When Is a Team Worth Convening? Separating Aggregation, Sampling, and Member Selection

Anonymous authors
Paper under double-blind review

Abstract

Multi-agent systems (MAS) increasingly convene multiple heterogeneous language models as a team, and the resulting gains are often attributed to collaboration. A team's inference, however, bundles collaboration with other mechanisms, including how candidate answers are aggregated, how much sampling budget is spent, and how the single-member alternative is selected, so an observed team gain cannot by itself identify collaboration as the source of the improvement. We isolate these sources through controlled experiments on verifiable tasks with open non-thinking models of up to 14B parameters. We use two simple voting policies throughout, named after the answer camps (groups of responses giving the same answer) they count: CAMP-1 takes a single plurality vote, whereas CAMP adds votes only when first-pass agreement is low. Our findings show that the apparent team advantage changes with each of the three controls. The relative ranking of voting and synthesis depends on the aggregator, and with full candidate inputs, synthesis shows no average loss to an order-neutral vote. At a comparable budget, a hindsight-selected single member outperforms the tested voting teams, whereas neither side has an established advantage when selection is separated from evaluation. The relative value of routing depends strongly on label budget and pool composition: teams outperform the router at small label budgets in the development-selected pool, but this advantage is not established at larger budgets, while broader pools with a dominant member favor routing. Together, these results provide a practical decomposition for deciding whether to convene a team: before attributing a team gain to collaboration, one should determine how much comes from the aggregation rule, additional sampling, and the information available for member selection. This reframes a basic question for MAS—not only how a team should collaborate, but whether the task warrants a team at all.

1 Introduction

Multi-agent systems (MAS) increasingly answer a query by convening several language models, often from different model families, as a team rather than calling one model: members answer independently and vote on the result (Li et al., 2024), an aggregator reads their proposals and synthesizes a new answer (Wang et al., 2025; Li et al., 2026), or members critique and revise one another over rounds of debate (Du et al., 2024). The appeal is complementarity: different models may solve different problems. When such a team outperforms a single model, the gain is read as the benefit of collaboration and used to justify running several models where one might suffice.

A team, however, is a composite inference setting whose output depends on more than collaboration. First, aggregation decides which answer survives: a synthesizer can recover an answer no member returned, but can also rewrite a correct proposal into a wrong one. Second, a team consumes more generations than a single call, so comparing it with one sample conflates spending more with combining models. Third, the alternative is a chosen single member, whose quality depends on the router's labels. They are usually studied apart: Mixture-of-Agents, Self-MoA and debate study how models combine answers (Wang et al., 2025; Li et al., 2026; Du et al., 2024); self-consistency and inference scaling study additional samples (Wang et al., 2023; Brown et al., 2024; Snell et al., 2024);

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

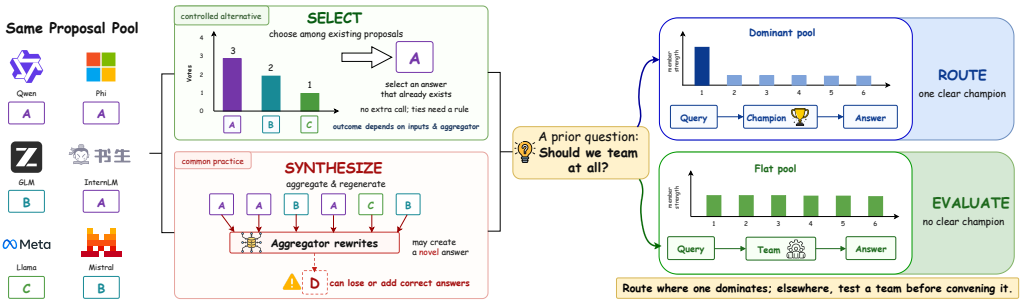

Figure 1: **From candidate answers to the value of a team.** Left: selection returns an existing answer; synthesis can also write a novel one, and which preserves more correctness depends on the inputs, tie rule and aggregator (§3). Right: whether to convene a team depends on how well a single member can be chosen and funded (§4–§5). Pools are schematic, not a validated routing threshold.

routing selects which model to call (Ong et al., 2025; Chen et al., 2023a). A team comparison spans all three, so its gain alone does not show collaboration is needed.

In this paper, we ask whether an observed team gain survives controls for aggregation, sampling budget and member selection, and thus whether collaboration is needed to explain it. Three complementary comparisons match these factors: fixed candidate records isolate aggregation; a fixed pool and generation library compare team and single-member sampling at similar budgets; and varying the information used to choose the single member tests whether team advantage survives realistic selection. Because these comparisons need answer-level scoring, we use verifiable tasks. The study covers a main pool of six open non-thinking models from six model families on fourteen benchmarks, a twelve-suite pool selected on development data, and a 4,032-cell pool–suite sweep. Heterogeneity makes member selection a real question: these models' errors are less correlated than one model's samples (Appendix F.1), yet they differ in strength. We test voting and single-round synthesis, not debate or tool use. Two simple voting policies serve as instruments throughout (Figure 2), named after the answer camps (groups of responses giving the same answer) they count: CAMP-1 takes one plurality vote over the members, and CAMP adds votes only when first-pass agreement is low. Each factor alone can change the verdict, as we show next.

Aggregation. Which operation wins depends on its implementation. When the aggregator reads only candidate prefixes and the vote breaks ties toward the largest member, selection leads synthesis by 9 points on MATH-500 and 14 on AIME. With full candidates and an order-neutral vote, no suite shows a vote advantage, and synthesis leads on AIME by 5 points. Synthesis loses where it writes answers no candidate proposed and gains where it picks among proposals; the aggregator decides the sign: on MATH-500 the two strongest add to the vote and the weakest subtracts (§3).

Sampling budget. Adaptive voting uses fewer generations than fixed thirty-vote sampling, and uniform sampling at the same average budget performs similarly: adaptive voting finds a budget rather than allocating one better. Beating shallow single-member sampling is not enough. On the main pool, a development-selected member shows no detected accuracy difference from the team at a third of its tokens, and in a same-library control a hindsight-chosen member at the team's budget exceeds both voting teams, partly through selection optimism (§4).

Member selection. A single member must also be found. In a pool selected with all development labels to minimize the gap between its two strongest members, after excluding the strongest candidate, both team policies outperform a router given ten labels per benchmark; the advantage is not established at larger label budgets or when selection is separated from evaluation. Across pools, a clearly dominant member favors routing, which is also cheaper (Figure 1, right; §5); there, routing finds the strongest member more often, but a missed choice costs far more.

Together, these results distinguish useful candidate answers from a useful team: team value depends on the model pool and on the information available to select a member, not on multi-model inference by default. Our contribution is this decomposition and the evaluation order it suggests: inspect the implemented aggregation on fixed candidates, give single members a comparable budget, and separate hindsight member quality from the selection information actually available, so that whether a task warrants a team is checked rather than assumed.

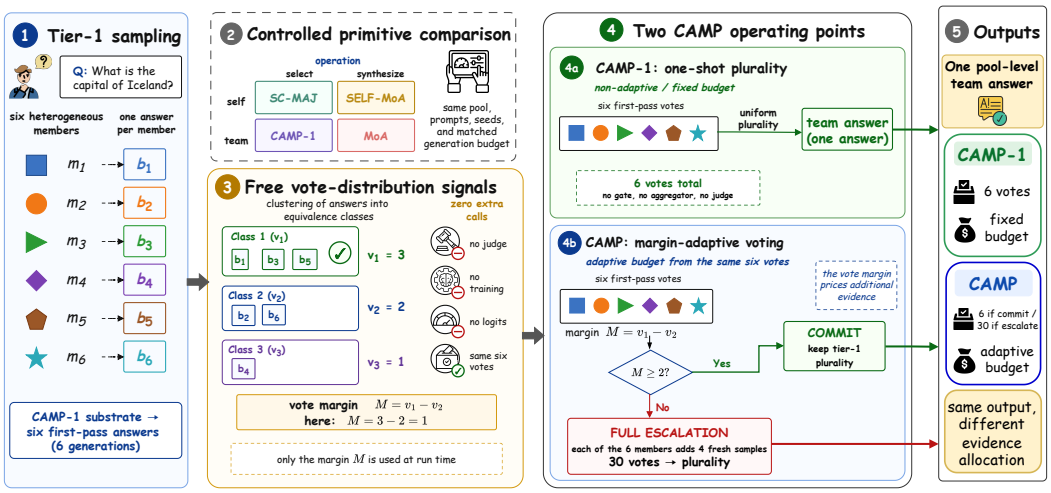

Figure 2: **The voting policies used in the controlled comparisons.** CAMP-1 takes one plurality vote over the six first-pass answers; CAMP expands to thirty votes when the first-pass margin is below two. The 2×2 inset summarizes comparison arms. The check mark illustrates correctness, which is unavailable at inference.

2 Methods and Experimental Design

We use a common set of answer operations and sampling policies to separate aggregation, budget, and member-selection effects.

2.1 Candidate Generation and Answer Operations

For an item x, a pool $\mathcal{P} = \{m_1, \ldots, m_6\}$ produces responses $r_1, \ldots, r_6$. A task-specific normalization $g(r)$ assigns an answer key; responses with the same nonempty key form an answer *camp*. *Selection* returns the earliest response in the largest camp, with camp ties broken by fixed member order. The output is a concrete response, and its own correctness supplies the evaluation score. The grouping and tie rule use no correctness labels. The main pool orders models by non-increasing parameter count, with Qwen3-14B before Phi-4; this is a member prior. We separately randomize tied camps and response order. *Synthesis* gives an aggregator the question and the candidate responses and asks it to generate a final answer. The archived pipeline passed character-capped prefixes; our main comparison passes full candidates (§3). Appendix A.5 gives the verbatim templates and a clipping audit. Each member serves as aggregator in turn; reporting the mean over these six hosts estimates an average operation effect, while fixed-host results show the variation that average conceals.

In the mechanism analysis, selection and synthesis use the same candidate records. The host's initial correctness and final correctness define four outcomes: keep-right, corrupted, repaired, and stay-wrong. We additionally partition synthesis outputs into matched answer strings, unmatched nonempty strings, and empty extractions, and count paired synthesis-only and selection-only successes in each partition.

2.2 Fixed and Adaptive Team Sampling

Naive denotes a single response, with accuracy and cost averaged over members. We call the voting policies CAMP, referring to their answer camps. CAMP-1 collects one response from each member and returns the plurality. CAMP adds a simple adaptive sampling step. With first-pass camp counts $v_1 \geq v_2 \geq \cdots$ (setting $v_2 = 0$ for one camp), its margin is $M = v_1 - v_2$:

$$\text{CAMP}(x) = \begin{cases} \text{plurality}(r_1, \ldots, r_6), & M \geq 2, \\ \text{plurality}(r_{1:30}), & M < 2. \end{cases}$$

In the second case, each member produces four additional responses; all thirty receive equal weight. Fixed-30 always draws those thirty votes. Thus the three policies vary the sampling schedule while

sharing the selection operation. CAMP uses the observed margin, without a trained selector or task-specific threshold labels. Self-consistency is the single-member sampling control, not an additional CAMP component.

