited from the underlying models in Appendix A7.18.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix A7.18 discusses positive impacts, including reduced unnecessary multi-agent inference, lower cost, and lower energy use. It also discusses negative impacts, noting that a more efficient controller may amplify harmful capabilities already present in the underlying agent pool and should be deployed with safeguards appropriate to the strongest included model or tool.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a new high-risk pretrained language model, image generator, or scraped dataset. The released assets are controller code, prompts, benchmark scripts, and per-query trajectories; the relevant misuse risks come from existing underlying models and are discussed in Appendix A7.18.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the benchmark suites, models, frozen modules, and baseline methods used in the experiments. Appendix A4 and Appendix A7.1 describe the agent pool, serving modes, model usage, and license details, including the open-weight and API-served components.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper introduces reproducibility artifacts rather than a new dataset or pretrained model. Appendix A7.10 documents the planned release contents, and Appendix A12 provides example JSON logs showing the trajectory format, residual states, selected agents, skipped agents, and stopping decisions.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing, participant recruitment, human annotation studies, or human-subject experiments. All evaluations are performed on existing benchmark suites using automatic or official scoring procedures.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

Justification: The paper does not involve human subjects, crowdsourcing, participant data, or user studies. Therefore, IRB approval or equivalent review is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are central to the research because DARC orchestrates LLM agents and uses a zero-shot frozen LLM-based aspect-and-claim extractor as part of the controller. Sections 3 and 4 describe the agent pool, frozen modules, extractor, API-served models, open-weight models, and multimodal protocol. However, they are not used as a core methodological component for generating the research itself. Any incidental use of LLMs for writing assistance does not affect the core methodology, scientific rigor, or originality of the work.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.