2.3 SINGLE-MEMBER CONTROLS: BUDGET AND SELECTION INFORMATION

A router chooses one member per benchmark from its development-set SC-5 accuracy (plurality vote over five of that member's own samples) and then samples that member on held-out items. Varying its number of development labels measures the value of selection information. A separate *test-optimal* reference chooses the highest-scoring member on the evaluated items; it describes the available single-member option in hindsight. We report both because they answer different questions.

For the selected-pool budget control, both teams and single-member policies consume prefixes of the same deep-sampling libraries. The shallow member uses margin stopping at two with at most five samples. The near-budget member searches thresholds $\{2, 3, 4, 6, 8, 12, 16, \infty\}$ and available depths, choosing the largest mean completion-token spend that does not exceed the team's budget within each benchmark. This is retrospective cost calibration using generation lengths, not accuracy. A separate split-half check chooses the member and calibrates its cost on one half, freezes both, evaluates on the other half, and swaps halves. It tests how much of the hindsight comparison survives separate selection and evaluation.

2.4 MODELS, EVALUATION, AND STATISTICAL UNITS

The main pool contains Qwen3-14B, Phi-4, GLM-4-9B-0414, InternLM3-8B, Llama-3.1-8B, and Mistral-7B-v0.3. We use non-thinking generation with vLLM in bf16, temperature 0.6, top-p 0.95, and fixed per-item seeds. Fourteen benchmarks span mathematics, knowledge, science, code and free-text QA. The routing and budget studies use twelve suites, including HLE-Math; their populations and generation passes are specified in Appendix A.3.

Answer camps use normalized mathematical strings, MCQ letters, code execution signatures on model-generated inputs, or normalized TriviaQA strings. Official code tests and TriviaQA gold aliases are used only for scoring. The core mechanism and selected-pool controls score concrete responses; Appendix A maps scoring coverage for the historical benchmark grids and their replays.

Cost counts consumed completion tokens per final answer. In the synthesis cost plot, we also charge aggregator prefill at one quarter of its token count. The shared-library budget control (§4) matches completion tokens only. These are conditional inference comparisons, with pool construction and development sampling outside the budget. Per-benchmark intervals resample paired items, keeping all hosts of an item together; aggregate budget and routing intervals resample benchmarks. Reported intervals are 95% and condition on the recorded generations. Exact sample sizes, resampling settings and scoring sensitivities appear with each supporting analysis in the appendix.

3 AGGREGATION DEPENDS ON INPUTS, TIE RULE AND AGGREGATOR

We compare selection and synthesis on identical candidate records, both scored by the same scorer (Appendix B.1). Two rarely reported choices decide the result: how much of each candidate the aggregator reads, and how the vote breaks ties.

3.1 THE SAME COMPARISON UNDER TWO INPUT CONDITIONS

The archived synthesis pipeline passed each aggregator only a fixed-length prefix of every candidate, which often cut off the candidate's final answer (Appendix A.5). We therefore repeated only the aggregation call with full candidates on four suites, keeping prompts, candidate order, temperature and per-item seeds. Each condition is paired on identical candidates; because the two conditions come from different generation runs, we report both rather than attribute their difference to clipping alone.

Table 1 reads as a chain. With prefix inputs and the size-ordered vote, selection wins by a wide margin on the mathematics suites (9–14 points). Full candidate inputs remove most of that mar-

Suite	Items	Size-ordered tie rule		Order-neutral tie rule
		Prefix input	Full input	Full input
MATH-500	500	-9.13 $[-11.60, -6.73]$	-3.00 $[-4.77, -1.20]$	$+0.37$ $[-1.47, +2.33]$
AIME 83–24	300	-14.11 $[-18.44, -9.89]$	-0.50 $[-3.28, +2.28]$	$+5.22$ $[+2.61, +7.94]$
MMLU-Pro	700	-3.21 $[-5.19, -1.26]$	-1.10 $[-2.83, +0.69]$	$+0.67$ $[-0.95, +2.33]$
GPQA	197	-6.60 $[-11.93, -1.35]$	$+0.34$ $[-3.98, +4.74]$	$+2.33$ $[-1.57, +6.26]$

Table 1: **Synthesis minus selection on identical candidates** (points; mean over six aggregators; paired item-bootstrap 95% intervals). Negative values favor the vote. The size-ordered rule breaks camp ties by member order, with larger models first; the order-neutral rule picks a tied camp at random. Prefix and full input are separate generation runs.

gin, and an order-neutral tie rule removes every detected vote advantage, with synthesis ahead on AIME. Ranking members by held-out accuracy instead reproduces the size order and its result (Appendix B.2): in this pool the size prior is what development labels would recommend, and such a vote still leads average synthesis on MATH-500 and MMLU-Pro; the order-neutral vote is the label-free case. A synthesis-versus-vote gap therefore measures an implementation, not the operations in general.

3.2 Two channels with opposite signs

A synthesized answer either repeats a string some candidate proposed (*matched*), introduces a new one (*novel*, a string-level category), or yields none. Splitting synthesis minus selection along these strata exposes two channels of opposite sign (Figure 3b). With prefix inputs, novel strings account for most of the loss (-6.57 of -9.13 points on MATH-500), consistent with an aggregator that cannot see a candidate's final answer writing its own. With full inputs this channel shrinks to about a third on MATH-500 and nearly vanishes on AIME, and under the order-neutral vote the matched channel, where synthesis chooses among proposed answers, favors synthesis, most clearly on AIME (Table 3). Synthesis loses by inventing answers and gains by choosing among them; the input it receives decides which dominates.

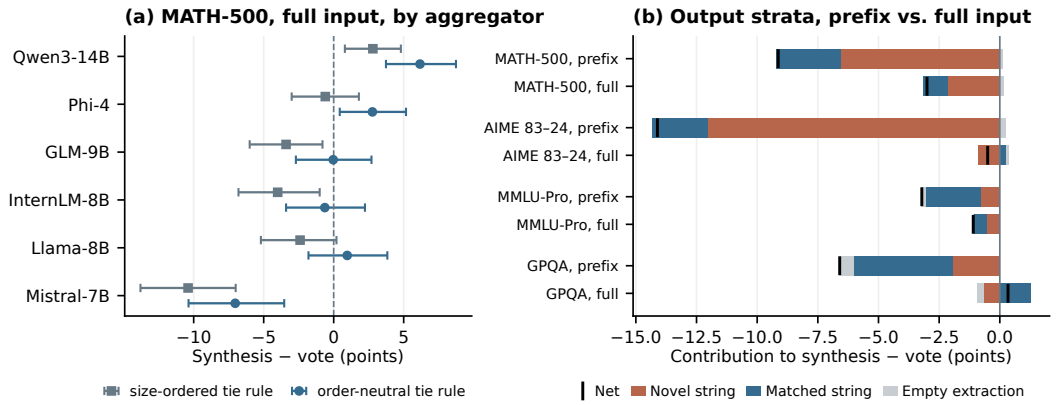

Figure 3: **Aggregator identity and output strata.** (a) Synthesis minus selection on MATH-500 with full candidate inputs, by aggregator, under the two vote tie rules; bars are paired item-bootstrap 95% intervals. (b) Contributions to synthesis minus selection by output stratum, with the size-ordered tie rule held fixed, for prefix and full inputs; ticks mark the net difference. Per-host differences for all four suites are in Appendix B.1.

3.3 The aggregator determines the sign

Access to complementary proposals is not the same as the ability to use them. The six-host mean hides a large identity effect (Figure 3a): with full inputs, the strongest aggregator, Qwen3-14B, exceeds the order-neutral vote on MATH-500 by 6.2 points, and the weakest, Mistral-7B, trails it by 7.0. On all four suites, in point estimate, the two strongest aggregators are ahead of this vote and Mistral-7B is behind (Table 4). Whether synthesis helps is thus a property of the aggregator as much as of the operation.

Implications. Three choices that papers rarely report (what the aggregator reads, how the vote breaks ties, and which model aggregates) can each reverse a vote-versus-synthesis comparison, so a comparison should state all three. In a heterogeneous team the tie rule also acts as a member prior: a plain vote that breaks ties toward the strongest member already performs a weak form of routing.

Scope. The four suites above are the ones with a full-input rerun. The broader fourteen-task grid and the within-model Self-MoA comparison used the prefix-limited pipeline; Appendix B.3 reports them as results of that pipeline, not as evidence about the operations. The full-input rerun has one generation and a 1,536-token output cap, which binds on 107 of 1,800 AIME syntheses. The next question is independent of the aggregation operation: could a strong single member use the team's sampling budget better?

4 Sampling budget changes the team comparison

4.1 Paying for more votes

Figure 4 places a single response (Naive), CAMP-1, CAMP and fixed thirty-vote sampling on one cost–accuracy plane. CAMP spends extra samples only where the first-pass vote is split, so it matches Fixed-30 without a detected accuracy difference at a fraction of its tokens (Appendix C). Uniform sampling given CAMP's measured average budget does as well (Appendix C.2): the margin finds a budget rather than allocating it better. We use CAMP to study team value at an adaptive budget, not as a new allocation principle.

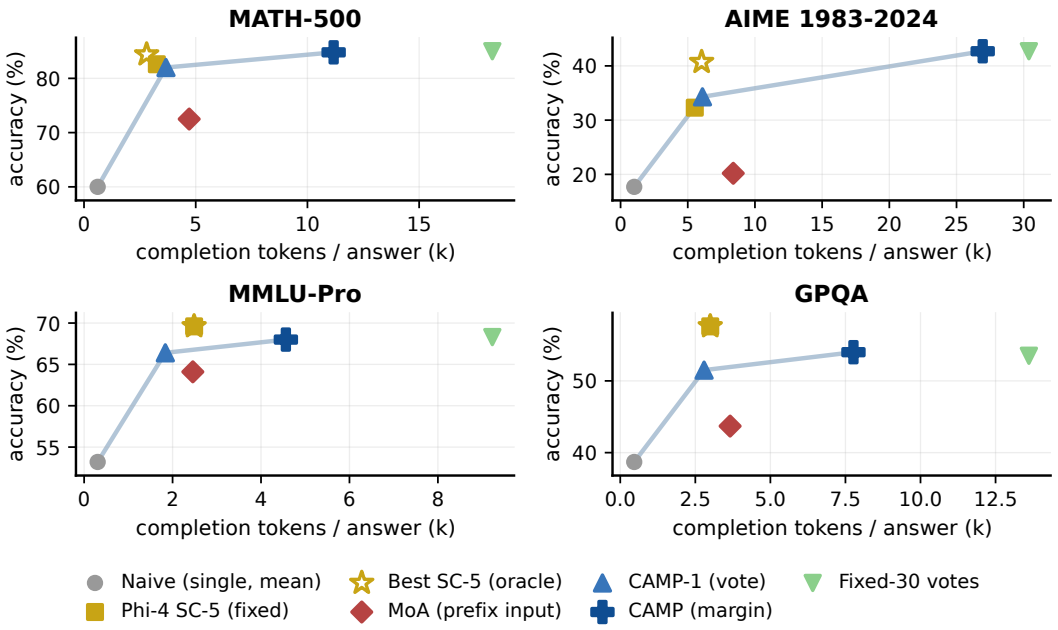

Figure 4: **From one response to fixed and adaptive voting.** Lines connect only Naive (one response, averaged over members), CAMP-1 and CAMP, in cost order. They are guides between measured operating points, not a global Pareto frontier. Fixed Phi-4, test-optimal SC-5, pool-average MoA (prefix-limited pipeline) and Fixed-30 remain separate reference points. Each answer pays for consumed generations; synthesis also includes the stated prefill charge. Figure data come from the cost-instrumented pass (Appendix A.3).

A strong member already challenges the team's cost advantage: on the main pool, a development-selected member shows no detected accuracy difference from CAMP at about a third of its tokens (Appendix C.3). A fair test must therefore let the single member spend approximately what the team spends.

4.2 A SHARED-LIBRARY BUDGET CONTROL

The selected-pool study uses all development labels across eleven candidate models: exclude the strongest aggregate candidate, then choose six members to minimize the aggregate best-minus-second gap. The resulting pool is fixed before evaluating 2,437 held-out items from twelve suites. Every arm reads the same newly generated response libraries. We first compare teams with a shallow member (margin two, at most five samples), then with a near-budget member using the cost calibration in §2. At each budget, the highest-scoring member on the test items is the hindsight reference; member identity can change as sampling deepens.

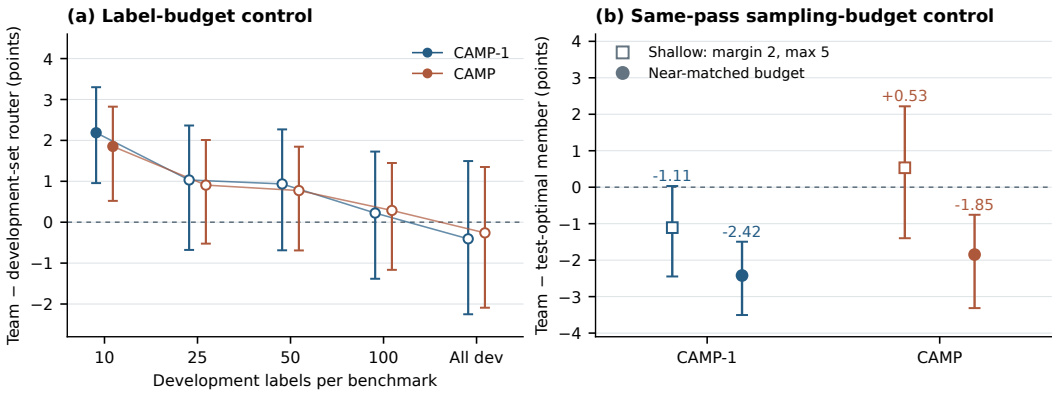

Figure 5: **Separate the information budget from the sampling budget.** (a) Team minus a member chosen from development labels, at near-matched completion-token budgets. (b) Team minus the test-optimal member under shallow and near-matched sampling, using the same generations. Both panels use concrete-response scoring and 95% benchmark-bootstrap intervals. Filled label-curve points exclude zero. The test-optimal reference is an oracle; the separated-selection comparison is reported in §5.

The comparison turns with the budget (Figure 5b). Against shallow single-member sampling, CAMP's estimate is positive but uncertain; at near-matched budgets, the hindsight-best member exceeds both CAMP-1 and CAMP, by 2.42 and 1.85 points. The reversal survives exact expected-cost matching, and correcting for the optimism of choosing the best of six members on the test items leaves CAMP-1 behind while bringing CAMP's interval to zero (Appendix D.3). A team's advantage over a cheap single member is therefore not evidence that its budget is better spent across members.

This control was completed after the original shallow-budget study and uses retrospective cost calibration. The frozen pool, split and label budgets, together with protocol deviations, are documented in Appendix D. Its role is to identify the budget dependence of the comparison. To know whether the stronger reference can be found in practice, we must also control member-selection information.

5 MEMBER-SELECTION INFORMATION CHANGES TEAM VALUE

5.1 A TEAM CAN HELP WHEN SELECTION LABELS ARE SCARCE

Conditional on the pool constructed with all development labels and the strongest candidate excluded (§4), we hold near-budget sampling fixed and vary the router's selection labels. With ten labels per benchmark, both teams outperform the router, CAMP-1 by 2.19 and CAMP by 1.85 points; from 25 labels on, no advantage is established (Figure 5a). The router they beat is informed rather than random: at ten labels it already exceeds a randomly chosen member (Appendix D.3). In this pool the team therefore insures against a poorly informed choice rather than beating a well-informed one. Nor is the oracle gap of §4 a deployable router gain: selecting and calibrating a member on one half of each benchmark and evaluating it on the other establishes neither side's advantage. Both patterns hold under aligned development scoring and small-cluster intervals (Appendix D.4). These are information budgets within the fixed pool, not the labels needed to construct it, and the registered crossover prediction is not confirmed (Appendix D.3).

5.2 Member dominance identifies a routing-favorable setting

We next vary pool composition. Nine models on twelve suites and every four- to six-member subset yield 4,032 overlapping pool–suite cells. The *anchor* is the member selected by development accuracy; the *dominance gap* is the best minus second-best development accuracy. Figure 6 reports both team-minus-router accuracy and their token ratio, so a positive accuracy estimate is not mistaken for a free gain.

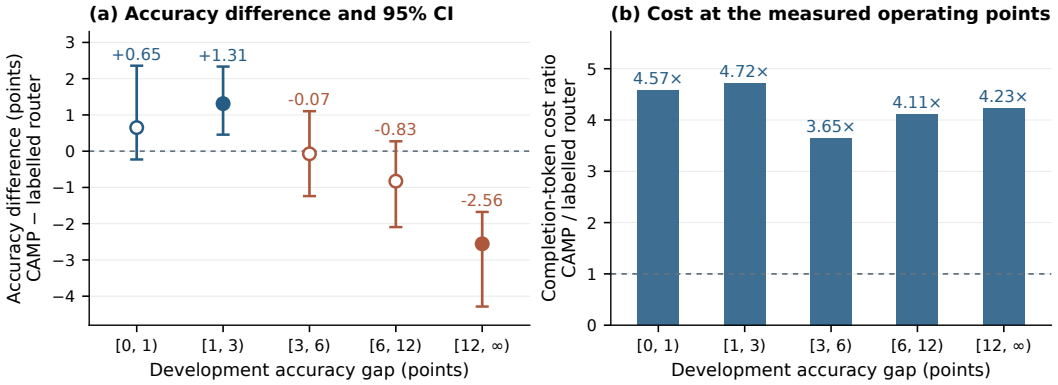

Figure 6: **A dominant member changes the case for aggregation.** CAMP minus the labelled router across 4,032 pool–suite cells, grouped by development dominance. Accuracy uses concrete-response scoring; intervals resample the twelve benchmarks. Costs describe the same cells at the policies' measured operating points; the router adaptively samples its selected member up to five times. This is not an equal-budget experiment. The most dominated bucket favors routing; the flattest bucket does not establish a team advantage. Appendix E gives construction, full grids and development-anchor sensitivity.

Dominance separates the regimes (Figure 6). When one member is clearly strongest, routing is both more accurate, by 2.56 points in the most dominated bucket, and several times cheaper. Toward the flat end the team's point estimates turn positive, but they cost several times the router's tokens, shrink to near zero or below under budget-capped routing (Table 16), and in an archived control shrink toward zero once the router's own selection error is removed (Appendix E.2). Recomputing development anchors and buckets preserves these patterns (Appendix E.1). Labels still matter in dominated pools, for a different reason: agreement-based selection finds the dominant member more often there, but a miss costs far more (Appendix E.3). The selected pool of §4 is a complementary information control, not a test of a "flat pools should convene" rule: only five of its twelve suites fall in the flattest bucket, and admitting the excluded strongest member lets a single member overtake the team under concrete-response scoring (Appendix D).

Implications. Whether to convene a team is a question about the pool and the labels at hand, not about multi-model inference as such. For pools like ours, and not as a universal gap threshold, the results suggest a simple reading: when development data reveal a dominant member, route to it; when members are close and labels are few, a vote guards against a wrong pick at several times the router's tokens; otherwise, and whenever a few labels cannot tell these cases apart, check the team against a budget-matched, label-matched single member before crediting collaboration.

6 Related work

Combining answers and preserving correctness. Self-consistency selects an existing answer (Wang et al., 2023); MoA and Self-MoA regenerate from proposals (Wang et al., 2025; Li et al., 2026). LLM-Blender separates learned ranking from generative fusion (Jiang et al., 2023); Universal Self-Consistency uses an LLM to select among candidates (Chen et al., 2023b). Debate and correctness-preservation studies examine how interaction changes answers (Du et al., 2024; Choi et al., 2025; Wynn et al., 2025; Nikeghbal et al., 2026). A single agent with strong prompts or self-consistency can match multi-agent discussion and automatically designed systems (Wang et al., 2024; Jwalapuram et al., 2026), and judge-based selection among diverse proposals can outperform

MoA-style synthesis (Maryanskyy et al., 2026). Choi et al. (2025) find that voting accounts for most measured debate gains; Smit et al. (2024) show the importance of protocol tuning in debate comparisons. Wunderlich et al. (2026) report MoA outperforming self-consistency by 2.7 points at equal compute on MMLU-Pro, with larger models and layered aggregation. Our paired accounting on identical candidates shows why such orderings are fragile: input exposure, tie rule and aggregator identity each change the sign. Where these studies compare a team with one single-system baseline, we vary aggregation, budget and selection information in turn on shared records. Self-consistency, More Agents and Self-MoA sample one model; our teams mix model families, so choosing which member to trust is part of the comparison.

Sampling budgets. Sampling and voting can scale performance (Li et al., 2024), though more calls need not improve it monotonically (Chen et al., 2024). Repeated sampling expands candidate coverage (Brown et al., 2024), while prompt-dependent compute allocation changes its efficiency (Snell et al., 2024). Adaptive-consistency and EMS study when to stop sampling (Aggarwal et al., 2023; Liu et al., 2026), and ModelSwitch uses answer consistency to switch among several models during repeated sampling (Chen et al., 2026). We use fixed and margin-adaptive voting only as instruments to compare team and single-member budgets, rather than claim a new stopping principle.

Member selection and complementary errors. Routers and cascades allocate calls across models (Ong et al., 2025; Chen et al., 2023a; Yue et al., 2024); Smoothie studies label-free routing (Guha et al., 2024). Concurrently, Chen (2026) analyzes a co-failure ceiling across 67 models and finds that combinations rarely beat the strongest member without a strong routing signal. We complement this coverage perspective by tracing what the aggregation operation does to fixed proposals and varying the labels available to select a member at comparable sampling budgets.

7 Discussion and conclusion

The experiments distinguish three sources of team performance. A heterogeneous pool supplies candidate answers, but whether aggregation preserves their correctness depends on the inputs the aggregator reads, the vote's tie rule and the aggregator itself. Additional samples help, but a stronger single-member baseline changes the apparent gain. Finally, with few selection labels in the development-constructed pool, the tested router leaves room for a team advantage.

The practical contribution is a way to check where a team gain comes from before crediting it to collaboration. For multi-agent systems, a more basic question than how a team should collaborate is whether a task warrants convening one.

Scope and limitations. Our principal comparisons concern small non-thinking models and verifiable answers. The budget and selection controls test voting teams; that full-input synthesis matches or exceeds the vote does not show that a synthesis team beats a comparably sampled single member. Reasoning-mode controls further favor strong single-member alternatives (Appendix G). The selected pool is itself chosen with all development labels after excluding the strongest candidate. Unknown training-data overlap can contribute to measured dominance. Intervals are unadjusted for multiplicity and condition on saved outputs from several generation passes and few benchmarks (Appendix A).

AI use statement

We used generative AI assistants to draft, restructure and polish manuscript text; to write and revise data-processing, analysis, figure and build scripts; and to produce simulated reviews of drafts, which flagged issues such as the truncated synthesis inputs and suggested checks. The research questions, experimental design and final claims were decided by the authors. The language models evaluated in this paper are experimental subjects, not assistants. No reported number was produced or edited by a generative AI tool: every number is computed by scripts from recorded model outputs, with deterministic answer scoring and fixed-seed bootstrap resampling. We checked AI-written code against the protocols stated in the paper and through reproduction checks. The analysis replay reproduces the archived results exactly. Re-scoring all 6,600 full-input mathematics answers changed no stored flag. Raw outputs are checksummed, and a release script compares every full-input number printed

in Section 3 and Appendix B with the analysis output. The authors revised and approved all AI-assisted text. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

Reproducibility statement

The supplementary material specifies answer grouping, evaluation passes, statistical units, and scoring coverage. The artifact includes candidate and aggregation records, selected-pool sampling libraries, CPU analysis scripts, and figure sources (Appendix A).

References

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with llms. *Proceedings of EMNLP*, 2023.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024. URL https://arxiv.org/abs/2407.21787.

Jianhao Chen, Zishuo Xun, Bocheng Zhou, Han Qi, Hangfan Zhang, Qiaosheng Zhang, Yang Chen, Wei Hu, Yuzhong Qu, Wanli Ouyang, and Shuyue Hu. Do we truly need so many samples? multi-LLM repeated sampling efficiently scales test-time compute. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

Josef Chen. When does combining language models help? a co-failure ceiling on routing, voting, and mixture-of-agents across 67 frontier models. *arXiv preprint arXiv:2606.27288*, 2026. URL https://arxiv.org/abs/2606.27288.

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)*, 2023a.

Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. Are more LLM calls all you need? towards the scaling properties of compound AI systems. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*, 2023b.

Hyeong Kyu Choi, Xiaojin Zhu, and Sharon Li. Debate or vote: Which yields better decisions in multi-agent large language models? In *Advances in Neural Information Processing Systems (NeurIPS), Spotlight*, 2025.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *International Conference on Machine Learning (ICML)*, 2024.

Neel Guha, Mayee F. Chen, Trevor Chow, Ishan S. Khare, and Christopher Ré. Smoothie: Label-free language model routing. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-Blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.

Prathyusha Jwalapuram, Hehai Lin, et al. The illusion of multi-agent advantage. *arXiv preprint arXiv:2606.13003*, 2026.

Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More agents is all you need. *Transactions on Machine Learning Research (TMLR)*, 2024.

Wenzhe Li, Yong Lin, Mengzhou Xia, and Chi Jin. Rethinking mixture-of-agents: Is mixing different large language models beneficial? *Transactions on Machine Learning Research (TMLR)*, 2026.

Yiqing Liu, Hantao Yao, Wu Liu, and Yongdong Zhang. EMS: Multi-agent voting via efficient majority-then-stopping. *arXiv preprint arXiv:2604.02863*, 2026.

Ivan Maryanskyy, Mikhail Budnikov, and Arman Kaliyev. When agents disagree: The selection bottleneck in multi-agent llm pipelines. *arXiv preprint arXiv:2603.20324*, 2026.

Nafiseh Nikeghbal, Amir Hossein Kargaran, Shaghayegh Kolli, and Jana Diesner. Who flips? self- and cross-model counterarguments reveal answer instability in llms. *arXiv preprint arXiv:2606.16011*, 2026.

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 with preference data. In *International Conference on Learning Representations (ICLR)*, 2025.

Andries Smit, Paul Duckworth, Nathan Grinsztajn, Thomas D. Barrett, and Arnu Pretorius. Should we be going MAD? a look at multi-agent debate strategies for llms. *International Conference on Machine Learning (ICML)*, 2024.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024. URL https://arxiv.org/abs/2408.03314.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. In *International Conference on Learning Representations (ICLR)*, 2025.

Qineng Wang, Zihao Wang, Ying Su, Hanghang Tong, and Yangqiu Song. Rethinking the bounds of LLM reasoning: Are multi-agent discussions the key? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6106–6131, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Haotian Wu, Bo Xu, Yao Shu, Menglin Yang, and Chengwei Qin. Thinking with nothinking calibration: A new in-context learning paradigm in reasoning large language models. *arXiv preprint arXiv:2508.03363*, 2025.

Florian Valentin Wunderlich, Lars Benedikt Kaesberg, Jan Philip Wahle, Terry Ruas, and Bela Gipp. Multi-agent reasoning improves compute efficiency: Pareto-optimal test-time scaling. *arXiv preprint arXiv:2605.01566*, 2026. URL https://arxiv.org/abs/2605.01566.

Andrea Wynn, Harsh Satija, and Gillian Hadfield. Talk isn't always cheap: Understanding failure modes in multi-agent debate. *ICML Workshop on Multi-Agent Systems; arXiv:2509.05396*, 2025.

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. Large language model cascades with mixture of thought representations for cost-efficient reasoning. In *International Conference on Learning Representations (ICLR)*, 2024.

A Evaluation protocols and evidence coverage

A.1 Generation and answer equivalence

Models are served with vLLM in bf16. The main campaign uses fixed per-item seeds, base seed 20260724, temperature 0.6, top-p 0.95, and a `/no_think` suffix for Qwen3-14B. The extended hard-mathematics campaign caps both proposal and synthesis completions at 1,536 tokens. The shared-library selected-pool caps are 1,280 for mathematics and 1,024 for MCQ. A common cap can affect proposal and synthesis truncation differently.

Mathematics uses `math-verify` 0.9.0 for correctness and normalized answer strings for camps. MCQ camps use extracted letters. HumanEval signatures execute responses on inputs generated from the function prompt, without official expected outputs; hidden official tests are used for scoring only. TriviaQA uses 800 validation items (rc.nocontext) and an `ANSWER:` line. Its camps use lowercasing and punctuation, article and whitespace normalization, with official aliases reserved for scoring. These tasks evaluate verifiable answers rather than open-ended preferences.

A.2 Statistical units and scoring

Per-task intervals use paired item bootstrap. The core mechanism uses 10,000 resamples, seed 7, retaining every host of each resampled item; main-protocol intervals generally use 4,000. Hosts are six fixed models, not an independently sampled model population. Aggregate routing and budget comparisons resample benchmarks, generally 20,000 times, because pool subsets overlap. S3 uses twelve clusters and additionally reports jackknife-t_{11} intervals. Exact exceptions are specified by their tables. Intervals are 95%, unadjusted for multiplicity, and conditional on saved generations. An interval containing zero does not establish equivalence.

Concrete-response scoring grades the actual selected response. Archived camp credit instead accepts a winning answer key if any response with that key is correct. These differ when equal keys have conflicting verifier flags. The mechanism and selected-pool analyses report concrete responses and sensitivities; MATH responses are also regraded symmetrically (Appendix B.3).

A.3 Comparison populations and generation passes

The fourteen-task grid supports broad operation comparisons. The checkpointed-budget, routing and selected-pool populations instead contain twelve suites including HLE-Math. Three passes have distinct roles:

- The main operation pass supplies the fourteen-task grid and the prefix-limited synthesis (Appendix B.3); the full-input rerun reuses its candidates (Appendix B.1).
- The twelve-suite checkpointed pass supplies CAMP versus Fixed-30 intervals. The eight-suite cost-instrumented pass supplies Figure 4. Their CAMP scores agree on shared suites, but Fixed-30 scores differ, including 3.1 points on GPQA; CAMP-1 differs from the main grid by at most 0.3 points. Each comparison retains its own records.
- The selected-pool shared-library pass (S3; S1–S3 are defined in Appendix D) supplies both team and single-member prefixes from one fresh deep-sampling library. Its samples are separate from the original selected-pool and excluded-member control.

Where main-text results come from. Section 3 (Table 1, Figure 3) uses the main operation pass for candidates and prefix-limited synthesis (AIME candidates from the hard-mathematics extension), and the full-input rerun for full-input synthesis. In Section 4, the CAMP versus Fixed-30 comparison uses the twelve-suite checkpointed pass, Figure 4 the cost-instrumented pass, and the main-pool router comparison the main-pool routing libraries; the shared-library control (Figure 5b) and the label curve of Section 5 (Figure 5a) use the S3 library, while the excluded-member control uses the original selected-pool pass. The dominance sweep (Figure 6) uses the nine-model sweep pass, and reasoning-mode pools use their own generations (Appendix G). Every comparison is computed within one pass. Where passes overlap, they differ by up to 3.1 points (Fixed-30 on GPQA), and selected-pool CAMP scores 64.67 in the original pass against 62.91 in the S3 library; we have not estimated variation across repeated generation campaigns.

A.4 ARTIFACT COVERAGE

Raw generations, normalized keys, stored correctness and costs, development splits, CPU replay scripts and figure inputs accompany the analysis. The full-input rerun, the selected-pool libraries (all 72) and the dominance sweep (all 4,032 paired cells) are replayed from raw records with concrete-response scoring. The fourteen-task grid and main-pool routing retain their archived scoring; their full-response regrading is incomplete. Missing cells (one GPQA synthesis item, one TriviaQA record) are not filled from other passes.

A.5 SYNTHESIS PROMPTS AND INPUT EXPOSURE

The recovered synthesis drivers are `generate_moa.py` and `generate_moa_from_samples.py`. Their system instructions are reproduced below; display wrapping is editorial.

Mathematics system message.

> You are given a math problem and candidate solutions from several assistants. Some may be wrong. Critically synthesize them into one correct solution. Reason concisely, then end with a line `FINAL_ANSWER: <answer>`.

Multiple-choice system message.

> You are given a multiple-choice question and candidate responses from several assistants. Some may be wrong. Critically synthesize them into the single best answer. Reason briefly, then end with `ANSWER: X` (single letter).

The mathematics user template is:

```
Problem:
{problem}
Candidate solutions:
[Assistant 1]
{proposal_1[:1400]}
...
[Assistant 6]
{proposal_6[:1400]}
Provide the single best final solution.{suffix}
```

Consecutive candidates are separated by two newline characters. For core MCQ, replace `Problem:` by `Question:`, `Candidate solutions:` by `Candidate responses:`, use 1,200 characters per candidate, and end with "Give the single best final answer." The extension driver uses 1,400 characters for both task types. The suffix is `/no_think` for Qwen3-14B and empty for the other hosts. The code driver uses 1,500 characters per candidate inside Python fences and instructs:

> You are given a programming problem and candidate solutions from several assistants. Some may be wrong. Synthesize them into one correct, self-contained solution. Return ONLY a single ```` ```python ... ``` ```` code block, no explanation.

These are task-specific synthesis prompts, not an unmodified implementation of every MoA architecture. Candidate order is the fixed six-member order in Appendix B.2. The full-input rerun uses the same templates without the character limits.

Table 2 applies these caps to the source proposals. The vote reads full recorded answer keys, so under these caps the two arms see unequal text. The archived records store no finish reasons or token usage; the full-input rerun records both (Appendix B.1).

B AGGREGATION EVIDENCE

This appendix supports Section 3. The full-input rerun (Appendix B.1) is the paired comparison of the two aggregation operations; the archived prefix-limited pipeline (Appendix B.3) is retained as the second input condition and for its broader task grid.

Suite	Complete items	Clipped proposals	Items with clipping	Markers cut off
MATH-500	500	1497/3000	380/500	987
GSM8K	500	275/3000	163/500	181
AIME 83–24	300	1632/1800	300/300	956
MMLU-Pro	700	1557/4200	633/700	1203
GPQA	197	727/1182	197/197	568

Table 2: Applying the archived character caps to source proposal texts. "Markers cut off" counts proposals whose last explicit `FINAL_ANSWER:` or `ANSWER:` marker starts beyond the cap; it is not a correctness judgment. Input clipping differs from output-token truncation. GPQA uses the complete paired population.

B.1 Full-input synthesis rerun

The rerun repeats only the aggregation call over the archived candidates, with every candidate passed in full: 6 hosts $\times$ MATH-500, AIME 83–24, MMLU-Pro and GPQA, 10,188 calls, served with a 16,384-token context and a 1,536-token output cap. Prompts, candidate order, temperature 0.6, top-p 0.95 and the per-item seed are those of the archived pipeline. Paired analyses use the 1,697 items common to the candidate records and both synthesis runs (GPQA: 197). All 6,600 full-input mathematics syntheses and AIME candidates were re-scored with `math-verify` 0.9.0 and the archived verifier; none of the stored correctness flags or normalized answers changed. The output cap binds on $27/3{,}000$ MATH-500, $107/1{,}800$ AIME, $9/4{,}200$ MMLU-Pro and $4/1{,}182$ GPQA syntheses. Intervals use 10,000 paired item resamples (seed 7), keeping all six hosts of a resampled item.

Table 3 decomposes the full-input entries of Table 1 into output strata and transitions, and Table 4 gives the per-aggregator differences behind Figure 3a. Under a uniformly random member order, which also randomizes the representative response, synthesis minus vote is $+1.34$ $[-0.40, +3.24]$, $+4.83$ $[+2.28, +7.44]$, $+0.67$ $[-0.95, +2.33]$ and $+2.33$ $[-1.57, +6.22]$ on the four suites. Dropping the capped AIME cells moves the random-order value to $+5.75$; this selects on difficulty and is only a diagnostic.

Suite	Vote tie rule	Total	Novel string	Matched string	Δcorrupt	Δrepair
MATH-500	size-ordered	-3.00 $[-4.77, -1.20]$	-2.17 $[-2.87, -1.50]$	-1.00 $[-2.43, +0.50]$	$+0.20$	-2.80
	order-neutral	$+0.37$ $[-1.47, +2.33]$	-1.42 $[-2.12, -0.71]$	$+1.61$ $[+0.09, +3.27]$	-0.67	-0.31
AIME 83–24	size-ordered	-0.50 $[-3.28, +2.28]$	-0.89 $[-2.28, +0.50]$	$+0.28$ $[-1.61, +2.28]$	-0.06	-0.56
	order-neutral	$+5.22$ $[+2.61, +7.94]$	$+0.31$ $[-0.96, +1.63]$	$+4.63$ $[+2.61, +6.83]$	-1.22	$+4.00$
MMLU-Pro	size-ordered	-1.10 $[-2.83, +0.69]$	-0.55 $[-0.83, -0.29]$	-0.52 $[-2.19, +1.19]$	-0.19	-1.29
	order-neutral	$+0.67$ $[-0.95, +2.33]$	-0.56 $[-0.84, -0.30]$	$+1.25$ $[-0.30, +2.83]$	-0.83	-0.16
GPQA	size-ordered	$+0.34$ $[-3.98, +4.74]$	-0.68 $[-1.52, +0.08]$	$+1.27$ $[-2.88, +5.50]$	-0.59	-0.25
	order-neutral	$+2.33$ $[-1.57, +6.26]$	-0.61 $[-1.34, +0.13]$	$+3.19$ $[-0.58, +6.99]$	-1.26	$+1.06$

Table 3: **Full-input synthesis minus vote, by output stratum and transition** (points of all host$\times$item cells). Empty-extraction contributions are at most 0.29 in magnitude and complete the sum. Δcorrupt and Δrepair are synthesis minus vote in the share of cells where a correct own proposal becomes wrong, and where a wrong one becomes right; Total $=$ Δrepair $-$ Δcorrupt.

	MATH-500		AIME 83–24		MMLU-Pro		GPQA	
Aggregator	size	neutral	size	neutral	size	neutral	size	neutral
Qwen3-14B	$\mathbf{+2.80}$	$\mathbf{+6.17}$	$+2.33$	$\mathbf{+8.06}$	-0.57	$+1.19$	$+2.03$	$+4.02$
Phi-4	-0.60	$\mathbf{+2.77}$	$+2.00$	$\mathbf{+7.72}$	$\mathbf{+3.86}$	$\mathbf{+5.62}$	$\mathbf{+9.14}$	$\mathbf{+11.13}$
GLM-9B	$\mathbf{-3.40}$	-0.03	-2.00	$\mathbf{+3.72}$	$+0.71$	$\mathbf{+2.48}$	$+2.54$	$+4.53$
InternLM-8B	$\mathbf{-4.00}$	-0.63	$+1.00$	$\mathbf{+6.72}$	$\mathbf{-3.29}$	-1.52	-4.06	-2.07
Llama-8B	-2.40	$+0.97$	$+2.33$	$\mathbf{+8.06}$	-1.14	$+0.62$	-0.51	$+1.48$
Mistral-7B	$\mathbf{-10.40}$	$\mathbf{-7.03}$	$\mathbf{-8.67}$	-2.94	$\mathbf{-6.14}$	$\mathbf{-4.38}$	$\mathbf{-7.11}$	-5.12

Table 4: **Per-aggregator full-input synthesis minus vote** (points), against the size-ordered and the order-neutral vote. Bold marks a 95% paired-bootstrap interval that excludes zero; intervals are unadjusted.

B.2 Vote tie rules

The size-ordered rule resolves a top-count tie by the earliest member in the order Qwen3-14B, Phi-4, GLM-9B, InternLM-8B, Llama-8B, Mistral-7B, and represents the winning camp by its earliest

response. It uses no labels but encodes a member preference: Qwen3-14B belongs to the winning tied camp on 85 of 86 MATH ties and 165 of 167 AIME ties. Top-count ties occur on $86/500$ MATH, $167/300$ AIME, $107/700$ MMLU-Pro and $39/197$ GPQA items. The order-neutral rule draws the winning camp uniformly among tied camps and keeps its first response; the random-member rule also randomizes that response. Both are integrated exactly over the randomization before the item bootstrap. Mathematical camps are keyed by normalized answer strings, and verification operates on full responses, so equal keys can carry different correctness flags (30 MATH winning camps); every vote is scored by its concrete representative response.

A development-accuracy tie rule. A deployable vote may also break ties by member accuracy measured on labelled items. We split each suite's items in half at random, rank members by candidate accuracy on the other half, and apply the fixed-order rule with that ranking, averaging over 20 splits; no item's own labels enter its vote. The ranking places Qwen3-14B and Phi-4 first in every split, so the rule nearly reproduces the size order. Its vote scores 82.00, 34.33, 68.29 and 49.64 on MATH-500, AIME 83–24, MMLU-Pro and GPQA, and full-input synthesis minus this vote is -3.40 $[-5.10, -1.67]$, -0.50 $[-3.28, +2.28]$, -2.10 $[-3.88, -0.26]$ and $+0.95$ $[-3.65, +5.63]$. Thus the synthesis lead in Table 1 holds against the label-free order-neutral vote, while a vote with development information keeps an advantage over average synthesis on MATH-500 and MMLU-Pro. The strongest aggregators still exceed the size-ordered vote on several suites (Table 4).

B.3 Prefix-limited pipeline

The archived pipeline passed each aggregator character-capped candidate prefixes (Appendix A.5); it supplies the first column of Table 1 and the prefix bars of Figure 3b. Table 5 gives its fourteen-task grid, which is the only evidence outside the four rerun suites and describes that pipeline, not the aggregation operations. Within one model on MATH-500, Self-MoA scores 61.4 against 64.1 for SC-5 (-2.70 $[-3.80, -1.63]$); InternLM's own paired difference is -13.4, while the other five hosts average -0.56. With each member as the sole fixed aggregator across twelve suites, Phi-4 is strongest on average (60.8) and matches the plain vote (60.7). Per aggregator on MATH-500, the prefix synthesis-minus-vote difference ranges from -0.2 $[-2.8, +2.4]$ (Phi-4) and -0.8 $[-3.4, +1.8]$ (Qwen3-14B) to -20.2 $[-24.2, -16.4]$ (Mistral-7B); as under full input, Qwen3-14B and Phi-4 fare best and Mistral-7B worst.

	single model		self+syn.	team, one answer each		team, adaptive sampling
Benchmark	SINGLE	SC-MAJ	SELF-MOA	MOA	CAMP-1	CAMP
MATH-500	59.5	64.1	61.4	72.5	81.8	**84.8**
MMLU-Pro	53.2	56.3	55.7	64.1	66.4	**68.0**
GPQA	38.7	41.2	40.5	43.7	51.8	**54.0**
HumanEval	68.1	71.0	70.9	76.3	**85.7**	–
MMLU	74.2	76.2	74.6	81.3	83.9	**84.1**
CSQA	76.8	77.8	74.7	81.0	83.0	**83.6**
AQuA	69.0	73.1	70.9	83.0	86.6	**87.4**
MedMCQA	58.8	59.7	58.5	66.8	69.2	**69.6**
TriviaQA (free-text)	48.6	53.0	51.3	55.2	**63.8**	–
GSM8K	84.9	88.4	87.2	94.4	95.8	**96.8**
AIME 24–25	8.3	10.6	7.8	8.6	15.0	**21.7**
AIME 1983–2024	17.8	–	–	20.2	34.3	**42.7**
OlympiadBench	34.3	–	–	41.2	**55.3**	54.3
Omni-MATH	24.5	–	–	27.9	**34.4**	–

Table 5: **Fourteen-task grid, prefix-limited pipeline** (accuracy, %). Host-based arms average the six models; MOA averages the six aggregators' synthesis outputs; CAMP-1 is the size-ordered vote. The adaptive CAMP column comes from a separate checkpointed pass with additional samples; costs are not matched. "–" marks an unavailable cell.

Scoring checks. A CPU rerun of the archived verifier with `math-verify` 0.9.0 reproduces all 6,000 stored MATH proposal and synthesis flags. Applying a stricter final-answer extractor identically to both arms, with camps and outputs frozen, moves the prefix MATH-500 difference from -9.13 to -13.17 (literal) or -10.67 (formatting-aware); the direction is unchanged and the magnitude is extractor-dependent. Other suites use stored verifier flags.

C SAMPLING SCHEDULES AND COST CONTROLS

C.1 CHECKPOINTED SAMPLING AND TOKEN COSTS

Table 6 supplies the twelve-suite CAMP–flat-30 comparison. Table 7 reports the separate cost-instrumented pass. The latter is used for cost plots, not to replace the former's intervals.

	Accuracy		Tokens / item				
Benchmark	CAMP	flat 30	CAMP	flat 30	Cost	Esc. %	$\Delta_{\text{2T-30}}$ 95% CI
GSM8K	96.8	96.8	2.5k	8.4k	30%	10%	$[-0.8, +0.8]$
CSQA	83.6	83.4	1.7k	6.0k	28%	9%	$[-0.8, +1.2]$
MMLU	84.1	84.7	2.4k	6.9k	35%	16%	$[-1.9, +0.7]$
AQuA	87.4	87.8	3.9k	9.8k	40%	18%	$[-2.4, +1.6]$
MedMCQA	69.6	69.2	3.0k	6.7k	44%	29%	$[-1.6, +2.6]$
MMLU-Pro	68.0	68.7	4.6k	9.2k	49%	30%	$[-2.6, +1.1]$
MATH-500	84.8	85.0	11.2k	18.3k	61%	40%	$[-1.6, +1.2]$
GPQA	54.0	56.6	7.8k	13.6k	57%	45%	$[-7.1, +1.5]$
OlympiadBench	54.3	56.3	20.7k	25.4k	82%	70%	$[-5.3, +1.7]$
AIME 83–24	42.7	41.3	26.9k	30.3k	89%	83%	$[-1.3, +4.3]$
AIME 24–25	21.7	25.0	28.8k	31.2k	92%	92%	$[-8.3, +0.0]$
HLE-Math	5.0	4.2	20.2k	23.7k	85%	80%	$[-1.1, +2.8]$

Table 6: CAMP vs. the flat 30-vote budget on all twelve suites (fresh tier-2 samples, seeds disjoint from tier 1; measured completion tokens).

	fixed single SC-5		MOA (pool)		CAMP-1		CAMP (margin)		flat 30	
Benchmark	acc	tok	acc	tok	acc	tok	acc	tok	acc	tok
GSM8K	96.4	1.5k	94.4	2.4k	95.8	1.7k	96.8	2.5k	97.0	8.4k
MMLU	87.3	2.4k	81.3	2.0k	83.9	1.4k	84.1	2.4k	83.9	6.9k
MedMCQA	70.0	2.7k	66.8	1.9k	69.2	1.3k	69.6	3.0k	69.0	6.7k
MMLU-Pro	69.6	2.5k	64.1	2.5k	66.4	1.8k	68.0	4.6k	68.3	9.2k
GPQA	57.6	3.0k	43.7	3.7k	51.5	2.8k	54.0	7.8k	53.5	13.6k
MATH-500	82.6	3.3k	72.5	4.7k	82.0	3.7k	84.8	11.2k	85.0	18.3k
OlympiadBench	51.7	4.3k	41.2	6.9k	55.0	5.1k	54.3	20.7k	54.7	25.5k
AIME 83–24	32.3	5.5k	20.2	8.4k	34.3	6.1k	42.7	26.9k	42.7	30.4k
Mean	68.4	3.2k	60.5	4.1k	67.3	3.0k	69.3	9.9k	69.3	14.9k

Table 7: **Accuracy and cost, priced per one team answer** (measured completion tokens; aggregator prefill discounted $4\times$). Every method is charged for all generations it consumes: CAMP-1 for six proposals, MOA for the same six *plus* its aggregation call, CAMP for the escalation samples it actually draws. MOA is the prefix-limited pipeline (Appendix B.3); full-input synthesis costs were not remeasured. Adaptive sampling trades additional tokens for accuracy; a strong *fixed* single model stays competitive on knowledge and science (§4).

C.2 BUDGET ATTRIBUTION

Given CAMP's *measured* per-benchmark budget, uniform allocation has no detected accuracy difference (-2.2 to $+1.9$, mean ≈ 0). This control does not establish superior item-level allocation by the margin. It is supplied with the average budget observed for each benchmark, whereas CAMP determines its sampling depth from the answers alone.

C.3 A DEVELOPMENT-SELECTED MEMBER ON THE MAIN POOL

On the main pool's twelve held-out suites, the development-set router reaches 70.59 at $2{,}685$ completion tokens per item, against CAMP's 70.45 at $8{,}130$. Router minus team is $+0.13$

$[-0.78, +1.06]$ points: the router spends one third as many tokens without a detected accuracy difference.

D SELECTED-POOL BUDGET AND INFORMATION CONTROLS

D.1 POOL CONSTRUCTION AND FIXED PROTOCOL

Eleven candidates comprise the nine-model sweep pool plus Qwen3-1.7B and Qwen3-0.6B. The pool rule averages development SC-5 accuracy over twelve suites, excludes the globally strongest candidate, then selects the six-member subset minimizing the best-minus-second gap; ties minimize best-minus-mean of the rest. It selects GLM-4-9B, InternLM3-8B, Llama-3.1-8B, Phi-4-mini, Qwen3-1.7B and Qwen2.5-7B. Pool construction uses development labels; the voting rule requires no further correctness labels at inference.

The aggregate development gap is 0.235, but only five of twelve suites have a per-suite gap below one point (Table 8). The median per-suite gap is 1.27 and the mean is 2.47. Thus this pool provides a fixed population for the budget and label comparisons, not a replication of the sweep's flattest bucket. The released development matrix reproduces the original pool selection.

Suite	Dev items	Top-two gap	Best member(s) in selected pool
GSM8K	250	1.20	Qwen2.5-7B
CSQA	250	0.00	GLM-9B, Qwen2.5-7B
MMLU	350	0.86	Qwen2.5-7B
AQuA	127	3.94	Qwen2.5-7B
MedMCQA	250	0.40	Llama-8B
MMLU-Pro	350	5.43	GLM-9B
MATH-500	250	6.80	InternLM-8B
GPQA	99	7.07	Qwen2.5-7B
OlympiadBench	150	1.33	InternLM-8B
AIME 83–24	150	2.00	Qwen2.5-7B
AIME 24–25	30	0.00	Phi-4-mini, Qwen2.5-7B
HLE-Math	180	0.56	Qwen3-1.7B

Table 8: Frozen selected pool under the per-suite dominance definition. Both tied champions are listed; a zero gap does not choose between them. The complete six-member development matrix and item IDs are released with the offline replay.

The 2026-08-16 recorded protocol (`sha256 d331ccda...`) fixed the candidate list, pool rule, split and label budgets before generation. S1 compares team with a development-selected member, S2 with a test-optimal member, and S3 specifies these comparisons at matched token budgets. The original shallow-sampling release omitted S3; the later shared-library control below supplies it. Joint threshold/depth cost calibration was added after diagnosing early-stop saturation. The registered crossover prediction $N^* \in [25, 50]$ is not confirmed; the current paper reports the label curve rather than interpreting interval overlap as equivalence.

An AQuA prompt correction for two candidates preceded the final pool inputs and leaves the selected six members unchanged.

D.2 AN EXCLUDED STRONG-MEMBER CONTROL

Phi-4 and Qwen3-14B each generate five samples on the same $2{,}437$ held-out items as the original selected-pool pass, with matching prompts, temperature, top-p and caps, and a distinct generation seed stage. All arms score the first concrete response in the winning camp. These direct measurements (Table 9) ask whether restricting the candidate pool changes the practical comparison.

CAMP spends $2.65\times$ Phi-4's tokens and $4.36\times$ Qwen's. The Phi-4 comparison is sensitive to grading convention: applying archived any-correct camp credit to both arms gives -2.26 $[-4.87, +0.55]$ instead of -3.00 $[-5.24, -0.46]$. The main text uses concrete-response scoring. This control uses the original selected-pool generations; it is separate from the shared-library budget comparison below.

Arm	Accuracy	Tokens/item	CAMP minus arm, points [95% CI]
CAMP	64.67	7933	—
Phi-4 SC-5	67.67	2997	-3.00 $[-5.24, -0.46]$
Qwen3-14B SC-5	65.53	1821	-0.86 $[-2.88, +1.40]$

Table 9: Direct excluded-member control on twelve suites, item-weighted. Both arms use concrete-response scoring. Intervals resample benchmarks 10,000 times, seed 20260913. Costs are measured completion tokens.

D.3 DEEP-SAMPLING BUDGET AND LABEL CONTROLS

This pass reports S1 (team minus a router chosen from development labels) and S2 (team minus the per-suite test-optimal member), read at near-matched completion-token budgets. It retains the frozen six-member pool, split and five label budgets.

Samples and scoring. Seventy-two member–suite files contain 5–37 fresh samples per item, with all 2,437 held-out IDs matching the frozen split and no duplicate IDs. Generation uses temperature 0.6, top-p 0.95, and caps of 1,280 tokens for mathematics and 1,024 for MCQ; a distinct seed stage separates this pass from the original five-sample pass. Both teams use the first five samples of these libraries, and the single-member controls consume prefixes of the same records. Every arm scores the first concrete response in its winning camp, using stored verifier flags. Pool order is GLM-9B, InternLM-8B, Llama-8B, Phi-4-mini, Qwen3-1.7B, Qwen2.5-7B; plurality ties use first occurrence. CAMP orders its first six proposals in this order, followed by each member's four additional proposals when escalation is triggered. Team accuracies are 61.26 and 62.91, at 2,313 and 7,750 tokens per item for CAMP-1 and CAMP. The original selected-pool pass scores 64.67; all comparisons here use the new shared library, not a splice between these generations.

Cost calibration and its limits. For each member and team budget within a suite, we enumerate stopping thresholds $\{2, 3, 4, 6, 8, 12, 16, \infty\}$ and depth caps $1, \ldots, K$. Sampling stops at the first prefix of at least two samples whose margin reaches the threshold, or at the cap (including a one-sample cap). We choose the configuration with the largest mean cost not exceeding the team's; ties retain enumeration order. Correctness does not enter this choice. The calibration does use held-out generation lengths, so it is a retrospective cost diagnostic, not a fully development-calibrated deployment policy. Fixing the threshold at two saturates early on high-agreement suites; increasing the depth cap alone need not spend the available budget. Joint threshold/depth enumeration was introduced after diagnosing that shortfall, rather than frozen in the registration.

Per member and suite, the chosen configurations spend 0.542 to approximately 1.000 of the team's tokens. Aggregate ratios in Tables 10 and 11 divide total single-member tokens by total team tokens; they are not averages of suite-level ratios. K was planned using the original pass's mean sample lengths. In the new pass, $11/72$ CAMP cells cannot reach the team budget even at full depth (full-library ratios 0.934–0.998). These gaps are distinct from discretization between reachable configurations.

Labels, hindsight selection, and inference. The router retains the frozen development SC-5 camp-credit scores, without accessing test correctness. We use 20 label draws at each N, truncating N to the available development items; random tie-breaking is retained, including at all-dev. The replay holds these draws and choices fixed when changing test-side scoring. At $N = 100$, GPQA (99 development items) and AIME 24–25 (30) use all available items; HLE-Math has 180 and is not truncated at this budget. For S2 we reselect the highest-accuracy member separately under each budget and scoring convention. This is a test-label oracle, not an achievable router, and selecting its winner on the evaluated test set can make it optimistic. Point estimates weight suites by held-out item count. Intervals resample the twelve benchmarks 20,000 times (seed 20260914), conditional on saved generations and routing draws. The near-matched S2 jackknife-t_{11} intervals remain negative: $[-3.66, -1.18]$ for CAMP-1 and $[-3.35, -0.34]$ for CAMP.

Label-free baseline and selection optimism. A member drawn uniformly at random, read at its own budget-matched configuration, scores 56.90 and 58.23 against the CAMP-1 and CAMP budgets. The teams exceed it by $+4.36$ $[+3.11, +5.39]$ and $+4.67$ $[+3.24, +5.77]$, and the router exceeds it

already at $N = 10$, by $+2.18$ $[+1.30, +2.96]$ and $+2.82$ $[+1.60, +4.01]$, rising to $+4.77$ and $+4.93$ with all development data. The test-optimal reference picks the best of six members on the evaluated items. Its bootstrap optimism (in each of 2,000 item resamples per suite, the resample-best member's resampled minus original accuracy) averages 0.81 and 0.85 points. Subtracting it per suite gives S2 differences of -1.61 $[-2.67, -0.63]$ for CAMP-1 and -0.99 $[-2.38, +0.04]$ for CAMP (jackknife-t_{11} $[-2.85, -0.36]$ and $[-2.41, +0.43]$). Configurations are chosen by cost alone, so the correction concerns member choice only.

Reading the label curve. At $N = 25$, both team–router intervals first contain zero; their point estimates first become nonpositive at all-dev. Neither event establishes population equivalence. The registered wording "ceases to favour the team" did not distinguish these criteria, so we report the full curve rather than a confirmed crossover threshold.

Scoring and cost sensitivities. The initial S3 script used archived any-correct camp credit and a different CAMP-1 tie order; restoring concrete-response scoring and the original pool order moves the S2 differences by at most 0.45 points, and both remain negative.

For a cost-only sensitivity, take the closest attainable costs below and above the target B, denoted c_L, c_H, and select the higher-cost configuration with probability $(B - c_L)/(c_H - c_L)$, independently of correctness. The resulting mixture matches *expected* cost on the fixed evaluation set; an individual run need not spend exactly B. All 72 CAMP-1 cells can be matched this way. The 11 insufficient CAMP libraries remain at full depth; its S2 reference reaches 99.5% of total team tokens. Both S2 estimates remain negative (Table 11). More sampling is not guaranteed to improve accuracy, so underspending is not assumed to conservatively favor the team. The mixture is supplementary and was not pre-specified.

The shallow comparison in the same table caps adaptive SC at five samples with threshold two. Its CAMP estimate of $+0.53$ does not exclude zero; its change to -1.85 at near-matched budgets shows the dependence on sampling budget. Member identity is reselected at each budget, so the change also includes differences in which member is test-optimal. Per-suite deltas, configurations, sample keys, flags and costs are included in the CPU replay artifact.

Policy	N	Team − router [95% CI]	Token ratio	Team accuracy
CAMP-1	10	$+2.19$ $[+0.96, +3.30]$	0.968	61.26
CAMP-1	25	$+1.03$ $[-0.68, +2.36]$	0.981	61.26
CAMP-1	50	$+0.94$ $[-0.69, +2.27]$	0.978	61.26
CAMP-1	100	$+0.22$ $[-1.38, +1.73]$	0.982	61.26
CAMP-1	all dev	-0.41 $[-2.25, +1.49]$	0.979	61.26
CAMP	10	$+1.85$ $[+0.52, +2.82]$	0.983	62.91
CAMP	25	$+0.91$ $[-0.52, +2.01]$	0.985	62.91
CAMP	50	$+0.77$ $[-0.69, +1.85]$	0.985	62.91
CAMP	100	$+0.29$ $[-1.16, +1.45]$	0.985	62.91
CAMP	all dev	-0.26 $[-2.09, +1.35]$	0.987	62.91

Table 10: S3 label-budget comparisons on the deep-sampling pass. Concrete-response scoring, original pool order, 20 label draws; 20,000 benchmark resamples, seed 20260914. Ratios are total single-member completion tokens divided by total team tokens.

Policy	Single-member budget	Team − test-optimal [95% CI]	Token ratio
CAMP-1	Margin 2, max 5	-1.11 $[-2.45, +0.03]$	0.622
CAMP-1	Near-matched	-2.42 $[-3.50, -1.49]$	0.986
CAMP-1	Cost mixture	-2.29 $[-3.41, -1.30]$	1.000
CAMP	Margin 2, max 5	$+0.53$ $[-1.40, +2.22]$	0.186
CAMP	Near-matched	-1.85 $[-3.31, -0.76]$	0.985
CAMP	Cost mixture	-1.83 $[-3.23, -0.79]$	0.995

Table 11: Same-pass S2 controls. The test-optimal member is reselected under each budget. The cost mixture matches expected cost in all 72 CAMP-1 cells; 11 CAMP cells remain below target because their full libraries are insufficient.

D.4 DEVELOPMENT SCORING AND SEPARATED SELECTION CHECKS

These sensitivities align development scoring and separate selection from evaluation using the existing generation libraries.

Development scoring. We rescore development SC-5 from any-correct camp credit to the first concrete response, retaining the original pool, label subsets, random tie values and test cost configurations. This changes 27 member–item scores and $52/2{,}400$ routing choices (2.17%), none at all-dev; the label-curve pattern persists: at $N = 10$ the gains become $+2.14$ $[+0.86, +3.31]$ and $+1.80$ $[+0.38, +2.80]$, and from $N = 25$ onward every interval contains zero. Reapplying the eleven-candidate pool rule still excludes Phi-4 and selects the same six members.

Separating selection, cost calibration and evaluation. Within each benchmark's S3 held-out library, shuffle sorted item IDs and split in half. On the calibration half only, measure the team's mean completion-token budget and choose each member's highest-cost configuration below it from thresholds $\{2, 3, 4, 6, 8, 12, 16, \infty\}$ and depths $1, \ldots, K$. No correctness is used to choose the configuration. Select the member with highest calibration accuracy at its chosen budget, breaking ties by pool order. Freeze member, threshold and depth; evaluate on the other half without retuning, then swap halves to cover all $2{,}437$ items. A companion check selects by calibration SC-5, keeping the same cost configurations, to isolate the depth-dependent selection target.

Use 20 prescribed seeds (20260915–20260934). Average repeated splits within benchmark before bootstrapping the twelve benchmarks; splits are not independent tasks. All new intervals use $20{,}000$ benchmark resamples, seed 20260915, conditional on saved generations.

	Team − selected member [95% CI]		Single/team tokens	
Selection / splits	CAMP-1	CAMP	CAMP-1	CAMP
Budget-specific, mean of 20	-1.11 $[-2.21, +0.04]$	-0.47 $[-1.60, +0.41]$	98.08%	97.99%
SC-5 selector, mean of 20	-0.24 $[-1.61, +1.32]$	$+0.17$ $[-1.26, +1.58]$	97.54%	98.03%

Table 12: Selection and evaluation on separate halves of the same existing library. Accuracy differences are points. Aggregate token ratios do not imply a strict cap on every evaluation fold.

In Table 12, the 20-split budget-specific mean retains a negative point estimate for both teams, but both intervals include zero. Budget-specific and SC-5 member choices differ in $156/480$ and $186/480$ folds for CAMP-1 and CAMP. Thus aligning the sampling target can matter, without establishing a universally better router.

This check uses roughly 15–175 additional labelled calibration items per suite and deep records for all six members. It is not the original $N = 10$ condition, a new external test set, or an end-to-end cost comparison. Calibration sampling and pool construction costs are excluded, and frozen configurations exceed the evaluation-half budget in 186 and 188 of 480 suite–folds; we neither retune nor discard them.

Small-cluster sensitivity. The same saved per-benchmark differences also support delete-one-benchmark jackknife standard errors, with a t_{11} critical value, reported alongside the percentile intervals in Table 13. The $N = 10$ gains remain positive under either development scoring rule; the separated-selection comparisons still include zero.

Comparison	CAMP-1, jackknife-t_{11}	CAMP, jackknife-t_{11}
$N = 10$, frozen development scores	$[+0.77, +3.61]$	$[+0.51, +3.20]$
$N = 10$, concrete development scores	$[+0.66, +3.62]$	$[+0.41, +3.19]$
Budget-specific, mean of 20 splits	$[-2.39, +0.18]$	$[-1.63, +0.69]$

Table 13: Small-cluster sensitivity for the label and separated-selection claims. Differences are team minus member, in percentage points, conditional on saved generations and selection draws.

E POOL DOMINANCE, ROUTING CONTROLS, AND LABEL COSTS

Construction. The sweep combines six main-campaign members and three additional members on the same twelve item sets. Each member's samples come from one campaign, under the same decoding settings; all four- to six-member subsets produce 336 pools per suite and 4,032 cells. Except for the explicitly re-scored Figure 6 comparison, this appendix preserves the archived camp-credit scoring of the sweep. The models' full-suite SC-5 accuracies are in Table 14; the policy comparisons use held-out halves.

suite	n	Qwen3 14B	Phi-4	GLM-4 9B	IntLM3 8B	Llama3 8B	Mist. 7B	Qwen3 8B	Phi-4 mini	Qwen2.5 7B
GSM8K	500	96.2	96.2	93.8	93.4	90.4	60.2	94.8	91.0	93.2
CSQA	500	80.2	82.0	83.6	77.2	76.2	68.0	80.0	73.6	84.4
MMLU	700	81.1	86.6	77.4	77.4	73.7	57.7	78.9	72.7	77.1
AQuA	254	83.5	83.5	79.5	80.7	71.7	43.3	83.9	73.2	80.7
MedMCQA	500	64.8	68.2	57.0	59.6	62.4	48.2	60.4	54.0	54.6
MMLU-Pro	700	65.7	69.6	58.6	53.7	51.0	35.9	63.9	52.3	54.6
MATH-500	500	86.2	82.4	73.0	79.0	55.4	18.8	83.0	63.6	73.6
GPQA	198	47.0	55.6	37.9	37.9	38.4	31.3	44.4	29.8	37.9
OlympiadBench	300	58.3	51.0	44.7	47.0	22.7	4.0	44.7	26.3	39.0
AIME 83–24	300	40.7	33.0	18.3	24.7	11.7	1.0	34.3	10.3	24.3
AIME 24–25	60	25.0	21.7	6.7	8.3	1.7	0.0	21.7	6.7	10.0
HLE-Math	361	4.2	4.7	4.4	4.7	4.4	5.0	3.0	3.9	3.9

Table 14: Per-(suite, model) SC-5 accuracy on the full common item set of the nine models (the sweep itself scores the held-out half of these items), from the sweep generation pass. These are not the main-protocol SC-5 records used in the within-model analysis (MATH Qwen3-14B is 83.8 there). Mistral-7B's 0.0 on AIME 24–25 (n=60) is a capability result: 1 of its 150 samples is correct, so its modal answer never is.

E.1 DOMINANCE CONTRASTS AND SCORING SENSITIVITY

dominance gap	$[0,1)$	$[1,3)$	$[3,6)$	$[6,12)$	$[12,\infty)$
Archived camp credit	$+0.84$	$\mathbf{+1.47}$	$+0.23$	-0.38	$\mathbf{-2.16}$
Archived 95% CI	$[-0.17,+2.85]$	$[+0.65,+2.57]$	$[-1.21,+1.55]$	$[-1.80,+0.92]$	$[-3.77,-1.45]$
Concrete response	$+0.65$	$+1.31$	-0.07	-0.83	-2.56
New 95% CI	$[-0.23,+2.36]$	$[+0.46,+2.33]$	$[-1.24,+1.10]$	$[-2.09,+0.27]$	$[-4.29,-1.68]$
Recomputed dev anchors	$+0.51$	$+1.21$	-0.02	-1.10	-2.32
Recomputed 95% CI	$[-0.39,+2.37]$	$[+0.39,+2.24]$	$[-1.12,+1.08]$	$[-2.23,-0.04]$	$[-3.81,-1.65]$

Table 15: CAMP minus labelled router. Figure 6 uses the concrete-response rows, with 20,000 benchmark resamples and seed 20260913. The original rows retain their reported intervals. Pool membership, development-based anchors, gap buckets, outputs and stopping remain fixed; the last two rows instead rescore development SC-5 by concrete response and recompute anchors and buckets (Appendix E.1).

Recomputing development anchors and buckets. A supplementary replay also changes development SC-5 from any-correct camp credit to the first concrete response, recomputing selection and gap buckets. The old anchors and rounded gaps reproduce exactly before this change. Of 4,032 cells, 66 anchors change and 215 migrate between buckets; pools, splits, generations and bucket edges remain fixed. The last two rows of Table 15 report this sensitivity (seed 20260915). Neither reselecting nor rebinning establishes a flat-end advantage, and the barely negative $[6,12)$ interval is not a transferable crossover threshold.

E.2 SENSITIVITY TO POOL COMPOSITION AND SELECTION NOISE

The 4,032-cell sweep of §5 crosses all k-member sub-pools ($k \in \{4,5,6\}$) of nine models with the twelve suites, replayed offline from logged generations at zero additional generation cost.

Controls for suite and mean pool ability. Computed *within* each suite, the correlation between dominance gap and team advantage is negative on 11 of the 12 suites (mean -0.357); only HLE-MATH, the coverage floor, is positive ($+0.401$). A suite fixed-effects regression with suite-clustered

standard errors gives a slope of -0.310 ($t = -4.87$) on the gap, and adding pool mean ability—the members' mean per-suite SC-5 accuracy, computed independently of the dependent variable—leaves it untouched (-0.312, $t = -4.75$; ability itself $t = -0.66$). The association remains under these controls; they do not exclude all model-family or error-structure confounding.

Gap or spread? The dominance gap and the best-minus-mean-of-rest spread are collinear ($r = +0.71$), and we do not claim the gap is the uniquely operative statistic. Entered together both survive (-0.212, $t = -2.90$ and -0.116, $t = -2.75$); in the fully saturated specification with pool ability the spread retains significance (-0.209, $t = -3.92$) while the gap becomes marginal (-0.143, $t = -1.73$). We organize the analysis by the gap as an interpretable separation between the two leading members: on spread buckets $[0, 2, 5, 10, 20)$ the advantage runs $+0.43$, $+0.88$, $+0.62$, $+0.26$, -0.18, against $+0.84$, $+1.47$, $+0.23$, -0.38, -2.16 on gap buckets.

Correcting the router's selection noise. The dominance gap and the dev-set router are estimated on the same development half, so where the gap is small the champion is close to a coin flip and the *selected* model is systematically worse on test than the pool's best. On the cells where every member's routed accuracy is identifiable, measuring against the test-optimal member instead of the chosen one moves the five buckets from $+0.79/+1.44/+0.27/+0.29/-0.59$ to $-0.02/+0.46/+0.23/+0.29/-0.59$: most of the flat-end advantage is the router's own selection regret, while the dominated end is untouched. This separates a routing-favorable dominated regime from small-gap comparisons affected by member-selection error.

Bucket composition and HLE-Math. Each bucket draws on nine to eleven of the twelve benchmarks; the largest single share is CSQA's 36.4% of the flat bucket. HLE-Math is the coverage floor: all nine models score 3.05–4.99, so its dominance gaps are largely sampling noise, and it supplies 22.2% of bucket $[1, 3)$. Dropping it changes that bucket's archived estimate from $+1.47$ to $+1.57$.

E.3 Agreement-based routing and development labels

Agreement recovers the development-labelled anchor on $47.5/39.2/68.7/72.9/83.4\%$ of cells across the five buckets. Conditional on choosing a different member, the losses are $3.73/1.99/6.49/8.00/19.89$ points. Marginal loss is miss rate times conditional loss because matched anchors produce identical routed answers. This is an accounting identity. Missed cells also have increasing mean dominance ($0.38/1.63/3.62/7.26/16.70$), so greater absolute loss does not establish greater loss relative to what was at stake. A label-budget sweep draws 5, 10 or 25 labelled development items (20 draws each), a cap that retains all twelve suites. Overall router accuracy is 63.06, 64.20 and 65.38, and 67.09 with all development data, against CAMP 67.46 and the plain vote 65.46 in the same pass. In the most dominated bucket, labels buy a 2.16-point accuracy advantage over CAMP; in flatter buckets they mainly buy lower generation cost.

E.4 Budget-capped routing

Deeper samples exist for five anchors on four suites and for Phi-4 and Qwen3-14B on all twelve. The control restricts the four-suite population to the same two anchors. For each population the routing frontier is read at a cost no greater than CAMP's measured spend.

Routing is cheaper in all $3{,}579$ cells; median router-to-team token ratios are 56%, 61%, and 57%. With anchors fixed, extending four suites to twelve moves the flat-bucket difference from $+0.70$ to -0.09. Overall intervals include zero, and the two four-suite populations are non-monotone across buckets. The twelve-suite population contains only twelve CSQA cells and sixteen HLE-Math cells, reflecting how rarely its two available anchors are the development champion there. The controls therefore do not establish an equal-cost flat-end advantage or a transferable crossover threshold.

dominance gap	$[0,1)$	$[1,3)$	$[3,6)$	$[6,12)$	$[12,\infty)$	all [95% CI]
four suites, five depth anchors (1,003 cells: GSM8K 240, MATH-500 279, MMLU-Pro 240, GPQA 244)						
CAMP	90.45	88.40	76.98	69.64	56.99	77.32
routing, budget-capped	90.24	88.40	76.67	70.81	64.22	78.26
difference	$+0.21$	-0.01	$+0.31$	-1.17	-7.23	-0.93 $[-3.66, +0.12]$
twelve suites, Phi-4 / Qwen3-14B anchors (1,848 cells)						
CAMP	84.13	91.71	76.02	66.52	52.09	71.84
routing, budget-capped	84.21	91.93	78.34	70.47	56.33	74.53
difference	-0.09	-0.23	-2.32	-3.95	-4.24	-2.69 $[-6.29, +0.38]$
control: four suites, restricted to the same two anchors (728 cells)						
difference	$+0.70$	$+0.10$	$+0.32$	-0.93	-7.23	-1.10 $[-4.92, +0.47]$

Table 16: Accuracy under a team-budget cap on three populations. Differences are CAMP minus routing, with reported benchmark-clustered intervals on overall differences. Routing always spends less than the cap.

F COMPLEMENTARITY AND TRAINING OVERLAP

F.1 ERROR DIVERSITY AND POOL COMPOSITION

Cross-model sampling improves over within-model sampling by -0.5 to $+17.7$ points (median $+9.8$) at approximately six generations; the negative endpoint is at the coverage floor. Error-indicator correlation ϕ on the three core suites is $0.60/0.41/0.64$ for within-model samples and $0.37/0.17/0.40$ across models. Leave-one-out costs at most 6.6 points, and the most consequential member depends on the task. These comparisons establish complementarity relative to average self-sampling, not superiority to the strongest available member.

F.2 TRAINING-DATA OVERLAP AND TRANSFER

Older benchmark items may overlap model training data. Pairing methods on the same pool controls the candidate source, but cannot guarantee that memorization affects selection and synthesis equally. Absolute accuracies should therefore be read with this limitation. Dominance is additionally a between-model quantity: if one model has memorized more items, its apparent routing advantage may not transfer. Splitting the sweep into seven older suites gives bucket differences $+0.88/+1.53/-0.24/-0.74/-3.56$; the three suites used for the lower-overlap sensitivity give $+0.17/+1.08/-3.99/+0.82/-1.95$. The endpoints agree in sign, but neither sequence is monotone. Three suites cannot distinguish training overlap from small-population variability, so clean-data transfer of the dominance diagnostic remains open.

G SCOPE WHEN REASONING-MODE MODELS ARE AVAILABLE

G.1 MIXED POOLS CONTAINING REASONING MEMBERS

Qwen3 in thinking mode at three sizes (14B, 8B, 1.7B) joined the nine non-thinking members on four shared suites, giving 8,844 (pool, suite) cells over pools of size 4–6. Among pools containing a thinking member, 53% have a development gap ≥ 6. There, CAMP minus the routed member is -4.80; on the rest it is $+0.44$. The bucket-level point estimates follow.

CAMP $-$ routed member	$[0,1)$	$[1,3)$	$[3,6)$	$[6,12)$	$[12,\infty)$	all
non-thinking pools (same items)	$+4.17$	$+2.11$	$+1.68$	$+0.54$	-1.43	$+0.99$
mixed pools	-0.42	$+0.91$	$+0.18$	-2.70	-7.76	-2.15
anchor is a thinking member	-0.81	-0.09	-0.71	-3.18	-8.44	-2.95

The dominated-end loss extends the main sweep pattern. At the flat end the team spends 55,495 tokens against the router's 19,398, a $2.9\times$ premium. These four-suite estimates do not establish accuracy equivalence or the prevalence of dominance in deployment fleets.

Decode-cap sensitivity. An inadequate decode cap can leave parseable but incorrect answers. For Qwen3-14B-thinking on GPQA, increasing the cap from 3k to 12k changed accuracy from 44.95

to 60.30. OlympiadBench required 24k to bring its at-cap rate below 3%. The reported results use corrected caps, with at-cap rates 0.3–2.7% and empty-answer rates 1.3–5.5% per (suite, model); the raw records behind these rates are not in the current release.

G.2 STRONG SINGLE-MODEL CONTROLS

Table 17 compares the non-thinking team vote with Qwen3-14B alone: five non-thinking samples, three thinking-mode samples, and JointThinking (Wu et al., 2025), which combines one thinking and one non-thinking pass and adds a second thinking round only on disagreement.

Benchmark	CAMP-1	NoThink-SC5	Always-think SC3	JointThinking
MATH-500	81.8	83.8	92.4 @8.5k	92.6 @7.0k
MMLU-Pro	66.4	65.4	74.7 @5.1k	74.4 @3.5k
MMLU	83.9	81.6	86.4 @1.9k	85.9 @1.1k
AIME 24–25	15.0	21.7	65.0 @27.0k	65.0 @27.7k
AIME 83–24	34.3	39.7	76.7 @22.2k	76.7 @21.0k
OlympiadBench	55.3	56.3	75.3 @17.7k	75.7 @16.7k

Table 17: **Thinking-mode controls** (accuracy, %, @ completion tokens per item). All single-model arms use Qwen3-14B.

When a reasoning-mode member is available, a single model exceeds the non-thinking team by 2.5 (MMLU) to 50 (AIME 24–25) points. JointThinking is within 0.5 points of always-thinking SC-3 on all six suites and uses fewer tokens on five. HLE-Math stays near the coverage floor (5.5) even with thinking-mode escalation. All main-line comparisons in the paper use non-thinking generation.